Grundkurs Informatik

Lösungen zu den Übungsaufgaben, 8. Auflage 2023

Hartmut Ernst, Jochen Schmidt und Gerd Beneken

Stand: 16. September 2024

Inhaltsverzeichnis

1 Lösungen zu Kapitel 1 – Einführung

Übungsaufgaben zu den Kapiteln 1.1 und 1.2

A 1.1 (T0) Aus welchen Wissenschaften ist die Informatik in erster Linie hervorgegangen? Welche weiteren Wissenschaften sind für die Informatik von Bedeutung?

Lösung

Historisch gesehen ist die Informatik in erster Linie aus der Mathematik und der Elektrotechnik hervorgegangen. Von Bedeutung sind daneben auch andere Ingenieurwissenschaften, außerdem Physik (insbesondere die Festkörperphysik), Linguistik und BWL sowie zahlreiche Disziplinen in speziellen Anwendungen, beispielsweise Medizin, Biologie und Medientechnik.

A 1.2 (T0) Was bedeutet „Kerninformatik“?

Lösung

Zur Abgrenzung gegen die angewandte Informatik fasst man die theoretische, praktische und technische Informatik unter dem Oberbegriff Kerninformatik (auch allgemeine Informatik) zusammen.

A 1.3 (T0) Grenzen Sie die Begriffe „technische Informatik“, „Technik der Informatik“ und „Informatik in der Technik“ voneinander ab.

Lösung

Arbeitsgebiet der „technischen Informatik“ oder treffender der „Technik der Informatik“ ist die Erforschung und Anwendung ingenieurwissenschaftlicher und physikalischer Grundlagen und Methoden für die Entwicklung und den Bau von Rechenanlagen und Peripheriesystemen. Die „Informatik in der Technik“ ist derjenige Teilbereich der angewandten Informatik, der sich mit der Entwicklung, Anwendung und Pflege von Programmen in technischen Anwendungen, beispielsweise im Maschinenbau, befasst. Im Sprachgebrauch ist manchmal die Abgrenzung zur technischen Informatik fließend.

A 1.4 (T0) Auf wen geht die binäre Arithmetik und Logik zurück?

Lösung

Erste Ideen stammen von Gottfried Wilhelm von Leibnitz (1646-1716), die detaillierte mathematische Ausarbeitung (Boolesche Algebra) lieferte George Boole (1815-1864).

A 1.5 (T1) Warum werden beim Morse-Alphabet manche Buchstaben mit kurzen und manche mit langen Folgen der Zeichen „.“ und „−“ dargestellt? Handelt es sich beim Morse- Alphabet um eine binäre Codierung?

Lösung

Um Zeit bei der Übertragung zu sparen, hat man damals schon, lange bevor es Computer gab, häufig auftretende Zeichen mit kurzen und selten auftretende Zeichen mit langen Folgen von „.“ und „−“ dargestellt. Beim Morse-Alphabet handelt es sich nicht um eine binäre Codierung, da neben „.“ und „−“ als drittes Zeichen die Pause verwendet wird.

A 1.6 (T0) Seit wann gibt es in Deutschland eigenständige Informatik-Studiengänge und wer hat die Entwicklung der Informatik in Deutschland maßgeblich geprägt?

Lösung

Ab ca. 1965 wurde Informatik als Spezialisierung in Mathematik- und Elektrotechnik-Studiengängen gelehrt. 1970 wurde an der Technischen Universität München (TUM) unter Leitung von F. L. Bauer mit dem ersten eigenständigen Studiengang Informatik in Deutschland begonnen.

A 1.7 (T0) Beschreiben Sie ausführlich den Begriff „4. Computergeneration“.

Lösung

Die ab ca. 1975 vorherrschende 4. Computergeneration ist durch den Einsatz von höchstintegrierten Schaltkreisen (Very Large Scale Integration, VLSI) und Mikroprozessoren (Intel) gekennzeichnet. Typische Vertreter dieser Generation sind Personal-Computer (Apple, Commodore, IBM) und erste Super-Computer der Firmen Control Data Corporation (CDC) und Cray. Auch elektronische Taschenrechner sind hier zu nennen. Kennzeichnend für die 4. Generation sind auf Software-Seite die Entwicklung des Betriebssystems MS-DOS durch Microsoft sowie die Entwicklung der Programmiersprache C und des Betriebssystems UNIX.

A 1.8 (T1) Charakterisieren Sie die Computergenerationen 0 bis 5 jeweils durch ein Schlagwort für das innovative Funktionsprinzip, geben Sie die ungefähre Zeit der ersten Einführung an und nennen Sie außerdem ein typisches Rechnermodell für jede Computergeneration.

Lösung

Generation	Innovatives Prinzip	Entstehungsjahr	Beispiel
0.	Elektromechanische Komponenten	1941	Z3, Mark1
1.	Elektronenröhren	1946	ENIAC, PERM
2.	Transistoren	1952	IBM 701
3.	ICs	1960	PDP8
4.	VLSI	1975	PCs, Großrechner
5.	Parallele Strukturen, KI	1985	Connection Machine

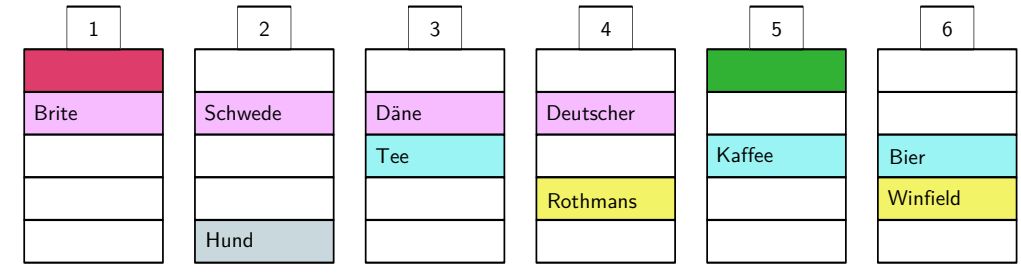
A 1.9 (L3) Die folgende Aufgabe stammt angeblich von Albert Einstein. Es wird behauptet, dass nur ca. 2% der Weltbevölkerung in der Lage seien, diese Aufgabe innerhalb von ca. einer Stunde zu lösen. Gehören Sie zu diesen schlauesten 2%? Zur Lösung ist kein Trick erforderlich, nur pure Logik.

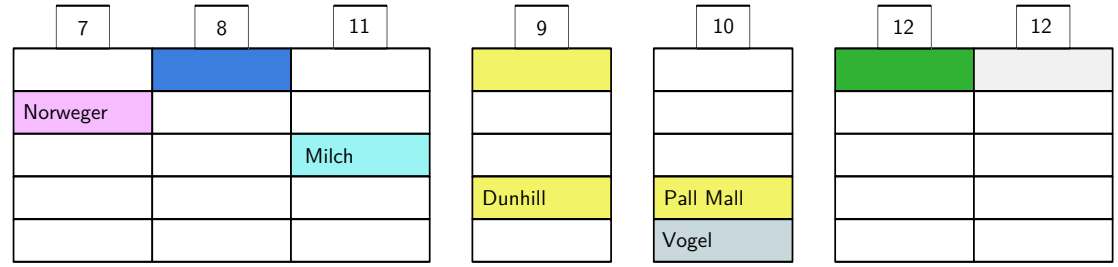
1. Es gibt fünf Häuser mit je einer anderen Farbe.
2. In jedem Haus wohnt eine Person einer anderen Nationalität.
3. Jeder Hausbewohner bevorzugt ein bestimmtes Getränk, hält ein bestimmtes Haustier und raucht eine bestimmte (zu Einsteins Zeiten populäre) Zigarettenmarke.
4. Keine der fünf Personen bevorzugt das gleiche Getränk, raucht die gleichen Zigaretten oder hält das gleiche Tier wie irgendeine der anderen Personen.
5. Der Däne trinkt gerne Tee.
   Der Brite lebt im roten Haus.
   Der Schwede hält einen Hund.
   Der Deutsche raucht Rothmanns.
   Der Norweger wohnt im ersten Haus.
   Der Winfield-Raucher trinkt gerne Bier.
   Der Besitzer des grünen Hauses trinkt Kaffee.
   Der Norweger wohnt neben dem blauen Haus.
   Der Besitzer des gelben Hauses raucht Dunhill.
   Die Person, die Pall Mall raucht, hat einen Vogel.
   Der Mann, der im mittleren Haus wohnt, trinkt Milch.
   Das grüne Haus steht unmittelbar links vom weißen Haus.
   Der Mann mit dem Pferd wohnt neben dem Dunhill-Raucher.
   Der Marlboro-Raucher wohnt neben dem, der eine Katze hält.
   Der Mann, der Marlboro raucht, hat einen Nachbarn, der Wasser trinkt.

Frage: Wer hat einen Fisch?

Lösung

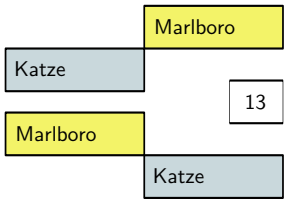
1. Die 15 Regeln kann man wie folgt grafisch darstellen:

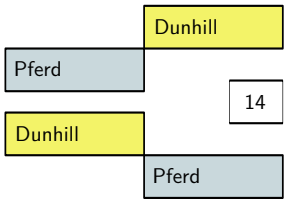


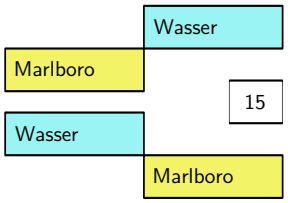


Erstes Haus, neben dem blauen. Mittleres Haus

Grünes Haus, links vom weißen.







2. Man versucht nun Regeln zu kombinieren:
Die Regeln 5 und 12 und die Regeln 1 und 11 können zusammengefasst werden. Damit liegt die Reihenfolge der Farben fest und Regel 9 lässt sich integrieren. Die Häuserzeile ist jetzt komplett.

7 9	8	1 11	5	12
Norweger		Brite		
		Milch	Kaffee	
Dunhill				

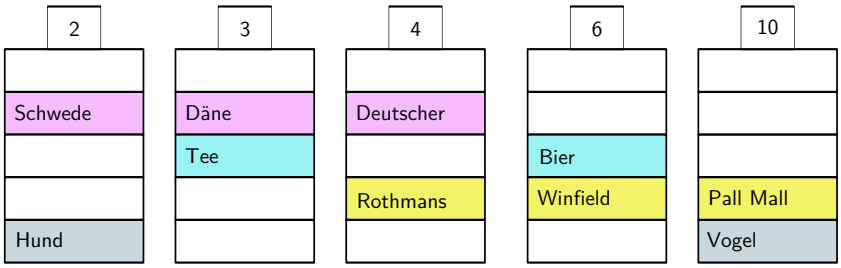
2	3	4	6	10
Schwede	Däne	Deutscher		
	Tee		Bier	
		Rothmans	Winfield	Pall Mall
Hund				Vogel

13	
Marlboro	
Katze	
	Marlboro
	Katze

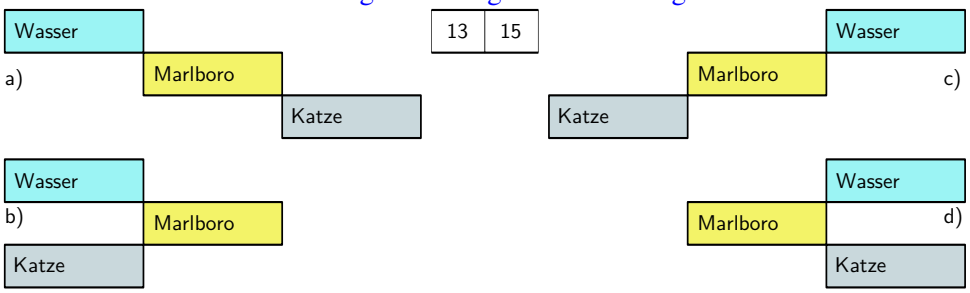
14	
Dunhill	
Pferd	
	Dunhill
	Pferd

15	
Wasser	
Marlboro	
	Wasser
	Marlboro

3. Weitere Kombinationen:
Regel 14 kann noch eindeutig eingefügt werden, aber weitere Regeln sind nicht eindeutig. Es stehen folgende Möglichkeiten zur Verfügung:
2 in 5 oder 12.
3 oder 6 in 14/8 oder 12.
4 in 14/8 oder 5 oder 12.
10 in 1/11 oder 5 oder 12.



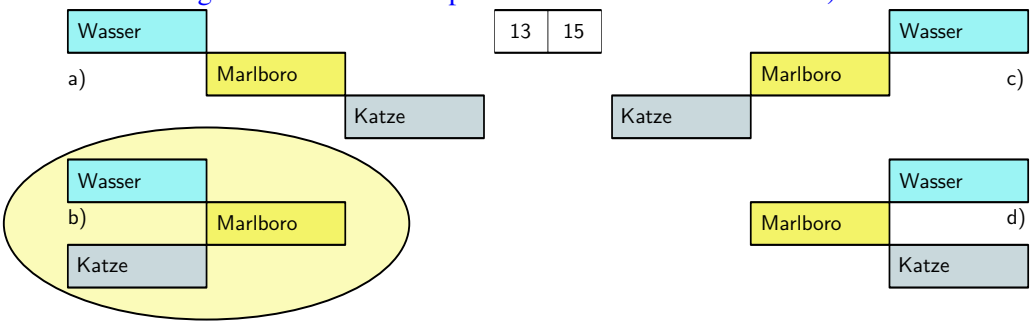
Kombination von 13 und 15 ergibt die folgenden vier Möglichkeiten:



4. Man probiert zwei Varianten, der Rest ist wieder eindeutig:
Für 2 gibt es nur die Alternativen 2 in 5 oder 2 in 12. Der Versuch 2 in 5 führt rasch zu einem Widerspruch.
Man setzt daher 2 in 12. Das weitere Vorgehen ist Jetzt wieder eindeutig. Man setzt ein:
3 in 14/8, 4 in 5, 6 in 12 und 10 in 1/11.

7	9	3	8	14	10	1	11	4	5	6	2	12
Norweger			Däne			Brite		Deutscher			Schwede	
			Tee			Milch		Kaffee			Bier	
Dunhill						Pall Mall		Rothmans			Winfield	
			Pferd			Vogel					Hund	

Von den vier Möglichkeiten für 13/15 passt offenbar nur die Variante b).



5. Die Lösung:
Fügt man noch 13/15 b) ein, so folgt, dass der Deutsche einen Fisch hat.

13	15	3	10	6
7	9	8 14	1 11	4 5	2 12
Norweger			Däne			Brite		Deutscher			Schwede	
Wasser			Tee			Milch		Kaffee			Bier	
Dunhill			Marlboro			Pall Mall		Rothmans			Winfield	
Katze			Pferd			Vogel					Hund	

Übungsaufgaben zu Kapitel 1.3

A 1.10 (T0) Was versteht man unter EVA, HIPO, DFÜ, 4GL, ADA, CASE und KISS?

Lösung

EVA: Eingabe, Verarbeitung, Ausgabe
HIPO: Hierarchical Input, Processing and Output, das englische Äquivalent zu EVA
DFÜ: Datenfernübertragung
4GL: Fourth Generation Language, Sprachen der vierten Generation (z. B. SQL)
ADA: Eine Programmiersprache, benannt nach Ada Countess of Lovelace
CASE: Computer Aided Software Engineering
KISS: Keep it stupid and simple

A 1.11 (T1) Was sind die typischen Unterschiede zwischen Digital-, Analog- und Hybridrechnern?

Lösung

Digitalrechner Zahlen sind diskret mit endlicher Stellenzahl als Binärzahlen dargestellt. Die Abarbeitung erfolgt seriell oder teilweise auch parallel Schritt für Schritt nach einem Programm, das einen Algorithmus implementiert. Digitalrechner sind die verbreitetste Art heutiger Rechner.

Analogrechner Zahlenwerte werden als analoge, d. h. kontinuierliche physikalische Größen dargestellt. In üblichen elektronischen Analogrechnern sind dies zeitlich veränderliche elektrische Spannungen und Ströme. Die Abarbeitung erfolgt entsprechend der Schaltung, die für jedes Problem neu konfiguriert werden muss. Anwendungsgebiete sind vor allem die Lösung von Differentialgleichungen sowie Simulationen komplexer Systeme. Werden heute praktisch nicht mehr verwendet.

Hybridrechner Verbindung von Digital- und Analogrechnern. Der Digitalteil dient dabei hauptsächlich zur Eingabe und Ausgabe sowie für die Festlegung der Konfiguration der Komponenten des Analogteils. Die Kommunikation zwischen Digitalteil und Analogteil erfolgt über D/A- und A/D-Wandler. Werden heute praktisch nicht mehr verwendet.

A 1.12 (M1) Ein Rechner habe 32 Datenleitungen und 21 Adressleitungen.

a) Wie viele Bits bzw. Bytes hat ein Wort dieser Anlage?
b) Wie lautet die größte damit in direkter binärer Codierung darstellbare Zahl?
c) Wie groß ist der Adressraum, d. h. wie viele Speicherzellen sind adressierbar?
d) Wie groß ist die maximal speicherbare Datenmenge in MByte?

Lösung

a) Ein Wort dieser Anlage umfasst 32 Bits bzw. 4 Bytes.
b) Das größte in direkter binärer Codierung darstellbare Wort besteht aus 32 Einsen:
$11111111\,11111111\,11111111\,11111111_2 = 2^{32} - 1 = 4\,294\,967\,295$
c) Die Anzahl der adressierbaren Speicherzellen ist $2^{21} = 2\,097\,152$
d) Die maximal speicherbare Datenmenge ist $2^{21} \cdot 4$ Byte $= 2^{10} \cdot 2^{10} \cdot 8$ Byte $= 8$ MByte

A 1.13 (M1) Eine Digitalkamera habe eine Auflösung von 3318×2488 Pixeln (Bildpunkten). Jedes Pixel besteht aus den Farbkomponenten r, g, und b, die jeweils ganzzahlige Werte zwischen $0 < r,g,b < 255$ annehmen können. Wie groß ist der Speicherbedarf in MByte für ein solches Bild? Welche Bandbreite wäre für die Filmdarstellung von 30 Bildern/s nötig?

Lösung

Speicherbedarf pro Bild: $3318 \cdot 2488 \cdot 3$ Byte = 24 765 552 Byte $\approx$ 23,6 MByte
Bandbreite für die Filmdarstellung von 30 Bildern/s:
$30 \cdot 24\,765\,552$ Byte/s $= 742\,966\,560$ Byte/s $\approx$ 708,5 MByte/s

A 1.14 (T1) Nennen Sie Kriterien zur Beurteilung der Leistungsfähigkeit eines Computers.

Lösung

Typ der CPU, Datenbusbreite, Adressbusbreite, Prozessor-Taktrate und Bus-Taktrate bzw. Busbandbreite, Benchmarks (MIPS, FLOPS, etc.), installierter Speicher, Art und Anzahl der Peripheriegeräte, Betriebssystem, verfügbare Software, Ausbaufähigkeit, Vernetzbarkeit, Ausfallsicherheit, Kompatibilität, Hersteller, Preis. Generell ist zu beachten, dass die Wichtung der verschiedenen Kriterien von der Anwendung und weiteren Umständen wie dem finanziellen Rahmen, dem Platzbedarf etc. abhängt.

A 1.15 (T0) Was versteht man unter dem Systembus eines Rechners?

Lösung

Alle zur Steuerung des Rechners nötigen Leitungen, z. B. Read enable, Write enable, Data ready etc.

A 1.16 (T0) Nennen Sie alle Ihnen bekannten Möglichkeiten zur Speicherung von Daten mit Hilfe eines Computers.

Lösung

Magnetische Aufzeichnung:	Festplatten, Magnetbänder
Optische Platten:	CD-ROM, DVD, Blu-Ray Disc
Halbleiterspeicher:	RAM, ROM, PROM, EPROM, EEPROM, Flash-EPROM, SSD

A 1.17 (M1) Berechnen Sie die maximal mögliche Datenrate in MByte/s für einen 64 Bit breiten und mit 180 MHz getakteten Bus.

Lösung

Man berechnet man die resultierende Datenrate r wie folgt:

$$r = 64 \text{ Bit} \cdot 180 \text{ MHz} = 64 \cdot 180 \cdot 10^6 \text{ Bit/s} = \frac{64 \cdot 180 \cdot 10^6}{8 \cdot 1024 \cdot 1024} \text{ MByte/s} \approx 1\,373 \text{ MByte/s}$$

Übungsaufgaben zu Kapitel 1.4

A 1.18 (M0) Wandeln Sie die folgenden Binärzahlen in oktale, hexadezimale und dezimale Darstellung um: 11 0101; 11 0111 0110 1001; 111,101 und 11 0101,0001 001.

Lösung

binär	11 0101	11 0111 0110 1001	111,101	11 0101,0001 001
oktal	65	33551	7,5	65,044
dezimal	53	14185	7,625	53,0703125
hexadezimal	35	3769	7,A	35,12

A 1.19 (M1) Wandeln Sie die folgenden Dezimalzahlen ins Binär-, Oktal- und Hexadezimalsystem um: 43; 6789; 26,4375; 102,375.

Lösung

dezimal	43	6789	26,4375	102,375
binär	101011	1101010000101	11010,0111	1100110,011
oktal	53	15205	32,34	146,3
hexadezimal	2B	1A85	1A,7	66,6

Für kleine Zahlen kann die Umwandlung am schnellsten mit Hilfe einer Tabelle der Potenzen der verwendeten Basen 2, 8 und 16 geschehen, die man mit etwas Übung (zumindest für Basis 2) auswendig weiß:

2^k	1	2	4	8	16	32	64	128	256	512	1024	2048	4096
8^k	1	8	64	512	4096	32768							
16^k	1	16	256	4096	65536								
2^{-k}	0,5		0,25		0,125		0,0625		0,03125				
8^{-k}	0,125		0,015625		0,001953125								
16^{-k}	0,0625		0,00390625		0,000244140625								

Für größere Zahlen verwendet man am günstigsten das Horner-Schema mit fortgesetzter Division durch die gewünschte Basis, also 2 für das Binärsystem, 8 für das Oktalsystem und 16 für das Hexadezimalsystem. Die gesuchte Stelle ist dann jeweils der Divisionsrest.
Bei der Umwandlung von Nachkommastellen muss man durch den Kehrwert der Basis dividieren, also mit der Basis multiplizieren. Dem Divisionsrest entspricht dann der Übertrag.
Die Umwandlung von Binärzahlen in Hexadezimalzahlen erfolgt am einfachsten durch Zusammenfassen von jeweils vier Binärstellen zu einer Hexadezimalstelle. Bei Vorkommastellen wird dabei mit dem niederwertigsten Bit begonnen und vom Komma aus nach links gearbeitet. Bei Nachkommastellen wird vom Komma aus nach rechts gearbeitet. Für die Umwandlung von Binärzahlen in Oktalzahlen fasst man jeweils drei Bit zusammen.

A 1.20 (M2) Wandeln sie 497,888 ins Fünfersystem und 768,3 ins Siebenersystem um.

Lösung

$497{,}888_{10} = 3442{,}421_5, 768{,}3_{10} = 2145{,}\overline{2046}_7$
Anmerkung: Einen periodischen Bruch wandelt man in einen echten Bruch um, indem man die Periode durch eine gleichlange Folge dividiert, die aus der höchsten Grundziffer gebildet wurde. Also beispielsweise: $0{,}\overline{2} = 2/4$ im Fünfersystem, $0{,}\overline{1234} = 1234/9999$ im Zehnersystem.

A 1.21 (M2) Führen Sie die folgenden binären Rechenoperationen durch:

$110101 + 11001$	$101{,}1101 + 1110{,}11$	$111011 + 11101$	$110{,}0111 + 1101{,}101$
$111011 - 10111$	$110{,}1001 - 101{,}11011$	$110001 - 1101101$	$1011{,}101 - 1001{,}1101$
$11011 \cdot 1011$	$101{,}11 \cdot 1011{,}101$	$1101 \cdot 1011$	$111{,}01 \cdot 1{,}0101$
$101000010 : 1110$	$1011{,}11 : 10{,}1111$	$111{,}01 : 10$	$1010{,}11 : 10{,}1$

Lösung

Addition:

110101	53	101,1101	5,8125	111011	59	110,0111	6,4375
+ 11001	25	+ 1110,1100	14,7500	+ 11101	29	+ 1101,1010	13,6250
= 1001110	78	= 10100,1001	20,5625	= 1011000	88	= 10100,0001	20,0625

Subtraktion:

Bei der Subtraktion $a - b$ wird im Zweierkomplement gerechnet. Dabei bleibt a unverändert. Von b wird zunächst das Stellenkomplement (St.K.) durch bitweise Inversion gebildet und dann daraus das Zweierkomplement (Zw.K.) durch Addition einer 1 in der letzten Stelle (LSB). Sodann werden a und das Zweierkomplement von b wie gewohnt addiert. Ein über die verwendete Stellenzahl hinausgehender Überlauf wird abgeschnitten. Hat die resultierende Zahl ein MSB von 0, dann ist sie positiv; bei einem MSB von 1 negativ. Der Zahlenwert des Ergebnisses kann dann durch Bildung des Zweierkomplements ermittelt werden.

0010111	23	0101,11011	5,84375	01101101	109	01001,1101	9,8125
1101000	St.K.	1010,00100	St.K.	10010010	St.K.	10110,0010	St.K.
1101001	Zw.K.	1010,00101	Zw.K.	10010011	Zw.K.	10110,0011	Zw.K.
0111011	59	0110,10010	6,56250	00110001	49	01011,1010	11,6250
= 0100100	36	= 0000,10111	0,71875	= 11000100	−60	= 00001,1101	1,8125
				00111011	St.K.		
				00111100	Zw.K. = 60		

Multiplikation:

$11011 \cdot 1011$	$27 \cdot 11 = 297$
11011	
11011	
11011	
100101001	Ergebnis

$10111 \cdot 1011101$	$5{,}75 \cdot 11{,}625 = 66{,}84375$
10111	
10111	
10111	
10111	
10111	
100001011011	$\rightarrow 1000010.11011$

$1101 \cdot 1011$	$13 \cdot 11 = 143$
1101	
1101	
1101	
10001111	Ergebnis

$11101 \cdot 10101$	$7{,}25 \cdot 1{,}3125 = 9{,}515625$
11101	
11101	
11101	
1001100001	$\rightarrow 1001{,}100001$

Division:

$101000010 : 1110 = 10111$ $322 : 14 = 23$ $1011{,}11 : 10{,}1111 = 100$ $11{,}75 : 2{,}9375 = 4$

-1110 $10111100 : 101111 = 100$

11000 -101111

-1110 00000000

10101

-1110

1110

-1110

0000

$111{,}01 : 10 = 11{,}101$ $7{,}25 : 2 = 3{,}625$ $1010{,}11 : 10{,}1 = 100{,}01\overline{0011}$ $10{,}75 : 2{,}5 = 4{,}3$

-10 $10101{,}1 : 101 = 100{,}01\overline{0011}$

11 -101

-10 00110

10 -101

-10 1000

010 -101

-10 110

00 -101

periodisch

A 1.22 (M1) Berechnen Sie $647 - 892$ im Dezimalsystem unter Verwendung der Zehnerkomplement-Methode.

Lösung

Die Subtraktion lässt sich in Analogie zur Zweierkomplement-Darstellung im Binärsystem auch in einem beliebigen anderen Zahlensystem in Komplement-Darstellung ausführen. Dabei wird das Stellenkomplement einer Zahl durch Ergänzen der einzelnen Ziffern auf die höchste Grundziffer bestimmt. So ist das Stellenkomplement von 4 im Zehnersystem $9 - 4 = 5$ und beispielsweise im Fünfersystem $5 - 4 = 1$. Führt man als Beispiel die Subtraktion $647 - 892$ im Dezimalsystem mit 4 Stellen unter Verwendung der Zehnerkomplement-Methode durch so ergibt sich:

9999

-0892

9107 Stellenkomplement von 892

$+\quad 1$

9108 Zehnerkomplement von 892

$+0647$

9755 Ergebnis im Zehnerkomplement, negativ

9999

-9755

0244 Stellenkomplement

$+\quad 1$

0245 Ergebnis $\rightarrow$ 9755 im Zehnerkomplement entspricht -245

Eine genauere Betrachtung der Rechnung zeigt, dass die Durchführung der Subtraktion in Zehnerkomplement-Darstellung offenbar nur eine andere Schreibweise ist: $999 - (647 + (999 - 892 + 1)) + 1 = 245$.

A 1.23 (T1) Warum ist die Subtraktion mit der Komplement-Methode gerade im Binärsystem so vorteilhaft?

Lösung

Das Zweierkomplement wurde eingeführt, um die Subtraktion durch Rückführung auf die Komplementbildung und die Addition zu vermeiden. Wie die vorherige Aufgabe zeigt, ist aber im Zehnersystem für die Komplementbildung eine Subtraktion erforderlich. Dies gilt auch für alle anderen Ziffernsysteme, mit Ausnahme des Dualsystems. Nur dort kann das Komplement sehr einfach durch bitweise Inversion und Addition einer 1 durchgeführt werden, so dass tatsächlich die Subtraktion vermieden werden kann.

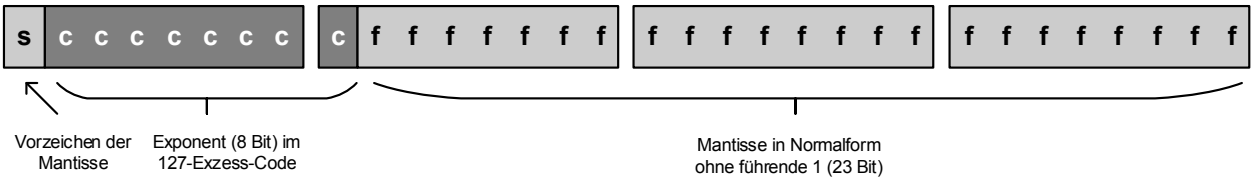
A 1.24 (M2) Schreiben Sie als 32-Bit Gleitkommazahlen:

0	64,625	-3258	0,0006912	$-21{,}40625 \cdot 10^4$
-2	75,4	$-4{,}532 \cdot 10^3$	71,46875	$-439{,}1248$

Lösung

Bei der Umwandlung einer Dezimalzahl in eine kurze binäre Gleitkommazahl geht man dementsprechend folgendermaßen vor:

1. Die Dezimalzahl wird in eine Binärzahl umgewandelt, ggf. mit Nachkommastellen.
2. Das Komma wird so weit verschoben, bis die Normalform $m = 1,f_0f_1\ldots f_{22}$ erreicht ist. Bei Verschiebung um je eine Stelle nach links wird der Exponent e der Basis 2 um eins erhöht, bei Verschiebung nach rechts um eins erniedrigt.
3. Das Vorzeichen s der Zahl (positiv: 0, negativ: 1) wird in das MSB des ersten Byte geschrieben.
4. Zum Exponenten e wird 127 addiert, das Ergebnis $c = e + 127$ wird in binäre Form mit 8 Stellen umgewandelt. Ist der Exponent positiv, so hat das führende Bit von c den Wert 1, sonst hat es den Wert 0. Die 8 Bit des Ergebnisses c werden im Anschluss an das Vorzeichenbit in die letzten 7 Bit des ersten und in das MSB des zweiten Byte eingefügt.
5. In die Bytes 2 (anschließend an das bereits für den Exponenten verwendete MSB), 3 und 4 werden schließlich die Nachkommastellen $f_0f_1\ldots f_{22}$ der Mantisse eingefügt.
6. Ist $c = 00000000$, also $e = -127$, so werden denormalisierte Gleitkommazahlen verwendet. Diese lauten $0,f_0f_1\ldots f_{22} \cdot 2^{-126}$. Details dazu werden weiter unten erläutert.



Im folgenden bezeichnet d die Zahl in Dezimaldarstellung, b die Zahl in Binärdarstellung, e den Exponenten, c den Exponenten mit Bias und g die Zahl im kurzen IEEE Gleitpunktformat.

$d = 0$
$b = 00000000\,00000000\,00000000\,00000000$
$e = -127, c = e + 127 = 0$
$g = 00000000\,00000000\,00000000\,00000000 = 00000000_{16}$
Anmerkung: Es gibt auch eine -0, diese sieht so aus:
$g = 10000000\,00000000\,00000000\,00000000 = 80000000_{16}$

$d = 64{,}625$
$b = 1000000{,}101 = 1{,}000000101 \cdot 2^6$

$e = 6_{10}, c = 6_{10} + 127_{10} = 133_{10} = 10000101_2$
$g = 01000010\,10000001\,01000000\,00000000 = 42\,81\,40\,00_{16}$

$d = -3258$
$b = 110010111010 = 1{,}10010111010 \cdot 2^{11}$
$e = 11_{10}, c = 11_{10} + 127_{10} = 138_{10} = 10001010_2$
$g = 11000101\,01001011\,10100000\,00000000 = C5\,4B\,A0\,00_{16}$

$d = 0{,}0006912$
$b = 0{,}002D4C696\ldots_{16} = 0{,}00000000001011010100110001101001 0110$
$= 1{,}01101010011000110100101 10 \cdot 2^{-11}$
$e = -11_{10}, c = -11_{10} + 127_{10} = 116_{10} = 01110100_2$
$g = 00111010\,00110101\,00110001\,10100101 = 3A\,35\,31\,A5_{16}$

$d = -21{,}40625 \cdot 10^4 = -214062{,}5$
$b = 3442E{,}8_{16} = 00110100\,01000010\,1110{,}1 = 1{,}101000100001011101 \cdot 2^{17}$
$e = 17_{10}, c = 17_{10} + 127_{10} = 144_{10} = 10010000_2$
$g = 11001000\,01010001\,00001011\,10100000 = C8\,51\,0B\,A0_{16}$

$d = -2$
$b = 10 = 1{,}0 \cdot 2^1$
$e = 1_{10}, c = 1_{10} + 127_{10} = 128_{10} = 10000000_2$
$g = 11000000\,00000000\,00000000\,00000000 = C0\,00\,00\,00_{16}$

$d = 75{,}4$
$b = 1001011{,}\overline{0110} \approx 1{,}00101101100110110011001 \cdot 2^6$
$e = 6_{10}, c = 6_{10} + 127_{10} = 133_{10} = 10000101_2$
$g = 01000010\,10010110\,11001101\,10011001 = 42\,96\,CD\,99_{16}$

$d = -4{,}532 \cdot 10^3 = -4532$
$b = 1000110110000 = 1{,}00011011 \cdot 2^{12}$
$e = 12_{10}, c = 12_{10} + 127_{10} = 139_{10} = 10001011_2$
$g = 11000101\,10001101\,10000000\,00000000 = C5\,8D\,80\,00_{16}$

$d = 71{,}46875$
$b = 1000111{,}01111 = 1{,}00011101111 \cdot 2^6$
$e = 6_{10}, c = 6_{10} + 127_{10} = 133_{10} = 10000101_2$
$g = 01000010\,10001110\,11110000\,00000000 = 42\,8E\,F0\,00_{16}$

$d = -439{,}1248$
$b \approx 110110111{,}000111111111001 = 1{,}101101110001111111111001 \cdot 2^8$
$e = 8_{10}, c = 8_{10} + 127_{10} = 135_{10} = 10000111_2$
$g = 11000011\,11011011\,10001111\,11111001 = C3\,DB\,8F\,F9_{16}$

A 1.25 (M2) Geben Sie die folgenden normalisierten und denormalisierten 32-Bit Gleitkommazahlen an: Die größte, die kleinste, die kleinste größer Null, die größte negative.

Lösung

Die größte 32-Bit Gleitpunktzahl:
Der größte nutzbare Exponent ist $e = 127_{10}$, dann ist $c = 127_{10} + 127_{10} = 254_{10} = 11111110_2$. Die größte Mantisse hat dreiundzwanzig 1en. Damit gilt:

$$\begin{aligned} g_{max} &= 01111111011111111111111111111111 = 7F7FFFFF_{16} \\ &= 1.11111111111111111111111 \cdot 2^{127} \\ &= (2-2^{-23}) \cdot 2^{127} \approx 3{,}4028234664 \cdot 10^{38}. \end{aligned}$$

Die kleinste 32-Bit Gleitpunktzahl:

$$\begin{aligned} g_{min} = -g_{max} &= 11111111011111111111111111111111 = FF7FFFFF_{16} \\ &= -1.11111111111111111111111 \cdot 2^{127} \\ &= -(2-2^{-23}) \cdot 2^{127} \approx -3{,}4028234664 \cdot 10^{38}. \end{aligned}$$

Die kleinste 32-Bit Gleitpunktzahl größer Null: Hier ist zu berücksichtigen, dass von den normalisierten Gleitpunktzahlen mit der Mantisse $1{,}f$ zu den denormalisierten Gleitpunktzahlen mit der Mantisse $0{,}f$ und dem festen Exponenten 2^{-126} übergegangen werden muss. Dies ist durch den Eintrag 0 für den Exponenten c kenntlich gemacht. Die kleinste 32-Bit positive Gleitpunktzahl hat also eine 1 in der letzten Stelle der Mantisse:

$$\begin{aligned} g_{min+} &= 00000000000000000000000000000001 = 00000001_{16} \\ &= 2^{-23} \cdot 2^{-126} = 2^{-149} \approx 1{,}4012984 \cdot 10^{-45}. \end{aligned}$$

Die größte negative 32-Bit Gleitpunktzahl:

$$\begin{aligned} g_{min-} = -g_{min+} &= 10000000000000000000000000000001 = 80000001_{16} \\ &= -2^{-23} \cdot 2^{-126} = -2^{-149} \approx -1{,}4012984 \cdot 10^{-45}. \end{aligned}$$

A 1.26 (M2) Berechnen Sie näherungsweise π unter Verwendung der ersten 11 Glieder der Leibnitz-Formel:

$$\frac{\pi}{4} = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{1}{2k-1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$$

Rechnen Sie stellenrichtig im Zehnersystem mit Gleitkommazahlen in Normalform und Mantissen mit drei Nachkommastellen, also $0{,}d_1d_2d_3 \cdot 10^e$ mit $d_1 > 0$. Ermitteln Sie die günstigste Strategie hinsichtlich der Minimierung der Abbrechfehler: (1) Abarbeitung der Terme von links nach rechts, (2) Abarbeitung der Terme von rechts nach links, (3) getrennte Addition aller positiven und negativen Terme von links nach rechts bzw. (4) von rechts nach links und anschließende Subtraktion der Zwischenergebnisse.

Lösung

Problematisch ist hier die begrenzte Stellenzahl der Mantisse bei der Gleitpunktdarstellung. Rechnet man mit Gleitpunktzahlen mit 3 Nachkommastellen im Zehnersystem so folgt für die einzelnen Terme:

1	$-1/3$	$1/5$	$-1/7$	$1/9$	$-1/11$	$1/13$	$-1/15$	$1/17$	$-1/19$	$1/21$
$0{,}100 \cdot 10^1$	$-0{,}333$	$0{,}200$	$-0{,}142$	$0{,}111$	$-0{,}909 \cdot 10^{-1}$	$0{,}769 \cdot 10^{-1}$	$-0{,}666 \cdot 10^{-1}$	$0{,}588 \cdot 10^{-1}$	$-0{,}526 \cdot 10^{-1}$	$0{,}476 \cdot 10^{-1}$

Die folgenden Rechnungen erfolgen mit Gleitpunktzahlen, die auf drei Nachkommastellen begrenzt sind. Bei der Ausführung der Addition entstehen daher Abbrechfehler.
Addition und Subtraktion von links nach rechts:
$0{,}100 \cdot 10^1 - 0{,}033 \cdot 10^1 = 0{,}067 \cdot 10^1 = 0{,}670$
$0{,}670 + 0{,}200 - 0{,}142 + 0{,}111 - 0{,}090 + 0{,}076 - 0{,}066 + 0{,}058 - 0{,}052 + 0{,}047 = 0{,}812.$

Addition und Subtraktion von rechts nach links:
$(0{,}476 - 0{,}526 + 0{,}588 - 0{,}66 + 0{,}796 - 0{,}909) \cdot 10^{-1} = -0{,}241 \cdot 10^{-1} \rightarrow -0{,}024$
$-0{,}024 + 0{,}111 - 0{,}142 + 0{,}200 - 0{,}333 = -0{,}188 \rightarrow -0{,}018 \cdot 10^1$
$-0{,}018 \cdot 10^1 + -0{,}100 \cdot 10^1 = 0{,}082 \cdot 10^1 = 0{,}820.$

Getrennte Addition aller positiven und aller negativen Terme von links nach rechts und anschließende Subtraktion:
Positive Terme:
$(0{,}100 + 0{,}020 + 0{,}011 + 0{,}007 + 0{,}005 + 0{,}004) \cdot 10^1 = 0{,}147 \cdot 10^1.$
Negative Terme:
$0{,}333 + 0{,}142 + 0{,}090 + 0{,}066 + 0{,}052 = 0{,}683.$
Ergebnis:
$0{,}147 \cdot 10^1 - 0{,}683 = 0{,}147 \cdot 10^1 - 0{,}068 \cdot 10^1 = 0{,}079 \cdot 10^1 = 0{,}790.$

Getrennte Addition aller positiven und aller negativen Terme von rechts nach links und anschließende Subtraktion:
Positive Terme:
$(0{,}476 + 0{,}588) \cdot 10^{-1} = 1{,}064 \cdot 10^{-1} = 0{,}106$
$0{,}106 + 0{,}076 + 0{,}111 + 0{,}200 = 0{,}493 \rightarrow 0{,}049 \cdot 10^1$
$0{,}049 \cdot 10^1 + 0{,}100 \cdot 10^1 = 0{,}149 \cdot 10^1.$
Negative Terme:
$(0{,}526 + 0{,}666) \cdot 10^{-1} = 1{,}192 \cdot 10^{-1} \rightarrow 0{,}119$
$0{,}119 + 0{,}090 + 0{,}142 + 0{,}333 = 0{,}684.$
Ergebnis:
$0{,}149 \cdot 10^1 - 0{,}684 = 0{,}149 \cdot 10^1 - 0{,}068 \cdot 10^1 = 0{,}081 \cdot 10^1 = 0{,}810.$

Rechnet man auf 5 Stellen genau, so erhält man 0,80807. Vergleicht man dies mit den obigen Ergebnissen, so wird deutlich, dass Verfahren (4), also die getrennte Addition aller positiven und negativen Terme von rechts nach links und anschließende Subtraktion der Zwischenergebnisse das genaueste Ergebnis liefert.

A 1.27 (M2, P1) Zur näherungsweisen Berechnung von Wurzeln $y = \sqrt{x}$ mit $x \geq 0$ ist die Newtonsche Iterationsformel $y_{k+1} = (y_k + \frac{x}{y_k})/2$ gut geeignet. Beginnend mit einem Startwert $y_0 \neq 0$ berechnet man Näherungswerte, bis der Fehler $\varepsilon = |y_{k+1} - y_k| < s$ mit gegebenem s wird.

a) Entwickeln Sie eine einfache Strategie zur Ermittlung eines Startwertes y_0, für den die zu erwartende Anzahl der Iterationen möglichst klein ist. Rechnen Sie dabei im Zehnersystem mit Gleitkommazahlen in Normalform, also $0{,}d_1d_2d_3 \cdots \cdot 10^e$ mit $d_1 > 0$.

b) Vergleichen Sie Ihre Strategie mit den Strategien $y_0 = 1$ und $y_0 = x$, indem Sie für einige repräsentative Beispiele die Anzahl der erforderlichen Iterationen zählen. Schreiben Sie dazu am besten ein Programm in einer beliebigen Programmiersprache Ihrer Wahl.

Lösung

a) Ist x als Gleitpunktzahl im Zehnersystem gegeben, so ist $10^{e/2}$ eine gute Näherung für den Startwert, da dies ja die Wurzel aus 10^e ist, so dass sich die Newtonsche Iteration nur noch auf die Mantisse bezieht, die

immer zwischen 0 und 1 liegt. Am einfachsten lässt sich dies mithilfe eines Programms nachvollziehen. Im Beispiel wurde $x = 123456$ verwendet, der Exponent ist also $e = 6$, entsprechend dem Startwert $y_0 = 1000$. Dafür sind nur 6 Iterationen erforderlich. Für den Startwert 1 sind dagegen 13 Iterationen erforderlich.

Iteration	Näherungswert
1	561,728027
2	390,753479
3	353,348450
4	351,368652
5	351,363068
6	351,363068

b) Der Vergleich der Anzahl der Iterationen für den Startwert $y_0 = 1$ mit Anzahl der Iterationen unter Verwendung des optimierten Startwert $y_0 = 10^{e/2}$ ist oben bereits erledigt. Der Startwert $y_0 = x$ ist mit dem Startwert $y_0 = 1$ identisch. Man erkennt dies unmittelbar durch Einsetzen in die Newtonsche Formel $y_1 = \frac{1}{2}(y_0 + \frac{x}{y_0}) = \frac{y_0+1}{2}$.

A 1.28 (L1) Berechnen Sie $(a \wedge \neg b) \vee c$ für $a = 10111011$, $b = 01101010$, $c = 10101011$. Zeigen Sie an diesem Beispiel, dass $(a \wedge \neg b) \vee c = (a \vee c) \wedge (\neg b \vee c)$ gilt.

Lösung

Berechnung von $(a \wedge \neg b) \vee c$:

a:	10111011
b:	01101010
$\neg b$:	10010101
$a \wedge \neg b$:	10010001
c:	10101011
$(a \wedge \neg b) \vee c$:	10111011

Berechnung von $a \vee c$:

a:	10111011
c:	10101011
$a \vee c$:	10111011

Berechnung von $\neg b \vee c$:

b:	01101010
$\neg b$:	10010101
c:	10101011
$\neg b \vee c$:	10111111

Berechnung von $(a \vee c) \wedge (\neg b \vee c)$:

$a \vee c$:	10111011
$\neg b \vee c$:	10111111
$(a \vee c) \wedge (\neg b \vee c)$:	10111011

Der Vergleich der beiden Ergebnisse zeigt, dass tatsächlich $(a \wedge \neg b) \vee c = (a \vee c) \wedge (\neg b \vee c)$ gilt.

A 1.29 (L1) Bestimmen Sie für 10110_{bin} und 10011011_{bin} jeweils das Ergebnis einer logischen und einer arithmetischen Verschiebung zunächst um eine und dann um zwei Stellen nach rechts und nach links. Gehen Sie dabei von einem 8-Bit-Register mit Übertragsbit aus.

Lösung

Bei der arithmetischen Verschiebung nach rechts bleibt das MSB erhalten, damit sich im Falle arithmetischer Operationen das Vorzeichen nicht ändert. Bei der logischen Verschiebung spielt das MSB keine Sonderrolle. Die logische und arithmetische Verschiebung nach links unterscheiden sich nicht.

Ausgangssituation:

Register	Übertrag	Register	Übertrag
00010110	0	10011011	0

Logische und arithmetische Verschiebung um eine Stelle nach links:

Register	Übertrag	Register	Übertrag
00101100	0	00110110	1

Logische und arithmetische Verschiebung um zwei Stellen nach links:

Register	Übertrag	Register	Übertrag
01011000	0	01101100	0

Logische Verschiebung um eine Stelle nach rechts:

Register	Übertrag	Register	Übertrag
00001011	0	01001101	1

Logische Verschiebung um zwei Stellen nach rechts:

Register	Übertrag	Register	Übertrag
00000101	1	00100110	1

Arithmetische Verschiebung um eine Stelle nach rechts:

Register	Übertrag	Register	Übertrag
00001011	0	11001101	1

Arithmetische Verschiebung um zwei Stellen nach rechts:

Register	Übertrag	Register	Übertrag
00000101	1	11100110	1

2 Lösungen zu Kapitel 2 – Nachricht und Information

Übungsaufgaben zu Kapitel 2.1

A 2.1 (T1) Welche Mengen sind Alphabete: die geraden Zahlen, die reellen Zahlen, die Menge der Verkehrszeichen, die Primzahlen, die Noten der musikalischen Notenschrift?

Lösung

Die geraden Zahlen bilden ein Alphabet, da es sich um eine abzählbare Menge mit Ordnungsrelation handelt.
Die reellen Zahlen bilden kein Alphabet. Es existiert zwar eine Ordnungsrelation, doch ist die Menge der reellen Zahlen nicht abzählbar.
Die Verkehrszeichen bilden kein Alphabet. Es handelt sich zwar um eine abzählbare Menge, doch gibt es keine allgemeingültige Ordnungsrelation.
Die Primzahlen bilden ein Alphabet, da es sich um eine abzählbare Menge mit Ordnungsrelation handelt.
Die Noten der musikalischen Notenschrift sind ein Grenzfall. Die Menge der musikalischen Noten ist abzählbar und mit der Tonhöhe ist auch eine Ordnungsrelation definiert, insofern kann man von einem Alphabet sprechen. Eine Komplikation wäre die Erweiterung um die Notenlängen, wobei aber die Länge der Noten ebenfalls als Ordnungsrelation verstanden werden kann. Nähme man noch weitere Symbole der Notenschrift hinzu (etwa Notenschlüssel, Taktangaben sowie Erhöhungs-, Erniedrigungs- und Auflösungszeichen), so gäbe es keine erkennbare Ordnungsrelation mehr, so dass man nicht mehr von einem Alphabet sprechen könnte.

A 2.2 (T1) Was ist der Unterschied zwischen Nachrichten- und Informationsverarbeitung?

Lösung

Bei Nachrichten handelt es sich um im mathematischen Sinne konkrete und eindeutig definierte Objekte. Damit ist auch deren Verarbeitung (insbesondere durch einen Computer) eindeutig beschreibbar. Informationen gehen durch Interpretationen, die nicht eindeutig sein müssen und subjektive Züge tragen können, aus Nachrichten hervor. Der Begriff „Informationsverarbeitung“ ist daher nicht klar definiert. Meist ist Nachrichtenverarbeitung gemeint, wenn man von Informationsverarbeitung spricht.

A 2.3 (T1) Lässt sich die Zahl π als Nachricht übermitteln?

Lösung

Die Zahl π ist eine reelle Zahl, und zwar eine transzendente Zahl mit unendlich vielen Stellen, deren Folge kein periodisches Muster bildet. Da von einer Nachricht gefordert werden muss, dass sie aus einer abzählbaren Folge von Einzelzeichen bestehen muss, kann π genau genommen nicht als Nachricht übermittelt werden. Man kann jedoch einen Näherungswert von π mit beliebig vielen Stellen übermitteln, oder auch eine Rechenvorschrift, die dem Empfänger prinzipiell die Berechnung beliebig vieler Stellen von erlaubt.

Übungsaufgaben zu Kapitel 2.2

A 2.4 (T0) Beschreiben Sie kurz die wesentlichen Details des genetischen Codes.

Lösung

Das Alphabet des genetischen Codes lautet {A, C, G, T}, wobei die Zeichen für die Nukleotide Adenin, Cytosin, Guanin und Thymin stehen. In den DNS-Molekülen codieren jeweils drei aufeinander folgende Zeichen eine Nukleinsäure. Da es $4^3 = 64$ verschiedene Kombinationsmöglichkeiten gibt, aber nur 20 Nukleinsäuren zu codieren sind, stehen in der Regel mehrere, oft bis zu 6 verschiedene Codewörter für dieselbe Nukleinsäure. Vier spezielle Codewörter (ATG, TAA, TGA und TGG) steuern den Beginn und den Abbruch der Synthese von Proteinen aus den Nukleinsäuren. Aus den Nukleinsäuren ist der genetische Code aufgebaut.

A 2.5 (T1) Vergleichen Sie biologische Gehirne mit digitalen Computern hinsichtlich Verarbeitungsgeschwindigkeit, Parallelität, Fehlertoleranz und Speicherprinzip.

Lösung

Biologische Gehirne weisen im Vergleich mit digitalen Computern eine niedrige Verarbeitungsgeschwindigkeit der Einzelkomponenten (Neuronen), eine hohe Parallelität und eine hohe Fehlertoleranz auf, d. h. Erhaltung der Funktionalität auch bei Ausfall einzelner Komponenten. Die hohe Parallelität biologischer Gehirne gleicht dabei die niedrige Arbeitsgeschwindigkeit der Einzelkomponenten teilweise wieder aus. Biologische Gehirne verwenden das assoziative Speicherprinzip, bei dem Inhalte gestreut gespeichert und durch Vergleich mit Mustern wieder gefunden werden. In üblichen Digitalcomputern erfolgt die Speicherung durch eindeutige Adressierung von Speicherzellen. Aus den unterschiedlichen Operationsprinzipien ergibt sich, dass biologische Gehirne auf schnelle Mustererkennung und Computer auf Exaktheit und Geschwindigkeit bei numerischen Berechnungen optimiert sind.

A 2.6 (M1) Der einer Tonhöheempfindung R_1 entsprechende physikalische Reiz S_1 betrage 1% des Maximalreizes $S_{\max}$. Das Verhältnis von Maximalreiz zu Reizschwelle hat den Wert $S_{\max}/S_0 = 10^3$. Um welchen Faktor muss ein physikalischer Reiz S_2 größer sein als S_1, damit sich die zugehörige Reizempfindung von R_1 auf R_2 verdoppelt?

Lösung

Nach dem Fechnerschen Gesetz gilt $R = c \cdot \log\left(\frac{S}{S_0}\right)$. Mit $S_{\max}/S_0 = 10^3$ rechnet man:

$$
\begin{aligned}
R_2 = c \cdot \log\left(\frac{S_2}{S_0}\right) = 2R_1 = 2c \cdot \log\left(\frac{S_1}{S_0}\right) &= c \cdot \log\left(\frac{S_1}{S_0}\right) + c \cdot \log\left(\frac{S_1}{S_0}\right) \\
\Rightarrow c \cdot \log\left(\frac{S_2}{S_0}\right) &= c \cdot \log\left(\frac{S_1}{S_0}\right) + c \cdot \log\left(\frac{S_1}{S_0}\right) \\
\Rightarrow \log\left(\frac{S_2}{S_0}\right) &= \log\left(\frac{S_1}{S_0}\right) + \log\left(\frac{S_1}{S_0}\right) \\
\Rightarrow \log\left(\frac{S_2}{S_0}\right) - \log\left(\frac{S_1}{S_0}\right) &= \log\left(\frac{S_1}{S_0}\right) \\
\Rightarrow \log\left(\frac{S_2}{S_1}\right) = \log\left(\frac{S_1}{S_0}\right) \quad &\Rightarrow \frac{S_2}{S_1} = \frac{S_1}{S_0} = 0{,}01 \cdot 10^3 = 10
\end{aligned}
$$

Der physikalische Reiz S_1 muss also um den Faktor 10 auf den physikalischen Reiz S_2 erhöht werden, damit sich die Empfindung der Lautstärke verdoppelt.

A 2.7 (M2) Die zu einem empfundenen Helligkeitsreiz R gehörige Pulsfrequenz f ist proportional zu R, es gilt also $f \sim R$. Außerdem gilt nach dem Fechnerschen Gesetz $R \sim \log(S/S_0)$. Für das Helligkeitsempfinden

kennt man die zum physikalischen Maximalreiz S_{max} gehörende maximale Pulsfrequenz $f_{max} = 250\,\text{Hz}$ sowie das Verhältnis von Maximalreiz zu Reizschwelle $S_{max}/S_0 = 10^{10}$ aus experimentellen Untersuchungen. Welche Pulsfrequenz gehört zu einem physikalischen Reiz $S_1 = S_{max}/10$?

Lösung

Aus $f \sim R$ und $R \sim \log(S/S_0)$ folgt $f = c\log(S/S_0)$ mit einer Proportionalitätskonstanten c.
Aus $f_{max} = c\log(S_{max}/S_0)$ und $250 = c \cdot \log 10^{10}$ folgt $250 = c \cdot 10$, also $c = 25$.
Also: $f_1 = c\log(S_1/S_0) = 25 \cdot \log(S_{max}/S_0/10) = 25(\log(S_{max}/S_0) - \log 10) = 25 \cdot (10-1) = 225\,\text{Hz}$.

A 2.8 (M3) Nach dem Weberschen Gesetz gilt für eine von biologischen Rezeptoren gerade noch auflösbare Differenz ΔS eines physikalischen Reizes S die Beziehung $\Delta S = k \cdot S$. Berechnen Sie daraus mit gegebener Proportionalkonstante k und gegebenem S_{max}/S_0 (Maximalreiz zu Reizschwelle) die Anzahl der aufgelösten Reizstufen für folgende Reize:

a) Helligkeit: $S_{max}/S_0 = 10^{10}$, $k = 0{,}02$
b) Lautstärke: $S_{max}/S_0 = 10^{12}$, $k = 0{,}09$
c) Tonhöhe: $S_{max}/S_0 = 10^3$, $k = 0{,}003$

Lösung

Aus $\Delta S = k \cdot S$ folgt:

$$\begin{aligned} \Delta S_1 = S_1 - S_0 = k \cdot S_0 \quad &\Rightarrow \quad S_1 = S_0(1+k) \\ \Delta S_2 = S_2 - S_1 = k \cdot S_1 \quad &\Rightarrow \quad S_2 = S_1(1+k) = S_0(1+k)^2 \\ &\vdots \\ &\quad S_n = S_0(1+k)^n \end{aligned}$$

Also:

$$\frac{S_n}{S_0} = (1+k)^n \Rightarrow n = \frac{\log(S_n/S_0)}{\log(1+k)}$$

Einsetzen der gegebenen Zahlen liefert:

a) Helligkeit:	$S_{max}/S_0 = 10^{10}$	$k = 0{,}02$	$n = \frac{10}{\log 1{,}02} = 1163$
b) Lautstärke:	$S_{max}/S_0 = 10^{12}$	$k = 0{,}09$	$n = \frac{12}{\log 1{,}09} = 320$
c) Tonhöhe:	$S_{max}/S_0 = 10^3$	$k = 0{,}003$	$n = \frac{3}{\log 1{,}003} = 2306$

A 2.9 (T1) Nennen Sie Parallelen zwischen biologischer und elektronischer Datenverarbeitung.

Lösung

Nachrichtenübertragung durch elektrische Signale; Verwendung der besonders störsicheren Pulscode-Modulation; Mustererkennung in Sprach- und Bildverarbeitung; Parallelverarbeitung biologischer Gehirne und technisch realisierter neuronaler Netze; assoziative Speicherung von Daten; Verwendung des digitalen genetischen Codes bzw. des digitalen Binär-Codes; Nachbildung der Evolution in genetischen Algorithmen.

Übungsaufgaben zu Kapitel 2.3

A 2.10 (M1) Ein elektronisches Thermometer soll einen Temperaturbereich von $-40°C$ bis $+50°C$ erfassen und eine zur Temperatur proportionale Spannung zwischen 0 und 5 V ausgeben. Die Aufnahme dieser Temperaturdaten soll mithilfe einer in einen PC einzubauenden Digitalisierungskarte erfolgen. Zur Auswahl einer geeigneten Karte sind einige Fragen zu klären.

a) Wie viele Bit sind mindestens für die Digitalisierung nötig, wenn die Auflösung ca. 0,1°C betragen soll?

b) Wie groß ist das durch das Quantisierungsrauschen bedingte Signal-Rausch-Verhältnis im mittleren Temperaturbereich?

c) Die Elektronik des Temperatursensors überlagert dem Temperatursignal ein elektronisches weißes Rauschen von $3 \cdot 10^{-4}$ des maximalen Ausgangssignals. Vergleichen Sie dieses elektronische Rauschen mit dem Quantisierungsrauschen. Ist das Ergebnis mit dem in Teilaufgabe a) gefundenen Ergebnis zu vereinbaren?

Lösung

a) Eine Auflösung von 0,1°C bei einem Wertebereich von $-40°C$ bis $+50°C$ bedeutet, dass 900 Teilschritte erforderlich sind. Dabei entsprechen 0 V einer Temperatur von $-40°C$ und 5 V einer Temperatur von $+50°C$. Temperaturschritte von 0,1°C entsprechen also einer Spannungsdifferenz von $5000/900 \approx 5{,}55$ mV. Eine Digitalisierung mit 9 Bit ergäbe $2^9 = 512$ Teilschritte, eine Digitalisierung mit 10 Bit ergäbe $2^{10} = 1024$ Teilschritte. Man muss also eine Digitalisierung mit 10 Bit wählen. Ein Teilschritt entspricht dann $90°C/1024 \approx 0{,}088°C$ bzw. $5000\,\text{mV}/1024 \approx 4{,}88$ mV.

b) Das Quantisierungsrauschen beträgt $r \approx 4{,}88\,\text{mV}/2 \approx 2{,}44$ mV. In der Mitte des Temperaturbereichs, also bei 5°C entsprechend 2500 mV, ist dann das Signal-Rausch-Verhältnis:
$\text{SNR} \approx 20 \cdot \log(2500/2{,}44) \approx 60{,}2$ dB.

c) Das elektronische weiße Rauschen des analogen Temperatursignals beträgt nach Angabe $3 \cdot 10^{-4} \cdot 5\,\text{V} = 1{,}5$ mV. Das in Teilaufgabe b) berechnete Quantisierungsrauschen ist mit 2,44 mV also deutlich höher als das elektronische Rauschen, so dass die Digitalisierung mit 10 Bit sinnvoll ist. Eine Digitalisierung mit 11 Bit, entsprechend 2048 Schritten, wäre vorzuziehen, da dann das Digitalisierungsrauschen mit 1,22 mV mit dem elektronischen Rauschen vergleichbar wäre. ADCs mit 11 Bit sind aber technisch unüblich. Eine Digitalisierung mit 12 Bit, entsprechend 2048 Schritten, wäre nicht mehr sinnvoll, da nun das elektronische Rauschen mit 1,5 mV erheblich höher wäre als das Quantisierungsrauschen mit 0,61 mV.

A 2.11 (M1) Ein Videosignal mit einer Grenzfrequenz von 7,5 MHz soll durch einen 10-Bit ADC digitalisiert werden. Welchen zeitlichen Abstand sollten die äquidistanten Abtastschritte mindestens einhalten? Geben Sie außerdem das SNR an.

Lösung

Die in dem Videosignal enthaltene Information wird exakt wiedergegeben, wenn die Abtastrate (Sampling Rate) gemäß der Nyquist-Bedingung mindestens doppelt so groß gewählt wird wie die Grenzfrequenz ν_G. Für den zeitlichen Abstand t_s der Abtastschritte folgt damit:

$$t_s \leq \frac{1}{2\nu_G} = \frac{10^{-6}}{2 \cdot 7{,}5} \approx 0{,}066[\mu s]$$

Bei Verwendung eines 10-Bit ADC wird das Videosignal bezogen auf seinen Maximalpegel f_{max} in $2^{10} = 1024$ äquidistante Schritte unterteilt. Ein Quantisierungsschritt beträgt damit $f_{max}/1024$ und der dabei maximal

mögliche Fehler, das Quantisierungsrauschen, gerade die Hälfte dieses Werts, also $r = f_{max}/2048$ Oft wird als Maß für die Güte einer Quantisierung auch der relative Quantisierungsfehler, also der Quotient f_{ave}/r angegeben, wobei $f_{ave} \approx f_{max}/2$ der mittlere Pegel ist. Da r meist viel kleiner ist als f_{ave}, kann f_{ave}/r sehr groß werden. Aus diesem Grund, und weil dies auch dem physiologischen Empfinden besser entspricht, verwendet man den als Signal-Rausch-Abstand (Signal-to-Noise Ratio, SNR) bezeichneten mit 20 multiplizierten Logarithmus dieses Quotienten mit der Maßeinheit dB (Dezibel):

$$\text{SNR} = 20 \cdot \log(f_{ave}/r) \quad \text{[dB]} \quad \text{Hier: SNR} = 20 \cdot \log(2048/2) \approx 60{,}20 \quad \text{[dB]} \quad .$$

A 2.12 (T1) Grenzen Sie die folgenden Begriffe gegeneinander ab: Digitalisierung, Diskretisierung, Abtastung, Sampling und Quantisierung.

Lösung

Unter Digitalisierung oder Diskretisierung versteht man die Transformation von Nachrichten aus der für gewöhnlich kontinuierlichen Darstellung in eine diskrete, elektronische Darstellung mit endlich vielen Stufen, die für die Verarbeitung in Digitalrechnern erforderlich ist. Als Abtastung oder Sampling bezeichnet man die Abtastung der Werte einer Funktion an bestimmten vorgegebenen Stellen, also die Diskretisierung des Definitionsbereichs der Funktion. Der Übergang von einer kontinuierlichen zu einer digitalen Nachricht erfordert nach der Abtastung noch einen zweiten Diskretisierungsschritt, die Quantisierung. Dazu wird der Wertebereich der zu diskretisierenden Funktion in eine Menge von Zahlen abgebildet, die das Vielfache einer bestimmten Zahl sind, des sogenannten Quantenschritts.

Übungsaufgaben zu Kapitel 2.4

A 2.13 (M1) Sie haben es in die Vorrunde des Millionenspiels geschafft. Die Einstiegsfrage lautet: Ordnen Sie die folgenden Mittelmeerinseln in der Reihenfolge von Ost nach West:
A: Kreta, B: Korsika, C: Sizilien, D: Zypern
Da Sie keine Ahnung haben, raten Sie. Mit welcher Wahrscheinlichkeit liegen Sie richtig?

Lösung

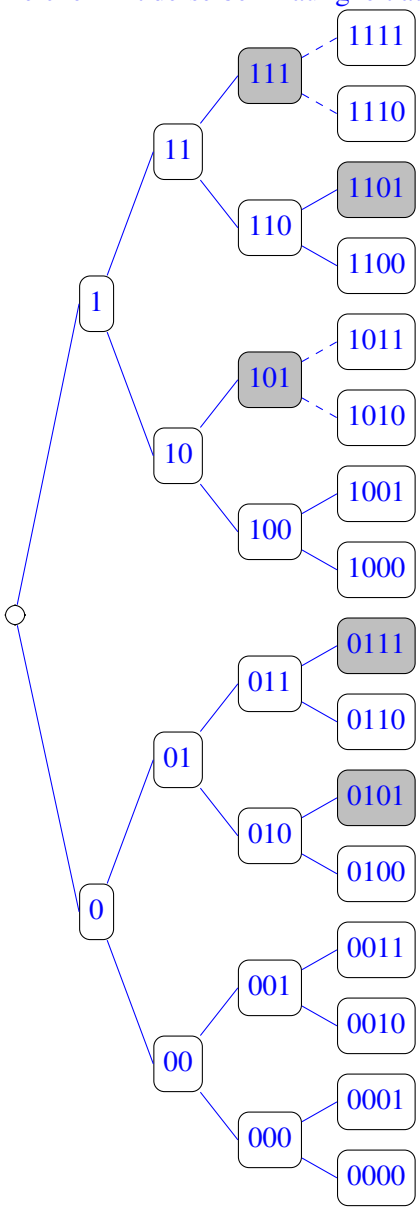
Nur eine der möglichen Anordnungen der 4 Elemente {A, B, C, D} ist richtig. Es handelt sich um Permutationen, also gibt es $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$ Möglichkeiten. Die Wahrscheinlichkeit dafür, das richtige Ergebnis zu raten beträgt also $1/24 \approx 0{,}042$ entsprechend ca. 4,2%. Dieses Ergebnis findet man auch durch systematisches Aufschreiben aller 24 Möglichkeiten:
ABCD ABDC ACBD ACDB ADBC ADCB
BACD BADC BCAD BCDA BDAC BDCA
CABD CADB CBAD CBDA CDAB CDBA
DABC DACB DBAD DBDA DCAB DCBA
Die korrekte Antwort lautet CBAD.

A 2.14 (L1) Mithilfe eines Zufallsgenerators werde eine Folge aus den Zeichen 0 und 1 erzeugt. Der Vorgang soll beendet werden, sobald entweder die Teilfolge 111 oder die Teilfolge 101 auftritt. Welcher Ausgang ist wahrscheinlicher?

Lösung

Es wird angenommen, dass der Zufallsgenerator die beiden Zeichen 0 und 1 mit jeweils 50% Wahrscheinlichkeit erzeugt. Wie die unten stehende Grafik zeigt, ist der Ausgang des Spieles mit der Folge 101

wahrscheinlicher, als mit der Folge 111. Wichtig ist allerdings, dass das Spiel nach Auftreten einer dieser Folgen beendet werden muss.
In einer unendlich langen Zufallsfolge aus 0en und 1en treten selbstverständlich alle Kombinationen aus drei Zeichen mit derselben Häufigkeit auf.



A 2.15 (M2) Wie wahrscheinlich ist es, im Lotto 6 aus 49, 5 bzw. 3 Richtige zu tippen?

Lösung

Beim Lottospiel werden 6 Elemente ohne Wiederholungen aus einer Menge von 49 Elementen ausgewählt, wobei die Reihenfolge der Auswahl keine Rolle spielt. Es handelt sich also um Kombinationen ohne Wiederholungen. Die Anzahl der Möglichkeiten ist demnach

$$\binom{49}{6} = 13983816$$

Da genau 6 Zahlen gezogen werden, gibt es auch nur ein Mal 6 Richtige. Daher folgt nach der Abzählregel „Anzahl der günstigen / Anzahl der möglichen Fälle“ für die Wahrscheinlichkeit, 6 Richtige zu tippen:

$p_6 = 1/13983816 \approx 7{,}151 \cdot 10^{-8}$.
Etwas schwieriger ist es, allgemein die Wahrscheinlichkeit für das Tippen von $m < k$ richtigen Zahlen aus $k = 6$ Gewinnzahlen zu berechnen, die aus einer Menge von $n = 49$ Zahlen gezogen wurden. Dazu ist zunächst die Anzahl der günstigen Fälle zu berechnen. Diese ergibt sich durch die Anzahl der Möglichkeiten, die m Gewinnzahlen aus den 6 gezogenen Zahlen auszuwählen, multipliziert mit der Anzahl, wie die $k-m$ getippten Nicht-Gewinnzahlen auf die verbleibenden $n-k$ nicht gezogenen Zahlen verteilt werden können. Insgesamt ergibt sich dann für $m = 5$ und $m = 3$:

$$p_5 = \frac{\binom{k}{m}\binom{n-k}{k-m}}{\binom{n}{k}} = \frac{\binom{6}{5}\binom{43}{1}}{\binom{49}{6}} = \frac{15 \cdot 903}{13983816} \approx \frac{1}{1032} \approx 0{,}00097 \quad ,$$

$$p_3 = \frac{\binom{6}{3}\binom{43}{3}}{\binom{49}{6}} = \frac{20 \cdot 12341}{13983816} \approx \frac{1}{56{,}656} \approx 0{,}01765 \quad .$$

Diese auch für viele andere Anwendungen wichtige Funktion trägt den Namen *hypergeometrische Verteilung*.

A 2.16 (M2) Ein Geschäftsmann reist von Tokyo über Singapur und Bahrein nach München. In jedem der vier Flughäfen ist die Wahrscheinlichkeit dafür, dass sein Koffer verloren geht p. Der Geschäftsmann wartet in München vergeblich auf seinen Koffer. Berechnen Sie für jeden der Flughäfen die Wahrscheinlichkeit, dass der Koffer gerade dort verloren ging.

Lösung

Die Wahrscheinlichkeit dafür, dass der Koffer in Tokyo verloren gegangen ist, beträgt $p_T = p$.
Für Singapur: $p_s = (1-p)p$
Für Bahrein: $p_B = (1-p)(1-p)p$
Für München: $p_M = (1-p)(1-p)(1-p)p$
Dies ergibt sich durch Anwendung der Multiplikationsregel für unabhängige Ereignisse: Wenn der Koffer genau in München verloren geht, dann darf er nicht in Tokyo, nicht in Singapur und nicht in Bahrein verloren gegangen sein; sonst wäre er ja in München gar nicht erst angekommen und könnte dort auch nicht verloren gehen.

A 2.17 (L3) Die Journalistin Marilyn vos Savant hat folgendes Fernseh-Quiz in Amerika populär gemacht: Einem Kandidaten werden 3 Türen gezeigt und mitgeteilt, dass sich hinter zwei Türen je eine Ziege befindet und hinter einer Tür ein Auto. Errät der Kandidat die Tür, hinter der das Auto steht, so erhält er es als Gewinn. Der Kandidat zeigt nun ohne diese zu öffnen auf eine Tür. Der Spielleiter öffnet nun von den verbleibenden beiden Türen diejenige, hinter der sich eine Ziege befindet. Der Kandidat darf nun nochmals wählen. Mit welcher Wahrscheinlichkeit gewinnt der Kandidat mit den drei folgenden Strategien das Auto?

a) Er bleibt bei der getroffenen Wahl.

b) Er wählt nun die andere noch geschlossene Tür.

c) Er wirft zur Wahl der beiden verfügbaren Türen eine Münze.

Lösung

a) Bleibt der Kandidat bei seiner Wahl, so spielt die Tatsache, dass eine Tür geöffnet wurde, keine Rolle. Der Kandidat hätte ebenso gut nach seiner anfänglichen Wahl nach Hause gehen können, ohne abzuwarten, welche Tür der Spielleiter nun öffnet. Die Wahrscheinlichkeit dafür, das Auto zu erhalten ist also nach der Abzählregel $= 1/3$.

b) Hier hat der Kandidat die Möglichkeit, das zusätzliche Wissen nach Öffnen einer Tür mit einzubeziehen. Es ergeben sich folgende drei gleich wahrscheinliche Möglichkeiten:

TÜR 1	TÜR 2	TÜR 3
Auto erste Wahl	*Ziege* neue Wahl geöffnet	*Ziege* geöffnet neue Wahl
Auto neue Wahl	*Ziege* erste Wahl	*Ziege* geöffnet
Auto neue Wahl	*Ziege* geöffnet	*Ziege* erste Wahl

Ändert der Kandidat also seine Wahl (neue Wahl), trifft er einmal auf „Ziege“ und zweimal auf „Auto“. Die Wahrscheinlichkeit dafür, das Auto zu gewinnen, ist bei dieser Strategie also $p = 2/3$. Die Überlegungen aus Teilaufgabe a) werden bestätigt: Bleibt der Kandidat bei seiner Wahl (erste Wahl), so trifft er zweimal auf „Ziege“ und einmal auf „Auto“. Es folgt $p = 1/3$.

c) Nach dem Öffnen einer Tür, die jetzt unbeachtet bleibt, entsteht durch das Würfeln ein neues Zufallsexperiment, bei dem nur noch zwischen zwei Türen gewählt wird. Da bekannt ist, dass hinter einer Türe das Auto und hinter der anderen Türe eine Ziege steht, ist die Wahrscheinlichkeit dafür, das Auto zu gewinnen $p = 1/2$.

Eine gängige Interpretation des Wahrscheinlichkeitsbegriffs bezieht sich auf die objektive Prognose künftiger Ereignisse. Häufig, so auch hier, drückt man mit Wahrscheinlichkeiten aber auch einen subjektiven, mehr oder weniger lückenhaften Kenntnisstand aus. Eine Änderung des Kenntnisstandes bewirkt somit auch eine Änderung von Wahrscheinlichkeiten. Die häufig zu beobachtende Verwirrung im Zusammenhang mit dem Ziegenproblem beruht gerade auf der Verwechslung der objektiven und subjektiven Wahrscheinlichkeit.

A 2.18 (L3) Schafkopfen ist ein Spiel mit 32 Karten für 4 Mitspieler, die jeweils 8 Karten erhalten. Wahrscheinlichkeiten und Kombinatorik sind dabei von großer Bedeutung.

a) Wie hoch ist die Wahrscheinlichkeit, ein „Sie“ (d. h. alle 4 Unter und alle 4 Ober) auf die Hand zu bekommen?

b) Wie hoch ist die Wahrscheinlichkeit, einen „Wenz mit 4“ (d. h. alle 4 Unter und sonst beliebige Karten) auf die Hand zu bekommen?

Lösung

a) Es gibt 32 Karten, davon erhält jeder der vier Mitspieler 8 Karten. Die Wahrscheinlichkeit dafür, dass einer der Spieler alle vier Ober und alle vier Unter erhält ist dann:

$$p = \underbrace{\left(\frac{8}{32}\cdot\frac{7}{31}\cdot\frac{6}{30}\cdot\frac{5}{29}\right)}_{\text{4 Ober}}\cdot\underbrace{\left(\frac{4}{28}\cdot\frac{3}{27}\cdot\frac{2}{26}\cdot\frac{1}{25}\right)}_{\text{4 Unter}} = 9{,}507\cdot 10^{-8} \quad .$$

Andere Lösungsmöglichkeit: Die Anzahl der Möglichkeiten, 8 Elemente aus einer Anzahl von 32 ohne Beachtung der Reihenfolge auszuwählen, berechnet man als Kombinationen $C(32,8)$ ohne Wiederholungen. Für die gesuchte Wahrscheinlichkeit folgt dann:

$$p = \frac{1}{C(32,8)} = \frac{1}{\binom{32}{8}} = \frac{8!(32-8)!}{32!} = 9{,}507\cdot 10^{-8} \quad .$$

b) Es gibt $C(8,4) = \binom{8}{4} = 70$ Möglichkeiten (Kombinationen ohne Wiederholungen), 4 Unter in 8 Karten anzuordnen. Für die Kombination UUUU???? Mit U = Unter und ? = Nicht-Unter folgt die Wahrscheinlichkeit:

$$P(\text{UUUU????}) = \frac{4}{32}\cdot\frac{3}{31}\cdot\frac{2}{30}\cdot\frac{1}{29}\cdot\frac{28}{28}\cdot\frac{27}{27}\cdot\frac{26}{26}\cdot\frac{25}{25} = \frac{4}{32}\cdot\frac{3}{31}\cdot\frac{2}{30}\cdot\frac{1}{29}$$

Für jede andere der 70 Möglichkeiten ergibt sich derselbe Wert. Man kann dies anhand eines Beispiels nachvollziehen:

$$P(\text{?U??UU?U}) = \frac{28}{32}\cdot\frac{4}{31}\cdot\frac{27}{30}\cdot\frac{26}{29}\cdot\frac{3}{28}\cdot\frac{2}{27}\cdot\frac{25}{26}\cdot\frac{1}{25} = \frac{4}{32}\cdot\frac{3}{31}\cdot\frac{2}{30}\cdot\frac{1}{29}$$

Die Gesamtwahrscheinlichkeit ist nun die Summe dieser 70 identischen Einzelwahrscheinlichkeiten:

$$P(\text{Wenz mit 4}) = 70\cdot\frac{4}{32}\cdot\frac{3}{31}\cdot\frac{2}{30}\cdot\frac{1}{29} = \frac{7}{29\cdot 31\cdot 4} = 2{,}947\cdot 10^{-3} \quad .$$

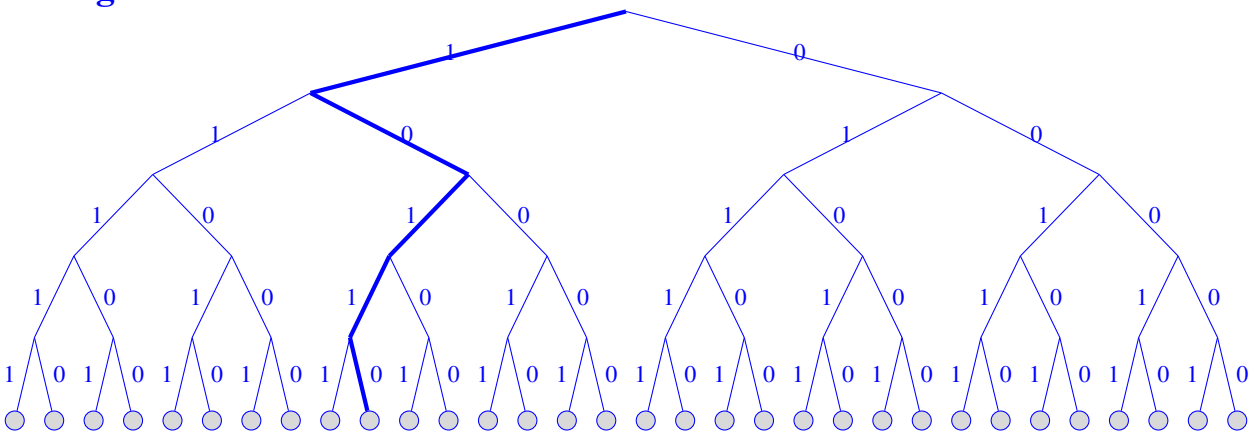
Andere Lösungsmöglichkeit: Die Anzahl der Möglichkeiten einen Wenz mit 4 zu erhalten ist gegeben durch die Anzahl der Möglichkeiten vier Unter zu ziehen, $C(4,4)$, multipliziert mit der Anzahl der Möglichkeiten, die vier Nicht-Unter aus den verbleibenden 28 Karten zu kombinieren, $C(28,4)$. Für die gesuchte Wahrscheinlichkeit folgt dann:

$$p = \frac{\binom{4}{4}\binom{28}{4}}{\binom{32}{8}} = \frac{1\cdot 8!(32-8)!28!}{32!4!(28-4)!} = \frac{5\cdot 6\cdot 7\cdot 8}{29\cdot 30\cdot 31\cdot 32} = \frac{7}{29\cdot 31\cdot 4} = 2{,}947\cdot 10^{-3} \quad .$$

Übungsaufgaben zu Kapitel 2.5

A 2.19 (L0) Wie viele Elementarentscheidungen sind zur Identifikation des Binärwortes 10110 erforderlich? Skizzieren Sie den zugehörigen Entscheidungsbaum (Codebaum) und markieren Sie den durch die getroffenen Elementarentscheidungen definierten Weg durch diesen Baum.

Lösung



A 2.20 (T0) Welcher Prozess bei der Nachrichtenverarbeitung ist prinzipiell immer mit einem Energieaufwand verbunden?

Lösung

Das Speichern und Löschen von Daten.

A 2.21 (T0) Was ist mit damit gemeint, wenn man sagt, das Auftreten von Buchstabenpaaren in deutschen Texten sei korreliert?

Lösung

Berechnet man die Auftrittswahrscheinlichkeiten von Zeichenpaaren durch Multiplikation der Auftrittswahrscheinlichkeiten der Einzelzeichen, so wird dabei angenommen, dass die Auftrittswahrscheinlichkeiten der Zeichen unabhängig vom vorangehenden Zeichen ist. Dies ist jedoch in der Praxis nicht der Fall: Die Zeichen sind korreliert, d. h. es handelt sich um bedingte Wahrscheinlichkeiten.

A 2.22 (M2) Gegeben sei die Menge $M = \{\text{a, b, c, d}\}$.

a) Wie viele Teilmengen von M gibt es? Geben Sie diese explizit an.

b) Auf wie viele verschiedene Arten lässt sich die Menge M in zwei nichtleere, disjunkte Teilmengen zerlegen, so dass die Vereinigungsmenge dieser beiden Teilmengen wieder M ergibt? Teilmengen sind disjunkt, wenn ihr Durchschnitt leer ist.

c) Auf wie viele verschiedene Arten lassen sich aus der Menge M zwei nichtleere, disjunkte Teilmengen bilden, wenn die Vereinigung der beiden Teilmengen nicht unbedingt M ergeben muss.

Lösung

a) Es gibt $N = 16$ Teilmengen, nämlich:
{ },
{a}, {b}, {c}, {d},
{a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d},
{a, b, c}, {a, b, d}, {a, c, d}, {b, c, d},
{a, b, c, d}

Die Elemente der Teilmengen ergeben sich durch Kombinationen ohne Wiederholungen von $m = 0,1,2,3,4$ aus $n = 4$ Elementen. Die Reihenfolge der Auswahl spielt keine Rolle. Man erhält daher:

$$N = \sum_{m=1}^{n} C(m,n) = \sum_{m=1}^{n} \binom{n}{m} = \binom{4}{0} + \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 1+4+6+4+1 = 2^4 = 16 \quad .$$

Von diesen 16 Teilmengen sind 15 nicht leer.

b) Folgende Zerlegungen von $M = \{\text{a, b, c, d}\}$ in zwei disjunkte Teilmengen, deren Vereinigung M ist, sind möglich:
{a, b, c}, {d}
{a, b, d}, {c}
{a, c, d}, {b}
{b, c, d}, {a}

{a, b}, {c, d}
{a, c}, {b, d}
{a, d}, {b, c}

Es handelt sich wieder um Kombinationen ohne Wiederholungen von $m = 1$ und $m = 2$ Elementen, wobei die Reihenfolge keine Rolle spielt:

$$C(1,4) = \binom{4}{1} = \binom{4}{3} = 4 \quad \text{und} \quad C(2,4)/2 = \binom{4}{2}/2 = 3 \quad .$$

c) Es gibt 25 derartige Zerlegungen. Zunächst gehören die 7 in b) gefundenen Mengen zu den gesuchten Zerlegungen. Außerdem gehören noch die $C(1,4)(C(1,3)/2) = 6$ disjunkten Zerlegungen dazu, bei denen beide Teilmengen nur ein Element enthalten sowie die $C(1,4)C(2,3) = 12$ Zerlegungen, bei denen die eine Menge ein Element enthält und die andere Menge zwei Elemente.

A 2.23 (M1) Die folgende Aufgabe ist als das Botenproblem bekannt. Zum Übertragen einer Nachricht stehen zwei Kanäle A und B zur Verfügung, wobei die Wahrscheinlichkeit für das Auftreten eines Fehlers bei der Übertragung einer Nachricht für Kanal B doppelt so hoch ist wie für Kanal A. Mit welcher der beiden Strategien erreicht eine Nachricht den Empfänger mit größerer Sicherheit:

a) Die Nachricht wird über Kanal A gesendet.

b) Die Nachricht wird zweimal über Kanal B gesendet.

Lösung

Es sei p_{fA} die Wahrscheinlichkeit dafür, dass bei der Übertragung einer Nachricht über Kanal A ein Fehler auftritt und p_{fB} die Wahrscheinlichkeit dafür, dass die Nachricht über Kanal B fehlerhaft übertragen wird. Nach Voraussetzung ist $p_{fB} = 2p_{fA}$.
Es gilt dann für die Wahrscheinlichkeiten p_A und p_B der korrekten Übertragung über Kanal A bzw. B:

$$p_A = 1 - p_{fA} \quad \text{und} \quad p_B = 1 - p_{fB} = 1 - 2p_{fA} \quad .$$

Die Wahrscheinlichkeit p_{BB} dafür, dass mindestens eine der zweimal über Kanal B gesendeten identischen Nachrichten ohne Störung den Empfänger erreicht ist:

$$\begin{aligned} p_{BB} =& P(\text{beide Nachrichten bleiben ungestört}) + \\ & P(\text{nur Nachrichten 1 bleibt ungestört}) + \\ & P(\text{nur Nachrichten 2 bleibt ungestört}) \\ =& p_B p_B + p_B(1-p_B) + (1-p_B)p_B = 2p_B - p_B^2 = p_A - p_B^2 < p_A \quad . \end{aligned}$$

Es ist also günstiger, die Nachricht nur einmal über Kanal A zu senden.

A 2.24 (L2) Gegeben seien ein Alphabet A sowie zwei Worte $s = s_1 s_2 s_3 \ldots s_n$ und $t = t_1 t_2 t_3 \ldots t_m$ mit $s_j, t_j \in A$ aus dem Nachrichtenraum A^* über diesem Alphabet. Geben Sie eine exakte Definition der lexikografischen Ordnung $s < t$.

Lösung

Für „s ist lexikografisch vor t“ wird hier $s < t$ geschrieben. Die Definition der lexikografischen Ordnung lässt sich dann beispielsweise in Form des folgenden Algorithmus angeben.
SETZE k = Minimum von n und m
FÜR $i = 1$ bis k
    WENN $s_i < t_i$ DANN GIB AUS „$s < t$“; ENDE
    WENN $t_i < s_i$ DANN GIB AUS „$t < s$“; ENDE
WENN $n < m$ DANN $s < t$; ENDE
WENN $m < n$ DANN $t < s$; ENDE
GIB AUS „t und s sind identisch“; ENDE

A 2.25 (M1) Eine Person wählt in Gedanken zwei verschiedene natürliche Zahlen aus, die beide nicht kleiner als 1 und nicht größer als 100 sind. Wie viele Fragen muss man mindestens stellen, um die beiden Zahlen mit Sicherheit bestimmen zu können, wenn die befragte Person immer nur mit „ja“ oder „nein“ antwortet?

Lösung

Der Versuch kann $\binom{100}{2} = 2950$ verschiedene Ausgänge haben.
Es handelt sich dabei um Kombinationen ohne Wiederholungen. Da auf jede Frage mit ja oder nein geantwortet wird, lässt sich jede möglich Kombination von Fragen und Antworten als Ast in einen Binärbaum interpretieren, der 4950 Endknoten (Blätter) hat. Die Tiefe t dieses Baumes, d. h. die Anzahl der Kanten der Äste von der Wurzel bis zu einem Blatt ist, da es sich um einen Binärbaum handelt, durch $t = \mathrm{ld}(4950) \approx 12{,}27 \approx 13$ gegeben. Es werden also 13 (richtig gestellte!) Fragen immer zur Ermittlung beider Zahlen ausreichen. Im Einzelfall kann man natürlich Glück haben und mit weniger als 13 Fragen die beiden Zahlen erraten.
Dasselbe Resultat erhält man auch durch folgende Überlegung: Wenn alle Ausgänge gleich wahrscheinlich sind, wird die zugehörige Entropie $H = \mathrm{ld}(4950) \approx 12{,}27$ sein. Auch daraus folgt, dass 13 Fragen genügen. Über eine Strategie zur Auswahl vernünftiger Fragen ist damit natürlich nichts gesagt.

A 2.26 (M1) Gegeben sei das Alphabet $A = \{\mathrm{a}, \mathrm{e}, \mathrm{i}, \mathrm{o}, \mathrm{u}\}$ mit den Auftrittswahrscheinlichkeiten $P(\mathrm{a}) = 0{,}25$, $P(\mathrm{e}) = 0{,}20$, $P(\mathrm{i}) = 0{,}10$, $P(\mathrm{o}) = 0{,}30$, $P(\mathrm{u}) = 0{,}15$. Berechnen Sie die Informationsgehalte der Zeichen von A und die Entropie von A.

Lösung

$$\text{Informationsgehalt: } I(x_i) = \mathrm{ld}\,\frac{1}{P(x_i)}, \quad \text{Entropie: } H = \sum_{i=1}^{n} P(x_i) I(x_i) \quad .$$

Zeichen	$P(x_i)$	$I(x_i)$
a	0,25	2,0000
e	0,20	2,3219
i	0,10	3,3219
o	0,30	1,7370
u	0,15	2,7370

$$\begin{aligned} H &= 0{,}25 \cdot 2{,}0000 + 0{,}20 \cdot 2{,}3219 + 0{,}10 \cdot 3{,}3219 + 0{,}30 \cdot 1{,}7370 + 0{,}15 \cdot 2{,}7370 \\ &= 0{,}5 + 0{,}46438 + 0{,}33219 + 0{,}5211 + 0{,}41055 = 2{,}22822 \text{ [Bit/Zeichen]} \quad . \end{aligned}$$

A 2.27 (M2) Berechnen Sie den mittleren Informationsgehalt des Textes dieser Aufgabe. Am besten schreiben Sie dazu ein einfaches Programm in C oder Java.

Lösung

Zunächst ist zu abzuzählen, wie oft die einzelnen Zeichen x_i im Text der Aufgabe auftreten. So findet man beispielsweise die Häufigkeiten $h_a = 8$ für das Zeichen „a“ und $h_T = 1$ für das Zeichen „T“. Die Gesamtzahl der Zeichen ist $n = 141$. Daraus berechnet man nun gemäß $p_i = h_i/n$ die Auftrittswahrscheinlichkeiten p_i als relative Häufigkeiten. Daraus folgt für die Informationsgehalte I_i der Zeichen.
Dies kann man etwas mühsam per Hand berechnen. Hier wurde ein C-Programm verwendet. Das Ergebnis $H \approx 4{,}243$ Bit/Zeichen mit den Zwischenergebnissen ist unten aufgelistet.

i	ASCII	x_i	h_i	p_i	I_i
1	32		20	0.1418	2.8176
2	46	.	2	0.0142	6.1396
3	65	A	2	0.0142	6.1396
4	66	B	1	0.0071	7.1396
5	67	C	1	0.0071	7.1396
6	73	I	1	0.0071	7.1396
7	74	J	1	0.0071	7.1396
8	80	P	1	0.0071	7.1396
9	83	S	2	0.0142	6.1396
10	84	T	1	0.0071	7.1396
11	97	a	8	0.0567	4.1396
12	98	b	3	0.0213	5.5546
13	99	c	3	0.0213	5.5546
14	100	d	5	0.0355	4.8176
15	101	e	23	0.1631	2.6160
16	102	f	3	0.0213	5.5546
17	103	g	3	0.0213	5.5546
18	104	h	4	0.0284	5.1396
19	105	i	9	0.0638	3.9696
20	108	l	2	0.0142	6.1396
21	109	m	5	0.0355	4.8176
22	110	n	11	0.0780	3.6801
23	111	o	4	0.0284	5.1396
24	114	r	8	0.0567	4.1396
25	115	s	7	0.0496	4.3322
26	116	t	6	0.0426	4.5546
27	117	u	2	0.0142	6.1396
28	118	v	1	0.0071	7.1396
29	120	x	1	0.0071	7.1396
30	122	z	1	0.0071	7.1396

A 2.28 (M2) Gegeben sei das Alphabet $A = \{\text{a, b, c, d}\}$.

a) Wie viele verschiedene Worte der Länge 8 lassen sich mit diesem Alphabet bilden?

b) Wie viele Worte enthält der eingeschränkte Nachrichtenraum A^8, d. h. wie viele verschiedene Worte mit der Maximallänge 8 lassen sich mit diesem Alphabet bilden?

c) Wie viele Worte können mit dem Alphabet A gebildet werden, wenn die Buchstaben innerhalb der Worte lexikografisch angeordnet sein sollen und wenn sich keine Buchstaben innerhalb eines Wortes wiederholen dürfen?

Lösung

a) Es handelt sich um Variationen mit Wiederholungen aus einem Alphabet mit 4 Zeichen. Also gibt es $4^8 = 65536$ Worte der Länge 8.

b) Dazu muss man die Anzahlen aller Worte mit den Längen 1 bis 8 addieren:
$4^1 + 4^2 + 4^3 + 4^4 + 4^5 + 4^6 + 4^7 + 4^8 = 4 + 16 + 64 + 256 + 1024 + 4096 + 16384 + 65536 = 87380.$

c) Es handelt sich um die Summe der Kombinationen von Worten der Länge 1, 2, 3, und 4 ohne Wiederholungen.

$$\binom{4}{1} = \frac{4!}{1!(4-1)!} = 4 \qquad \{a,b,c,d\}$$
$$\binom{4}{2} = \frac{4!}{2!(4-2)!} = 6 \qquad \{ab,ac,ad,bc,bd,cd\}$$
$$\binom{4}{3} = \frac{4!}{3!(4-3)!} = 4 \qquad \{abc,abd,acd,bcd\}$$
$$\binom{4}{4} = \frac{4!}{4!(4-4)!} = 1 \qquad \{abcd\}$$

3 Lösungen zu Kapitel 3 – Codierung

Übungsaufgaben zu Kapitel 3.1

A 3.1 (L1) Geben Sie für die folgenden Manipulationen auf ASCII-Zeichen einfache logische Operationen mit Bit-Masken an:

a) Extraktion des Zahlenwertes aus den ASCII-Codes für die Ziffern 0 bis 9.

b) Umwandlung von Kleinbuchstaben in Großbuchstaben und umgekehrt.

Lösung

a) Die ASCII-Codes der Ziffern 0 bis 9 lauten: 011 0000 = „0“ bis 011 1001 = „9“.
Ist bekannt, dass der umzuwandelnde Code c einer Ziffer entspricht, so kann man aus c den entsprechenden Zahlenwert z sehr einfach durch folgende UND-Verknüpfung extrahieren:

$$z = c \wedge 0001111 \qquad \text{für 7-Bit ASCII-Codes}$$
$$z = c \wedge 00001111 = c \wedge 0F_{16} \qquad \text{für 8-Bit ASCII-Codes}$$

Soll zusätzlich sichergestellt werden, dass das Codewort c tatsächlich eine Ziffer ist, so wäre zu prüfen, ob $0110000 < c < 0111001$ gilt.

Man könnte zur Umwandlung von c in den binären Zahlenwert auch $0110000_2 = 48_{10}$ von c subtrahieren oder die logische Verknüpfung c XOR 0011 0000 bilden. Dies dauert jedoch länger als die logische UND-Verknüpfung.

b) Kleinbuchstaben: a = 1100001 bis z = 1111010
danach Umlaute und schließlich ß = 1111110
Großbuchstaben: A = 1000001 bis Z = 1011010
danach Umlaute und schließlich ~ = 1011110
Offenbar unterscheiden sich die Codes nur in einem Bit. Die Umwandlung von Kleinbuchstaben in Großbuchstaben erfolgt daher durch AND-Verknüpfung mit der Maske $101\,1111 = 5F_{16}$ (7-Bit-ASCII) bzw. mit der Maske $1101\,1111 = DF_{16}$ (8-Bit-ASCII). Die Umwandlung von Großbuchstaben in Kleinbuchstaben erfolgt durch die OR-Verknüpfung mit der Maske $0100000 = 20_{16}$ (7-Bit-ASCII). Bei 8-Bit ASCII-Code lautet die Maske $00100000 = 20_{16}$.

A 3.2 (T0) Was ist ein Blockcode, was ist der BCD-Code? Was sind Pseudo-Tetraden?

Lösung

Ein Block-Code ist ein Code mit konstanter Wortlänge, d. h. alle Codewörter haben identische Wortlängen. Diese stimmt dann offenbar mit der mittleren Wortlänge überein.
Für die Codierung von Zahlen verwendet man vor allem für finanzmathematische Anwendungen den BCD-Code (von Binary Coded Decimal). Dies erlaubt das Rechnen mit Geldbeträgen im Dezimalsystem, damit auch bei sehr großen Summen stellengenaue Ergebnisse garantiert werden können.

Beim BCD und damit verwandten Codes zur Codierung von Ziffern werden für jede Ziffer vier binäre Stellen verwendet. Nach diesem Schema konstruierte Codes werden als Tetraden-Codes bezeichnet (von griechisch tetra = vier). Da es 16 verschiedene Codewörter mit Wortlänge 4 gibt, aber für die Codierung der Ziffern von 0 bis 9 nur 10 Wörter benötigt werden, existieren offenbar 6 Vier-Bit-Wörter, denen keine Ziffer entspricht. Man bezeichnet diese redundanten Wörter als Pseudo-Tetraden.

A 3.3 (L1) Ein Code ist eine umkehrbar eindeutige Abbildung von einem Nachrichtenraum A^* in einen Nachrichtenraum B^*. Welche Aussagen sind richtig:

a) Es kann Elemente aus A^* geben, die kein Bild in B^* haben.

b) Es kann Elemente in B^* geben, die kein Urbild in A^* haben.

c) Es ist zulässig, dass ein Element aus A^* auf zwei verschiedene Elemente aus B^* abgebildet wird.

d) Es ist ausgeschlossen, dass zwei verschiedene Elemente aus A^* auf dasselbe Element aus B^* abgebildet werden.

Lösung

a) ist falsch, da ein Code jedes Element aus A^* erfassen muss.

b) ist richtig, da die Abbildung von A^* *in* B^* und nicht *auf* B^* definiert ist.

c) ist falsch, da die Abbildung umkehrbar eindeutig (eineindeutig) sein muss.

d) ist richtig, da die Abbildung umkehrbar eindeutig (eineindeutig) sein muss.

A 3.4 (M2) Das Alphabet $A = \{\text{a, e, i, o, u}\}$ mit den Auftrittswahrscheinlichkeiten $P(\text{a}) = 0{,}25$, $P(\text{e}) = 0{,}20$, $P(\text{i}) = 0{,}10$, $P(\text{o}) = 0{,}30$ und $P(\text{u}) = 0{,}15$ sei mit $B = \{11, 10, 010, 00, 011\}$ binär codiert. Berechnen Sie die mittlere Wortlänge und die Redundanz dieses Codes.

Lösung

Mittlere Wortlänge:

$$L = \sum_{i=1}^{n} p_i l_i \quad ,$$

Redundanz:

$$R = L - H \quad .$$

Hier ist H die Entropie. Es ergibt sich:

x_i	$P(x_i)$	$I(x_i)$	l_i
a	0,25	2,0000	2
e	0,20	2,3219	2
i	0,10	3,3219	3
o	0,30	1,7370	2
u	0,15	2,7370	3

Entropie: $H = 2{,}2282$ Bit/Zeichen.
Mittlere Wortlänge: $L = 2{,}25$ Bit/Zeichen.
Redundanz: $R = 2{,}25 - 2{,}2282 = 0{,}0218$ Bit/Zeichen.

Übungsaufgaben zu Kapitel 3.2

A 3.5 (M3) Gegeben sei das unten tabellierte Alphabet $\{x_i\}$ mit den zugehörigen Auftrittswahrscheinlichkeiten $\{p_i\}$:

x_i	A	E	I	O	U	Y
p_i	0,105	0,22	0,105	0,04	0,45	0,08

a) Berechnen Sie die Informationsgehalte $I(x_i)$ sowie die Entropie.

b) Bilden Sie einen Binärcode durch Aufstellen eines einfachen Codebaums, wobei für die Wortlängen der Codewörter die ganzzahlig aufgerundeten zugehörigen Informationsgehalte der zugehörigen Zeichen x_i gewählt werden sollen.

c) Bilden Sie den optimalen Binärcode mithilfe des Huffman-Algorithmus.

d) Bilden Sie einen Binärcode unter Verwendung des Fano-Algorithmus.

e) Berechnen Sie für die in b), c) und d) gebildeten Codes die mittleren Wortlängen und die Redundanzen.

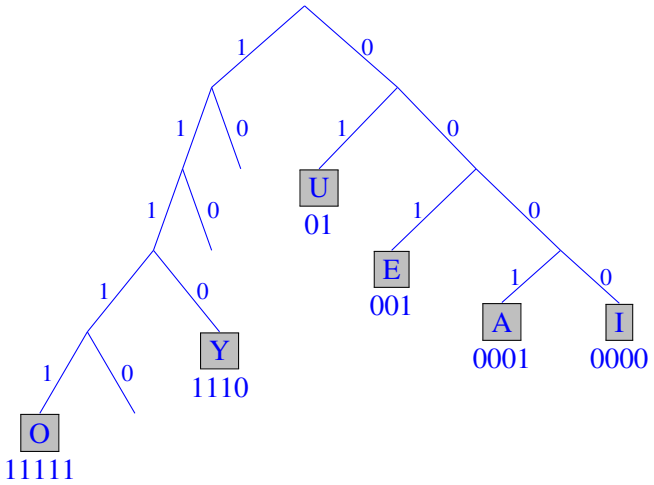
Lösung

a) Tabelle mit Informationsgehalt:

x_i	p_i	$I(x_i)$
A	0,105	3,2515
E	0,22	2,1844
I	0,105	3,2515
O	0,04	4,6439
U	0,45	1,1520
Y	0,08	3,6439

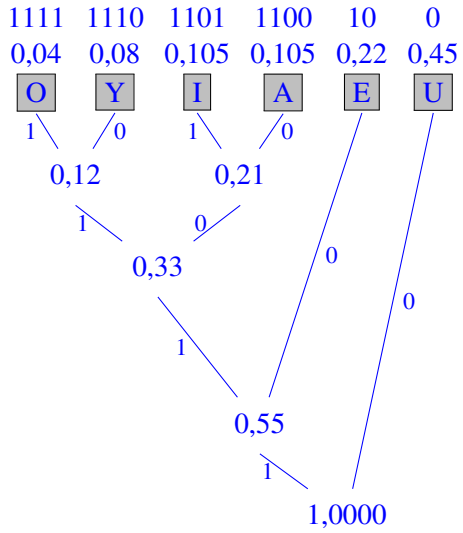
Entropie: $H = 2{,}1590$ Bit/Zeichen.

b) Beispiel für einen einfachen Code:



x_i	p_i	$I(x_i)$	l_i	einfacher Code
A	0,105	3,2515	4	0001
E	0,22	2,1844	3	001
I	0,105	3,2515	4	0000
O	0,04	4,6439	5	11111
U	0,45	1,1520	2	01
Y	0,08	3,6439	4	1110

c) Der Huffman-Code entsteht durch Konstruktion des folgenden Baums:



d) Der Fano-Algorithmus liefert den folgenden Code:

x_i	p_i	$\sum p_i$	Code
U	0,45	1,00	1
E	0,22	0,55	011
A	0,105	0,33	010
I	0,105	0,225	001
Y	0,08	0,12	0001
O	0,04	0,04	0000

e) einfacher Code:
Mittlere Wortlänge: $L = 2{,}92$ Bit/Zeichen.
Redundanz: $R = 2{,}92 - 2{,}1590 = 0{,}761$ Bit/Zeichen.

Huffman:
Mittlere Wortlänge: $L = 2{,}21$ Bit/Zeichen.
Redundanz: $R = 2{,}21 - 2{,}1590 = 0{,}051$ Bit/Zeichen.

Fano:
Mittlere Wortlänge: $L = 2{,}22$ Bit/Zeichen.
Redundanz: $R = 2{,}22 - 2{,}1590 = 0{,}061$ Bit/Zeichen.

A 3.6 (L2) Gegeben sei das binäre Alphabet $B = \{0,1\}$. Geben Sie alle Teilmengen des Nachrichtenraums über B^* an, welche folgende Bedingungen erfüllen: Die Teilmengen umfassen mindestens drei Wörter; die Wörter bestehen aus höchstens zwei Zeichen; die Wörter erfüllen die Fano-Bedingung.

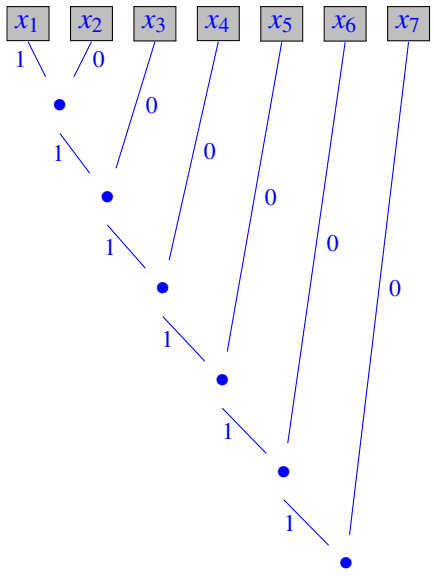
Lösung

Die gesuchten Teilmengen lauten:
{00, 01, 10, 11},
{00, 01, 10}, {00, 01, 11}, {00, 10, 11}, {01, 10, 11},
{0, 10, 11}, {1, 00, 01}

A 3.7 (L3) Gegeben sei ein Alphabet mit n Zeichen. Bestimmen Sie die Wortlänge des Zeichens mit dem längsten Codewort, das eine binäre Huffman-Codierung im Extremfall liefern kann. Welche Bedingung muss dann für die Auftrittswahrscheinlichkeiten p_i gelten?

Lösung

Das Codewort des längsten Wortes hat die Länge $n-1$, mindestens jedoch 1. Dies folgt aus dem sich im Extremfall ergebenden entarteten Huffman-Baum, der nachfolgend für den Fall $n = 7$ angegeben ist. Das längste Codewort 111111 für das Zeichen x_1 hat also die Länge 6.



Dieser Extremfall ergibt sich, wenn folgende Bedingung erfüllt ist: Höchstens eine der verbleibenden Auftrittswahrscheinlichkeiten von Einzelzeichen ist kleiner als die durch Zusammenfassung der beiden Knoten mit den beiden kleinsten Auftrittswahrscheinlichkeiten gebildete Summe. Ordnet man die Auftrittswahrscheinlichkeiten p_i mit $i = 1$ bis n aufsteigend der Größe nach, so muss also gelten:

$$\sum_{j=1}^{k} p_j \leq p_{k+1} \quad \text{für alle } k \text{ von 1 bis } n-1.$$

A 3.8 (M3) Gegeben sei das Alphabet {A, B} mit den Auftrittswahrscheinlichkeiten $p_A = 0{,}7$ und $p_B = 0{,}3$. Geben Sie die Huffman-Codes für die Codierung von Einzelzeichen, von Gruppen aus zwei Zeichen, von Gruppen aus drei Zeichen und von Gruppen aus vier Zeichen an. Berechnen Sie dazu jeweils die mittleren Wortlängen und die Redundanzen in Bit/Zeichen.

Lösung

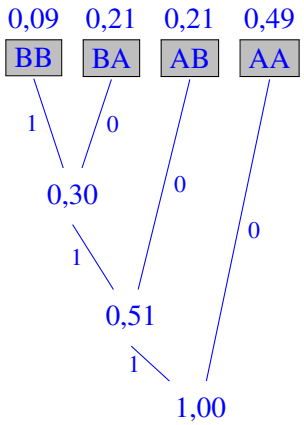
Entropie: $H = 0{,}8813$ Bit/Zeichen.

Codierung von Einzelzeichen: A $\leftrightarrow$ 0, B $\leftrightarrow$ 1
Mittlere Wortlänge: $L = 1$ Bit/Zeichen.
Redundanz: $R = 1 - 0{,}8813 = 0{,}1187$ Bit/Zeichen.

Codierung von Zweiergruppen:

x_i	p_i	Code
AA	0,49	0
AB	0,21	10
BA	0,21	110
BB	0,09	111

Huffman-Baum:

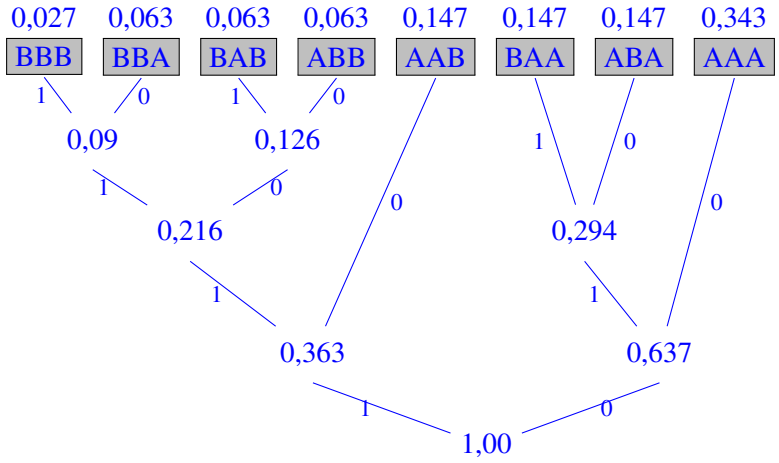


Mittlere Wortlänge: $L = 1{,}81$ Bit/2 Zeichen $= 0{,}905$ Bit/Zeichen.
Redundanz: $R = 0{,}905 - 0{,}8813 = 0{,}0237$ Bit/Zeichen.

Codierung von Dreiergruppen:

x_i	p_i	Code
AAA	0,343	00
AAB	0,147	10
ABA	0,147	010
BAA	0,147	011
ABB	0,063	1100
BAB	0,063	1101
BBA	0,063	1110
BBB	0,027	1111

Huffman-Baum:



Mittlere Wortlänge: $L = 2{,}7260$ Bit/3 Zeichen $= 0{,}9087$ Bit/Zeichen.
Redundanz: $R = 0{,}9087 - 0{,}8813 = 0{,}0274$ Bit/Zeichen.
Offenbar ist das Ergebnis etwas schlechter als bei der Codierung von Zweiergruppen!

Codierung von Vierergruppen:

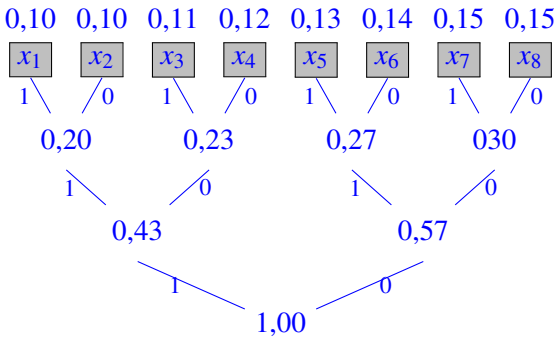
x_i	p_i	Code
AAAA	0,2401	10
AAAB	0,1029	000
AABA	0,1029	001
ABAA	0,1029	010
BAAA	0,1029	011
AABB	0,0441	11100
ABAB	0,0441	11101
BAAB	0,0441	11000
ABBA	0,0441	11001
BABA	0,0441	11110
BBAA	0,0441	11010
ABBB	0,0189	110110
BABB	0,0189	110111
BBAB	0,0189	111110
BBBA	0,0189	1111110
BBBB	0,0081	1111111

Mittlere Wortlänge: $L = 3{,}5672$ Bit/4 Zeichen $= 0{,}8918$ Bit/Zeichen.
Redundanz: $R = 0{,}8918 - 0{,}8813 = 0{,}0105$ Bit/Zeichen.
Gegenüber der Codierung mit Einzelzeichen hat sich die Redundanz um den Faktor 10 verringert. Gruppen-Codierung ist tatsächlich eine Möglichkeit, die Redundanz bis nahe 0 zu reduzieren. Allerdings sind hierbei die Korrelationen zwischen den Zeichen einer Gruppe nicht berücksichtigt. Die Berücksichtigung der Korrelationen (etwa durch die LZW-Codierung) erlaubt eine weitere wesentliche Redundanzreduktion.

A 3.9 (M2) Welche Bedingungen sind dafür hinreichend und notwendig, dass der Huffman-Algorithmus einen Blockcode liefert?

Lösung

Unter einem Block-Code versteht man einen Code mit konstanter Wortlänge, d. h. alle Codewörter haben dieselbe Länge. Damit dies erreicht wird, muss der Huffman-Baum ausgeglichen sein, dafür ist erforderlich, dass die Anzahl der Zeichen eine Potenz von 2 ist. Für jeden Knoten auf einem bestimmten Niveau des Huffman-Baums muss außerdem gelten, dass die ihm zugeordnete Wahrscheinlichkeit größer ist, als jede Wahrscheinlichkeit in dem vorhergehenden Niveau. Dies ist zum Beispiel jedenfalls dann der Fall, wenn alle Auftrittswahrscheinlichkeiten gleich sind. Die Bedingung kann aber weiter gefasst werden: es müssen auf jedem Niveau, die den Knoten zugeordneten Wahrscheinlichkeiten größer sein, als die Wahrscheinlichkeiten aller vorangehenden Knoten. Das folgende Beispiel demonstriert dies:



A 3.10 (L1) Genügt die Deutsche Sprache der Fano-Bedingung?

Lösung

Nein: Es gibt Wörter, die Präfix eines anderen Wortes sind (und zwar sehr viele).
Beispiel: *Hund* und *Hundehütte*.

A 3.11 (T0) Warum führt die LZW-Kompression in der Regel zu einer stärkeren Redundanz-Minimierung als das Huffman-Verfahren?

Lösung

Beim Huffman-Verfahren werden nur Einzelzeichen codiert, beim LZW-Verfahren dagegen Zeichengruppen. Dadurch werden auch bedingte Auftrittswahrscheinlichkeiten bzw. Korrelationen berücksichtigt, so dass in Nachrichten, die solche Korrelationen enthalten, eine weitere Redundanzminimierung erreicht wird.

A 3.12 (T1) Ordnen Sie die fünf Kompressionsverfahren MPEG, JPEG, LZW, RLC (Run Length Code), PDC (Prädiktive Differenzcodierung) den folgenden fünf Datenarten zu, für welche sie jeweils am besten geeignet sind: Videofilm, Computerprogramm, CAD-Strichzeichnung, Digitalfoto, Temperaturmesswerte mit 100 Messungen/s.

Lösung

Videofilm	MPEG
Computerprogramm	LZW
CAD-Strichzeichnung	RLC
Mit einer Digitalkamera aufgenommenes Foto	JPEG
Temperaturmesswerte mit 100 Messungen/sec	PDC

A 3.13 (L3) Komprimieren Sie das Wort HUMUHUMUNUKUNUKUAPUAA (ein nach einem Fischchen benannter Cocktail aus Hawaii) mit arithmetischer und LZW-Kompression.

Lösung

Arithmetische Kompression

Der Quelltext HUMUHUMUNUKUNUKUAPUAA ist arithmetisch zu codieren. Dazu werden zunächst die Auftrittswahrscheinlichkeiten p_i der Einzelzeichen x_i ermittelt. Danach wird jedem Zeichen ein Intervall $[u,o[$ mit zum Intervall gehöriger Untergrenze u und nicht mehr zum Intervall gehöriger Obergrenze o zugeordnet, dessen Länge zu der entsprechenden Auftrittswahrscheinlichkeit proportional ist. Man erhält folgende Tabelle:

Zeichen c_i	Auftrittswahrsch. p_i	Intervall $[u;o[$
A	$3/21$	$[0{,}000000000000000; 0{,}142857142847143[$
H	$2/21$	$[0{,}142857142847143; 0{,}238095238095238[$
K	$2/21$	$[0{,}238095238095238; 0{,}333333333333333[$
M	$2/21$	$[0{,}333333333333333; 0{,}428571428571429[$
N	$2/21$	$[0{,}428571428571429; 0{,}523809523809524[$
P	$1/21$	$[0{,}523809523809524; 0{,}571428571428571[$
U	$9/21$	$[0{,}571428571428571; 1{,}000000000000000[$

Zur eigentlichen Codierung wird als Startwert das Intervall $[0{,}0; 1{,}0[$ mit der Untergrenze $u = 0{,}0$ und der (nicht mehr zum Intervall gehörenden) Obergrenze $o = 1{,}0$ hergenommen. Die Untergrenze und die Obergrenze dieses Intervalls werden nun durch die Schritt für Schritt eingelesenen Zeichen des Textes entsprechend der Unter- und Obergrenze ihres Intervalls immer weiter eingegrenzt. Dies geschieht durch Berechnung immer neuer, monoton wachsender Untergrenzen und monoton fallender Obergrenzen. Damit geschieht die Kompression Schritt für Schritt wie folgt:

Zeichen c_i	Intervalllänge d	Untergrenze u	Obergrenze o
Initialisierung	0,0	0,0	1,0
H	1.000000000000000	0.142857142857143	0.238095238095238
U	0.095238095238095	0.197278911564626	0.238095238095238
M	0.040816326530612	0.210884353741497	0.214771622934888
U	0.003887269193392	0.213105650423435	0.214771622934888
H	0.001665972511454	0.213343646496499	0.213502310545209
U	0.000158664048710	0.213434311667191	0.213502310545209
M	0.000067998878019	0.213456977959864	0.213463454043484
U	0.000006476083621	0.213460678579076	0.213463454043484
N	0.000002775464409	0.213461868063822	0.213462132393766
U	0.000000264329944	0.213462019109504	0.213462132393766
K	0.000000113284262	0.213462046081948	0.213462056870925
U	0.000000010788977	0.213462052247077	0.213462056870925
N	0.000000004623847	0.213462054228726	0.213462054669093
U	0.000000000440366	0.213462054480364	0.213462054669093
K	0.000000000188728	0.213462054525300	0.213462054543274
U	0.000000000017974	0.213462054535571	0.213462054543274
A	0.000000000007703	0.213462054535571	0.213462054536671
P	0.000000000001100	0.213462054536147	0.213462054536199
U	0.000000000000052	0.213462054536177	0.213462054536199
A	0.000000000000022	0.213462054536177	0.213462054536180
A	0.000000000000003	0.213462054536177	0.213462054536177

Das Ergebnis lautet also: $x = 0{,}213462054536177$. Es sind also 15 Dezimalstellen erforderlich, eine im Vergleich zu den ursprünglich 21 Zeichen erhebliche Kompressionsleistung, zumal die Dezimalstellen als Block-Code mit weniger Bit darstellbar sind als das Ausgangsalphabet der ASCII-Zeichen. In einem professionellen Programm würde man natürlich Hexadezimalziffern statt Dezimalziffern verwenden, so dass die Darstellung noch etwas komprimierter wäre und 4 Bit pro Ziffer für die Codierung genügen. Dazu kommt allerdings, dass die Häufigkeiten der Einzelzeichen ebenfalls mit übertragen werden müssen, da diese für die Dekompression erforderlich sind.
Aus dem codierten Text, also der Gleitpunktzahl $x = 0{,}213462054536177$, kann der Ursprungstext durch Umkehrung des Kompressionsalgorithmus wie folgt wieder gewonnen werden.

Die Berechnung ist per Hand auch mit Taschenrechner sehr mühsam. Es empfiehlt sich, dafür ein Programm zu schreiben.

LZW-Kompression

Beim LZW-Algorithmus wird zunächst eine Code-Tabelle mit allen Einzelzeichen des Quell-Alphabets, wobei die Wortlänge des Ziel-Alphabets um einige Bit größer gewählt wird, als die des Quell-Alphabets. Im Laufe der Kompression wird dann die Code-Tabelle um Codes für Zeichengruppen erweitert, so dass die Kompression immer effizienter wird.
Für die Code-Tabelle werden in diesem Beispiel Einträge mit vier Bit gewählt. Da das Quell-Alphabet nur sieben Zeichen umfasst, bleiben nach der Vorbesetzung noch neun Plätze für spätere Einträge frei:

Zeichenkette	Ausgabe-Code (binär)	Ausgabe-Code (hex)
A	0000	0
H	0001	1
K	0010	2
M	0011	3
N	0100	4
P	0101	5
U	0110	6
–	0111	7
–	1000	8
–	1001	9
–	1010	A
–	1011	B
–	1100	C
–	1101	D
–	1110	E
–	1111	F

Der Codierungsvorgang läuft damit folgendermaßen ab:

Schritt	Zeichen c	Präfix P	Eintrag in Code-Tabelle	Ausgabe
0	–	–	Vorbesetzung	–
1	H	H	–	–
2	U	U	HU = 7	1
3	M	M	UM = 8	6
4	U	U	MU = 9	3
5	H	H	UH = A	6
6	U	HU	–	–
7	M	M	HUM = B	7
8	U	MU	–	–
9	N	N	MUN = C	9
10	U	U	NU = D	4
11	K	K	UK = E	6
12	U	U	KU = F	2
13	N	N	Tabelle voll	6
14	U	NU		–
15	K	K		D
16	U	KU		–
17	A	A		F
18	P	P		0
19	U	U		5
20	A	A		6
21	A	A		0
22				0

Nach der Kompression hat die Code-Tabelle die folgende Form:

Zeichenkette	Ausgabe-Code (binär)	Ausgabe-Code (hex)
A	0000	0
H	0001	1
K	0010	2
M	0011	3
N	0100	4
P	0101	5
U	0110	6
HU	0111	7
UM	1000	8
MU	1001	9
UH	1010	A
HUM	1011	B
MUN	1100	C
NU	1101	D
UK	1110	E
KU	1111	F

Der Ausgangsstring HUMUHUMUNUKUNUKUAPUAA wurde in den aus 17 Zeichen mit je vier Bit bestehenden Ergebnisstring 1636794626DF05600 konvertiert, der somit 68 Bit umfasst. Die sieben Zeichen des Quell-Alphabets kann man mit einem Block-Code von 3 Bit Länge darstellen, so dass die 21 Zeichen des zu komprimierenden Wortes nur 63 Bit umfassen. So gesehen ergibt sich keine Kompressionsleistung. Legt man jedoch als Alphabet die ASCII-Zeichen mit 8 Bit Länge zu Grunde und wählt man für die Code-Tabelle 9 Bit, so ergäbe sich immerhin eine Kompression von $21 \cdot 8 = 168$ Bit auf $17 \cdot 9 = 153$ Bit.
Anzumerken ist noch, dass bei der Dekompression keine weiteren Informationen erforderlich sind, wenn man sich zuvor auf ein bestimmtes Alphabet verständigt hat.

A 3.14 (L2) Lässt sich der String ANTANANARIVO (das ist die Hauptstadt von Madagaskar) oder PAPAYAPALMEN (die wachsen in Madagaskar, sind aber eigentlich keine Palmen) durch den LZW-Algorithmus effizienter codieren? Bitte begründen Sie Ihre Antwort.

Lösung

Die Strings ANTANANARIVO und PAPAYAPALMEN sind beide 12 Zeichen lang und sie bestehen beide aus 7 verschiedenen Buchstaben. Es sei n die Anzahl der pro Zeichen in der Code-Tabelle verwendeten Bits. Teilt man die Strings in Zeichengruppen auf, für deren Codierung jeweils n Bits erforderlich sind, so erhält man:

A N T AN ANA R I V O	Länge: $9n$ Bit
P A PA Y AP A L M E N	Länge $10n$ Bit

Die Kompression ist also für ANTANANARIVO um 10% effizienter als für PAPAYAPALMEN.

A 3.15 (M1) Es soll ein aus 2^{16} Zeichen bestehender Text komprimiert werden, dessen Quellalphabet aus 40 mit gleicher Häufigkeit auftretenden Zeichen besteht. Weiter sei nichts über den Text und das Alphabet bekannt.

a) Berechnen Sie die Entropie dieses Alphabets.

b) Wie viele Bits pro Zeichen benötigt man mindestens für eine Codierung mit einem binären Blockcode?

c) Ist das folgende Kompressionsverfahren für dieses Beispiel sinnvoll? Man kann eine Anzahl n von Codewörtern mit 5 Bit und die verbleibenden $40 - n$ Codewörter mit 6 Bit codieren. Dabei muss n so

bestimmt werden, dass die mittlere Wortlänge möglichst klein wird.

d) Wie viele Byte umfasst dann der komprimierte Text? Welchem Kompressionsfaktor entspricht dies im Vergleich mit dem Blockcode aus a)?

Lösung

a) Für die Entropie gilt mit $p_i = 1/40$:

$$H = \sum_{i=1}^{40} p_i \operatorname{ld} \frac{1}{p_i} = 40 \cdot \frac{1}{40} \operatorname{ld} 40 = 5{,}3219 \text{ Bit/Zeichen.}$$

b) Wegen $H = 5{,}3219$ Bit/Zeichen benötigt man für einen Block-Code (Code mit konstanter Wortlänge) mindestens 6 Bit pro Zeichen.

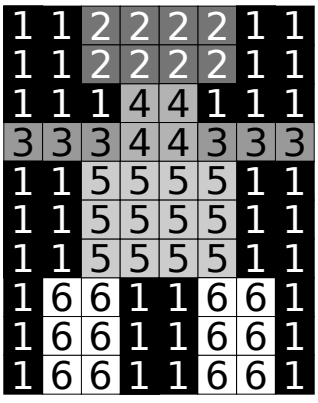
c) Man berechnet für dieses Verfahren:

$$(5n + (40-n)\cdot 6)/40 \geq 5{,}3219 \quad \Rightarrow n \leq 240 - 5{,}3219 \cdot 40 = 27{,}124 \quad \Rightarrow n = 27.$$

Allerdings ergibt sich dafür das Problem, dass die Fano-Bedingung nicht notwendigerweise erfüllt ist. Man könnte zur Erfüllung der Fano-Bedingung beispielsweise alle 6-Bit-Worte mit 0 beginnen und alle 5-Bit-Worte mit 1. Man hat also Codewörter der Art 0xxxxx und 1xxxx. Dies ergibt 32 Wörter mit 6 Bit und 16 Wörter mit 5 Bit. Man wird also als Optimum $n = 16$ wählen, also 16 Wörter mit 5 Bit und 24 Wörter mit 6 Bit codieren.

d) Die mittlere Wortlänge ist $L = (16 \cdot 5 + 24 \cdot 6)/40 = 5{,}6$.
Die Anzahl der Bytes des komprimierten Textes ist also $5{,}6 \cdot 2^{16}/8 = 0.7 \cdot 2^{16}$.
Die Anzahl der Bytes des unkomprimierten Textes ist $6 \cdot 2^{16}/8 = 0{,}75 \cdot 2^{16}$.
Der Kompressionsfaktor beträgt $k = 5{,}6/6 = 0{,}93\overline{3}$.

A 3.16 (L3) Gegeben sei das unten stehende, aus den Grauwerten 1, 2, 3, 4, 5 und 6 bestehende Bild.



a) Bestimmen Sie die Auftrittswahrscheinlichkeiten der Grauwerte.

b) Geben Sie für die Grauwerte einen Binärcode mit minimaler konstanter Wortlänge an und berechnen Sie die Größe (in Bit) des so codierten Bildes.

c) Berechnen Sie die Entropie des gegebenen Bildes.

d) Konstruieren Sie unter Verwendung des Huffman-Verfahrens einen optimalen Code mit variabler Wortlänge. Berechnen Sie dabei auch die mittlere Wortlänge und die Redundanz. Wie viel Bit umfasst nun das Bild?

e) Konstruieren Sie nun einen möglichst effizienten Lauflängen-Code. Wie viele Bit umfasst das Bild, wenn man es mit diesem Lauflängen-Code codiert?

f) Wie groß ist das Bild, wenn man es, so weit möglich, mit einem Quadtree codiert?

g) Codieren Sie nun das Bild mit dem LZW-Verfahren. Wie groß ist das Bild jetzt?

Lösung

a) Die Auftrittshäufigkeiten und die daraus berechneten Auftrittswahrscheinlichkeiten sind in der folgenden Tabelle eingetragen.

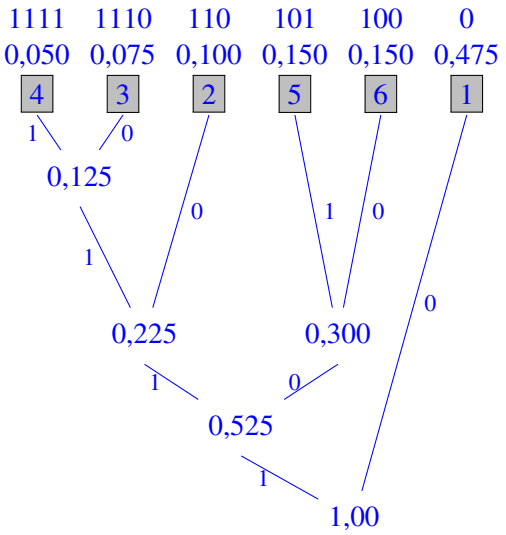
b) Bei sechs mit einem Code konstanter Wortlänge zu codierenden Worten ist die minimale Wortlänge 3. In der Tabelle ist ein entsprechender Code als $\text{Code}_{\text{Block}}$ eingetragen. Code_{H} ist der Huffman-Code aus Aufgabe (d).

Als Bildgröße ergibt sich damit $80 \cdot 3 = 240$ Bit.

x_i	Anzahl	p_i	$I(x_i)$	$\text{Code}_{\text{Block}}$	Code_{H}
1	38	0,475	1,0740	001	0
2	8	0,100	3,3219	010	110
3	6	0,075	3,7370	011	1110
4	4	0,050	4,3219	100	1111
5	12	0,150	2,7370	101	101
6	12	0,150	2,7370	110	100

c) Entropie: $H \approx 2{,}1598$ Bit/Zeichen.

d) Huffman-Baum:



Mittlere Wortlänge: $L = 2{,}1750$ Bit/Zeichen.
Redundanz: $R = 2{,}1750 - 2{,}1598 = 0{,}0152$ Bit/Zeichen.

Das Bild hat damit die Länge $38 \cdot 1 + 8 \cdot 3 + 6 \cdot 4 + 4 \cdot 4 + 12 \cdot 3 + 12 \cdot 3 = 174$ Bit.

e) Für die Lauflängen-Codierung kann man beispielsweise folgende Zuordnung wählen:
$1 = 01, 2 = 10, 3 = 11, 4 = 00$.

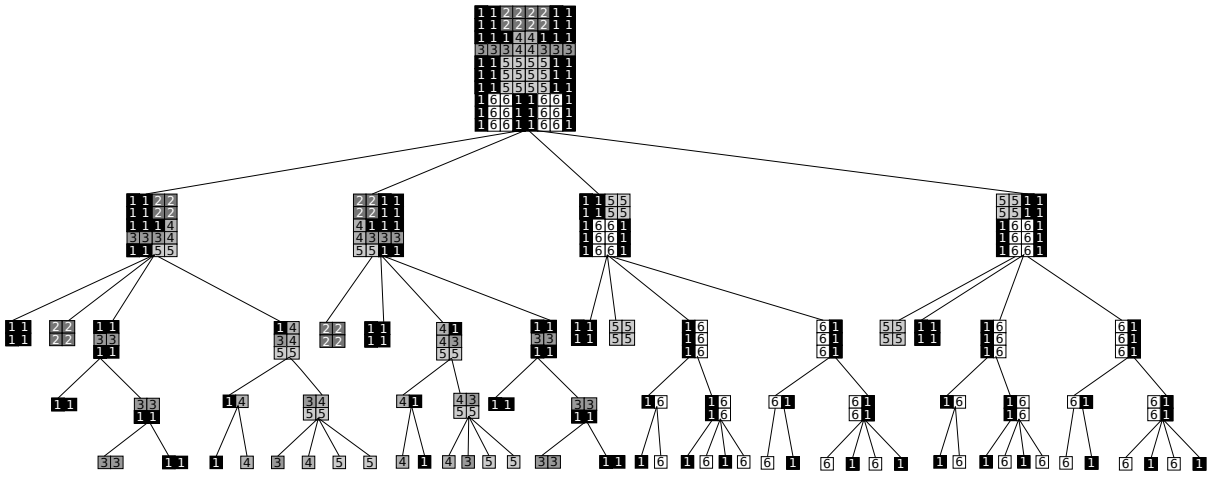
Bildgröße für Lauflängen-Codierung und Block-Code: 180 Bit.

1 1 2 2 2 2 1 1	10001 00010 10001
1 1 2 2 2 2 1 1	10001 00010 10001
1 1 1 4 4 1 1 1	11001 10100 11001
3 3 3 4 4 3 3 3	11011 10100 11011
1 1 5 5 5 5 1 1	10001 00101 10001
1 1 5 5 5 5 1 1	10001 00101 10001
1 1 5 5 5 5 1 1	10001 00101 10001
1 6 6 1 1 6 6 1	01001 10110 10001 10110 01001
1 6 6 1 1 6 6 1	01001 10110 10001 10110 01001
1 6 6 1 1 6 6 1	01001 10110 10001 10110 01001

Bildgröße für Lauflängen-Codierung und Huffman-Code: 142 Bit.

1 1 2 2 2 2 1 1	100 00110 100
1 1 2 2 2 2 1 1	100 00110 100
1 1 1 4 4 1 1 1	110 101111 110
3 3 3 4 4 3 3 3	111110 101111 111110
1 1 5 5 5 5 1 1	100 00101 100
1 1 5 5 5 5 1 1	100 00101 100
1 1 5 5 5 5 1 1	100 00101 100
1 6 6 1 1 6 6 1	010 10110 100 10110 010
1 6 6 1 1 6 6 1	010 10110 100 10110 010
1 6 6 1 1 6 6 1	010 10110 100 10110 010

f) Ein Quadtree könnte wie folgt aussehen:



Benötigt werden 41 Bit für die Baumstruktur, gefolgt von 50 Grauwerten (Blätter). Diese benötigen bei Verwendung des Block-Codes 150 Bit, also insgesamt 191 Bit. Verwendet man den Huffman-Code an Stelle des Block-Codes, dann reduziert sich der Platzbedarf der Grauwerte auf $22 \cdot 1 + 20 \cdot 3 + 8 \cdot 4 = 114$ Bit, insgesamt also 155 Bit.

g) Dargestellt ist im Folgenden eine LZW-Kompression mit 4 Bit pro Codewort. Die Codetabelle wird dabei jeweils neu initialisiert, wenn sie voll ist. Der benötigte Speicherplatzbedarf ist damit 184 Bit. Ohne Neuinitialisierung wären es 224 Bit.

Zeichenkette	Ausgabe-Code (binär)	Ausgabe-Code (hex)
1	0000	0
2	0001	1
3	0010	2
4	0011	3
5	0100	4
6	0101	5
–	0110	6
–	0111	7
–	1000	8
–	1001	9
–	1010	A
–	1011	B
–	1100	C
–	1101	D
–	1110	E
–	1111	F

Der Codierungsvorgang läuft damit folgendermaßen ab:

Schritt	Zeichen c	Präfix P	Eintrag in Code-Tabelle	Ausgabe
0	–	–	Vorbesetzung	–
1	1	1	-	-
2	1	1	11	1 =0
3	2	2	12	1 =0
4	2	2	22	2 =1
5	2	22	-	-
6	2	2	222	22 =8
7	1	1	21	2 =1
8	1	11	-	-
9	1	1	111	11 =6
10	1	11	-	-
11	2	2	112	11 =6
12	2	22	-	-
13	2	222	-	-
14	2	2	2222	222 =9
15	1	21	-	-
16	1	1	211	21 =10
17	1	11	-	-
18	1	111	-	-
19	1	1	1111	111 =11

Codetabelle voll:

Zeichenkette	Ausgabe-Code (hex)
1	0
2	1
3	2
4	3
5	4
6	5
11	6
12	7
22	8
222	9
21	A
111	B
112	C
2222	D
211	E
1111	F

Schritt	Zeichen c	Präfix P	Eintrag in Code-Tabelle	Ausgabe
20	4	4	14	1 =0
21	4	4	44	4 =3
22	1	1	41	4 =3
23	1	1	11	1 =0
24	1	11	-	-
25	3	3	113	11 =8
26	3	3	33	3 =2
27	3	33	-	-
28	4	4	334	33 =10
29	4	44	-	-
30	3	3	443	44 =6
31	3	33	-	-
32	3	3	333	33 =10
33	1	1	31	3 =2
34	1	11	-	-
35	5	5	115	11 =8

Codetabelle voll:

Zeichenkette	Ausgabe-Code (hex)
1	0
2	1
3	2
4	3
5	4
6	5
44	6
41	7
11	8
113	9
33	A
334	B
443	C
333	D
31	E
115	F

Schritt	Zeichen *c*	Präfix *P*	Eintrag in Code-Tabelle	Ausgabe
36	5	5	55	5 =4
37	5	5	55	5 =4
38	5	55	-	-
39	1	1	551	55 =6
40	1	1	11	1 =0
41	1	11	-	-
42	1	1	111	11 =8
43	5	5	15	1 =0
44	5	55	-	-
45	5	5	555	55 =6
46	5	55	-	-
47	1	551	-	-
48	1	1	5511	551 =7
49	1	11	-	-
50	1	111	-	-
51	5	5	1115	111 =9
52	5	55	-	-
53	5	555	-	-
54	5	5	5555	555 =11
55	1	1	51	5 =4
56	1	11	-	-
57	1	111	-	-

Codetabelle voll:

Zeichenkette	Ausgabe-Code (hex)
1	0
2	1
3	2
4	3
5	4
6	5
55	6
551	7
11	8
111	9
15	A
555	B
5511	C
1115	D
5555	E
51	F

Schritt	Zeichen c	Präfix P	Eintrag in Code-Tabelle	Ausgabe
58	6	6	1116	111 =9
59	6	6	66	6 =5
60	1	1	61	6 =5
61	1	1	11	1 =0
62	6	6	16	1 =0
63	6	66	-	-
64	1	1	661	66 =6
65	1	11	-	-
66	6	6	116	11 =8
67	6	66	-	-
68	1	661	-	-
69	1	1	6611	661 =10
70	6	16	-	-
71	6	6	166	16 =9
72	1	61	-	-
73	1	1	611	61 =7
74	6	16	-	-
75	6	166	-	-
76	1	1	1661	166 =13
77	1	11	-	-
78	6	116	-	-

Codetabelle voll:

Zeichenkette	Ausgabe-Code (hex)
1	0
2	1
3	2
4	3
5	4
6	5
66	6
61	7
11	8
16	9
661	A
116	B
6611	C
166	D
611	E
1661	F

Schritt	Zeichen *c*	Präfix *P*	Eintrag in Code-Tabelle	Ausgabe
79	6	6	1166	116 =11
80	1	1	61	6 =5
81	-	-	-	1 =0

Ende. Codetabelle:

Zeichenkette	Ausgabe-Code (hex)
1	0
2	1
3	2
4	3
5	4
6	5
61	6
–	7
–	8
–	9
–	A
–	B
–	C
–	D
–	E
–	F

A 3.17 (P3) Schreiben Sie C-Programme für folgende Datenkompressions-Methoden:

- Differenzcodierung mit vier Bit langen Blockcodes und zwei Codewörtern pro Byte.
- LZW-Datenkompression.

Lösung

Der Source-Code wird separat zur Verfügung gestellt.

Übungsaufgaben zu Kapitel 3.3

A 3.18 (T0) Wie berechnet man die Hamming-Distanz? Was ist die Hamming-Distanz eines Codes?

Lösung

Hamming-Distanz: Anzahl der Stellen, in denen sich zwei gleich lange Codewörter unterscheiden.
Hamming-Distanz eines Codes: Die minimale in einem Block-Code auftretende Hamming-Distanz.

A 3.19 (L1) Bestimmen Sie die Hamming-Distanzen der folgenden Codes:
a) {110101, 101011, 010011, 101100} b) {2B, 4A, 78, A9} (Binärcode, hexadezimal).

Lösung

a)

	110101	101011	010011	101100
110101	-	-	-	-
101011	4	-	-	-
010011	3	3	-	-
101100	3	3	6	-

Hamming-Distanz des Codes = minimale Hamming-Distanz der Codewörter: $h = 3$.

b) Es handelt sich um einen Binärcode, der zur kompakteren Notation hexadezimal angegeben wurde:

	2B	4A	78	A9
	00101011	01001010	01111000	10101001
00101011	-	-	-	-
01001010	3	-	-	-
01111000	4	3	-	-
10101001	2	5	4	-

Hamming-Distanz des Codes = minimale Hamming-Distanz der Codewörter: $h = 2$.

A 3.20 (L2) Bestimmen Sie die Hamming-Distanz für den Code {1101011, 1010110, 0000011, 0001100} und modifizieren Sie diesen Code dann durch Änderung eines einzigen Bit so, dass sich eine um eins erhöhte Hamming-Distanz ergibt.

Lösung

	1101011	1010110	0000011	0001100
1101011	-	-	-	-
1010110	5	-	-	-
0000011	3	4	-	-
0001100	5	4	4	-

Die Hamming-Distanz des Codes ist also $h = 3$. Modifikation:

	1101**1**11	1010110	0000011	0001100
1101**1**11	-	-	-	-
1010110	4	-	-	-
0000011	4	4	-	-
0001100	4	4	4	-

Weitere Lösungen: 1101011 → 1111011, 1101011 → 1101001, 1101011 → 1101010.

A 3.21 (L2) Bei einer seriellen Datenübermittlung werden mit 7 Bit codierte ASCII-Zeichen mit einem zusätzlichen Paritätsbit und einem Längsprüfwort nach jeweils 8 Zeichen gesendet. Es gilt gerade Parität. Im Sender wird folgende Nachricht empfangen:

MSB	1 0 1 1 1 1 1 1	0
	0 1 1 1 1 1 1 1	1
	0 1 0 0 0 0 0 1	0
Datenbits	0 0 0 1 0 1 0 0	0 Prüfspalte
	1 0 1 0 0 0 1 0	1
	1 1 0 0 1 0 0 1	0
LSB	0 0 1 1 0 1 1 0	0
Paritätsbits	1 0 0 0 1 0 0 0	0

Wie lautet die empfangene Nachricht? Sind Übertragungsfehler aufgetreten? Wenn ja, wie lautet die korrekte Nachricht? Bestimmen Sie die durch die Paritätsbits bedingte zusätzliche Redundanz.

Lösung

Die empfangene Nachricht lautet: F2eibier.

Es ist offenbar ein Fehler an der markierten Stelle aufgetreten. Durch Invertieren des entsprechenden Bits erhält man die korrekte Nachricht: Freibier.

MSB	1 **0** 1 1 1 1 1 1	0 ⇐
	0 1 1 1 1 1 1 1	1
	0 1 0 0 0 0 0 1	0
Datenbits	0 0 0 1 0 1 0 0	0 Prüfspalte
	1 0 1 0 0 0 1 0	1
	1 1 0 0 1 0 0 1	0
LSB	0 0 1 1 0 1 1 0	0
Paritätsbits	1 0 0 0 1 0 0 0	0
	⇑	

Zusätzliche Redundanz:

$$R = \frac{k+s+1}{k} = \frac{8+7+1}{8} = 2 \text{ [Bit/Wort]}.$$

A 3.22 (T0) Was ist ein Gray-Code? Geben Sie einen Gray-Code für die direkt binär codierten Zahlen von 0 bis 16 an.

Lösung

in Gray-Code ist ein Ziffern-Code, bei dem benachbarte Codewörter auch benachbarte Ziffern codieren. Im Idealfall führt ein Ein-Bit-Fehler daher entweder zu einem Fehlerwort oder zu einem benachbarten Codewort, das dann einer zu der ursprünglich codierten Ziffer benachbarten Ziffer entspricht. Der dadurch bedingte Fehler ist in diesem Sinne minimal.

Gray-Code für die Zahlen von 0 bis 16:

Dezimal	Gray
0	00000
1	00001
2	00011
3	00010
4	00110
5	00111
6	00101
7	00100
8	01100
9	01101
10	01111
11	01110
12	01010
13	01011
14	01001
15	01000
16	11000

A 3.23 (L3) Finden Sie einen Binärcode mit Wortlänge 4 für die Ziffern 0 bis 9, wobei folgende Bedingung einzuhalten ist: Wenn bei der Übertragung für eine Ziffer x ein 1-Bit Fehler auftritt, so soll entweder ein Fehlerwort entstehen oder für die zu dem wegen des Fehlers entstehenden Codewort gehörende Ziffer y soll gelten: $|x-y| \leq 2$. Kann man unter Einhaltung dieser Bedingung noch mehr Ziffern codieren?

Lösung

Die Ziffern 0 bis 9 werden in das folgende Diagramm eingetragen. Das Zeichen „-“ markiert ein nicht verwendetes Codewwort, also ein Fehlerwort. Z. B. erhält die Ziffer 4 das Codewort 0111.

	00	01	11	10
00	0	1	2	-
01	-	3	4	5
11	8	-	6	7
10	-	-	-	9

Durch einige Umstellungen gelingt es, auf diese Weise auch die Ziffern 0 bis 10 zu codieren:

	00	01	11	10
00	0	2	3	-
01	-	4	5	6
11	9	-	7	8
10	-	1	-	10

A 3.24 (M2) Wie viele verschiedene Codewörter kann ein 2-aus-6-Code und wie viele ein 1-aus-15-Code maximal enthalten? Geben Sie für die beiden Codes die Hamming-Distanz und die Redundanzen an. Dabei kann für alle Codewörter dieselbe Auftrittswahrscheinlichkeit angenommen werden.

Lösung

Die Anzahl n der Codewörter ergibt sich als Anzahl der möglichen Kombinationen mit erlaubten Wiederholungen:

$$n = \binom{6}{2} = 15 \quad \text{für den 2-aus-6-Code,}$$
$$n = \binom{15}{1} = 15 \quad \text{für den 1-aus-15-Code.}$$

Die minimale Stellendistanz und damit auch die Hamming-Distanz ist in beiden Fällen (wie bei allen m-aus-n-Codes) $h = 2$. Man erkennt dies auch daran, dass man von einem Codewort zu einem unmittelbar benachbarten Codewort kommt, wenn man eine 1 in eine 0 ändert und an einer anderen Stelle eine 0 in eine 1, entsprechend also einer Stellendistanz von 2.
Der 2-aus-6-Code hat die konstante Wortlänge $L = 6$. Da alle Codewörter dieselbe Auftrittswahrscheinlichkeit p besitzen sollen, ist diese $p = 1/15$. Für die Entropie erhält man daraus:

$$H = \sum_{i=1}^{10} p \operatorname{ld} \frac{1}{p} = 15 \cdot \frac{1}{15} \operatorname{ld} 15 = \operatorname{ld} 15 \approx 3{,}9069 \quad .$$

Die Redundanz ist also: $R = L - H = 6 - 3{,}9069 = 2{,}0931$ Bit/Zeichen.
Für den 1-aus-15-Code ist $L = 15$ und die Entropie ebenfalls $H = \operatorname{ld} 15 \approx 3{,}9069$ Bit/Zeichen, da 15 Zeichen mit identischen Auftrittswahrscheinlichkeiten codiert werden sollen.
Somit ist die Redundanz: $R = L - H = 15 - 3{,}9069 = 11{,}0931$ Bit/Zeichen.

A 3.25 (M1) Zeigen Sie, dass 40182735 ein korrekt gebildeter 8-stelliger EAN-Code ist.

Lösung

Der 8-stellige EAN-Code hat die Form $a_1a_2a_3a_4a_5a_6a_7p$, wobei die Prüfziffer p so gebildet wird, dass die Summe $3 \cdot a_1 + a_2 + 3 \cdot a_3 + a_4 + 3 \cdot a_5 + a_6 + 3 \cdot a_7$ durch Addition der Prüfziffer p auf eine durch 10 teilbare Zahl ergänzt wird. Hier erhält man die Summe $3 \cdot 4 + 0 + 3 \cdot 1 + 8 + 3 \cdot 2 + 7 + 3 \cdot 3 = 45$. Addition der Prüfziffer 5 ergibt 50, also tatsächlich eine durch 10 teilbare Zahl. Die Zahl 40182735 ist daher ein korrekt gebildeter EAN-Code.

A 3.26 (M1) Bestimmen Sie die Prüfziffer p in der ISBN-10-Nummer 0-521-43108-p.

Lösung

Zur Ermittlung der Prüfziffer p berechnet man zunächst:

$$10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9 \quad .$$

Hier:

$$10 \cdot 0 + 9 \cdot 5 + 8 \cdot 2 + 7 \cdot 1 + 6 \cdot 4 + 5 \cdot 3 + 4 \cdot 1 + 3 \cdot 0 + 2 \cdot 8 = 127 \quad .$$

Anschließend bestimmt man p so, dass die obige Summe durch Addition von p auf eine ohne Rest durch 11 teilbare Zahl ergänzt wird und fügt p als letzte Stelle an die ISBN an. Da hierbei $0 \leq p \leq 10$ gilt, kann auch die zweistellige Prüfziffer 10 auftreten; diese wird dann durch das Einzelzeichen X ersetzt. Für eine korrekte ISBN-10 gilt also:

$$(10a_1 + 9a_2 + 8a_3 + 7a_4 + 6a_5 + 5a_6 + 4a_7 + 3a_8 + 2a_9 + p) \bmod 11 = 0 \quad .$$

Hier muss also $p = 5$ gewählt werden.

A 3.27 (M1) Gegeben sei ein Hamming-Code mit der nebenstehenden Kontrollmatrix $\boldsymbol{M}$. Geben Sie für die beiden empfangenen Worte 1011010 und 1101011 an, ob es sich um ein Codewort oder ein Fehlerwort handelt. Im Falle eines Fehlerwortes: Wie lautet unter der Annahme eines 1-Bit Fehlers das korrekte Wort? Extrahieren Sie auch die Informationsbits aus den empfangenen Nachrichten.

$$\boldsymbol{M} = \begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \end{pmatrix}$$

Lösung

Man interpretiert das empfangene Wort als Zeilenvektor und multipliziert es mit der Kontrollmatrix. Die Rechnung erfolgt modulo 2. Die Informationsbits ergeben sich aus der Bit-Reihenfolge $\begin{pmatrix} p_1 & p_2 & i_1 & p_3 & i_2 & i_3 & i_4 \end{pmatrix}$. Für 1011010 rechnet man:

$$\boldsymbol{x} \cdot \boldsymbol{M} = \begin{pmatrix} 1\,0\,1\,1\,0\,1\,0 \end{pmatrix} \begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \end{pmatrix} = \begin{pmatrix} 0\,0\,0 \end{pmatrix} \quad . \tag{3.1}$$

Da das Ergebnis der Nullvektor ist, handelt sich um ein korrektes Codewort. Die Information lautet: 1010.

Für 1101011 rechnet man:

$$\boldsymbol{x} \cdot \boldsymbol{M} = \begin{pmatrix} 1\,1\,0\,1\,0\,1\,1 \end{pmatrix} \begin{pmatrix} 0\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \end{pmatrix} = \begin{pmatrix} 1\,1\,0 \end{pmatrix} \quad . \tag{3.2}$$

Am Ergebnis liest man ab, dass das Bit an Position 6 fehlerhaft ist. Die Zählung der Positionen beginnt von links mit Position 1. Das korrekte Wort lautet: 1101001. Die Information lautet: 0001.

A 3.28 (M2) Geben Sie allgemein an, wie viele verschiedene Codewörter ein Code mit s Stellen und vorgegebener Hamming-Distanz h maximal umfassen kann. Wie viele Codewörter kann ein Code mit $s = 6$ Stellen und Hamming-Distanz $h = 5$ maximal umfassen?

Lösung

Eine obere Grenze für die Anzahl von Codewörtern, die unter dieser Bedingung gebildet werden können ergibt sich unmittelbar aus der Bedingung:
maximale Anzahl der Codewörter $\leq$ Gesamtvolumen / Volumen einer Kugel mit Radius e.
Als Maß für das Volumen ist hier die Anzahl der enthaltenen Vektoren zu verstehen. Der gesamte lineare Raum B^s hat daher das Volumen 2^s, da er bei gegebener Dimension s, d. h. gegebener Stellenzahl der Codewörter, gerade 2^s Vektoren umfasst. Das Volumen einer Kugel mit Radius 1 umfasst den Mittelpunkt der Kugel sowie alle über eine Kante erreichbaren nächsten Nachbarn, es beträgt also $V_1 = 1 + s$. Als obere Grenze für die Anzahl n_1 der Codewörter, für welche die Korrigierbarkeit von Einzelfehlern gefordert wird,

ergibt sich also:

$$n_1 \leq \frac{2^s}{V_1} = \frac{2^s}{1+s} \quad .$$

Im allgemeinen Fall von e pro Codewort korrigierbaren Fehlern ist in (28) an Stelle von V_1 das Volumen V_e einer Kugel mit Radius e einzusetzen. Durch Abzählen aller Punkte, die von dem betrachteten Punkt aus über maximal e Kanten erreichbar sind, erhält man:

$$V_e = 1 + \sum_{i=1}^{e} \binom{s}{i} \quad .$$

Die obere Grenze für die Anzahl n_e der s-stelligen Codewörter eines Codes, für welche die Korrigierbarkeit von e Fehlern möglich ist, lautet damit:

$$n_e \leq \frac{2^s}{V_e} = \frac{2^s}{1+\sum_{i=1}^{e} \binom{s}{i}} \quad .$$

Dieses Ergebnis bedeutet nur, dass n_e eine Obergrenze für die Anzahl der Codewörter ist. Es wird jedoch nichts darüber ausgesagt, ob diese Anzahl tatsächlich erreichbar ist und wie derartige Codes konstruiert werden können.
Für $s = 6$ und $h = 5$ folgt zunächst $e = (h-1)/2 = 2$ und damit:

$$n_2 \leq \frac{2^6}{1+\binom{6}{1}+\binom{6}{2}} = \frac{2^6}{1+6+15} = \frac{64}{22} \approx 2{,}91 \quad .$$

4 Lösungen zu Kapitel 4 – Verschlüsselung

A 4.1 (M1) Warum ist bei der Verschlüsselung mit multiplikativen Schlüsseln bei einem Alphabet mit $n = 26$ Zeichen der multiplikative Schlüssel $k = 6$ nicht verwendbar?

Lösung

$k = 6$ ist nicht verwendbar, da die Gleichung $x^{-1} \cdot 6 \bmod 26 = 1$ nicht erfüllbar ist und 6 daher keine modulare Inverse hat. Feststellbar ist dies, weil gilt $\text{ggT}(26,6) = 2 \neq 1$. Die Verwendung von $k = 6$ hätte daher zur Folge, dass die Verschlüsselung nicht eindeutig wäre.

A 4.2 (M2) Bob möchte Alice die Nachricht „ferien“ senden. Die beiden haben die Verwendung der 26 lateinischen Großbuchstaben vereinbart, A entspricht also 0, B entspricht 1 usw. Sie benutzen zur Verschlüsselung den multiplikativen Schlüssel 5 und den additiven Schlüssel 8. Wie lautet der Geheimtext, den Bob sendet? Für die Entschlüsselung benötigt Alice die modulare Inverse des multiplikativen Schlüssels 5. Berechnen Sie diese und entschlüsseln Sie die von Alice empfangene Nachricht.

Lösung

Zur Verschlüsselung eines Zeichens x dient folgende Formel:

$$y = (k \cdot x + d) \bmod n$$

mit $k = 5$, $d = 8$ und $n = 26$.
Im Folgenden werden alle Operationen modulo 26 durchgeführt. Bob rechnet also:

Klartext:	f	e	r	i	e	n
numerische Darstellung:	5	4	17	8	4	13
Multipl. mit $k = 5$:	25	20	7	14	20	13
Add. von $d = 8$:	7	2	15	22	2	21
Ergebnis:	H	C	P	W	C	V

Für die Entschlüsselung muss Alice rechnen:

$$x = (y - d) \cdot k^{-1} \bmod n.$$

Dabei ist k^{-1} die modulare Inverse von k. Diese lässt sich z. B. aus dem Satz von Euler bestimmen:

$$k^{-1} = k^{\Phi(n)-1} \bmod n \quad .$$

Mit $\Phi(26) = \Phi(2)\Phi(13) = 1 \cdot 12 = 12$ erhält man $k^{-1} = 5^{11} \bmod 26 = 21$. Zur Prüfung des Ergebnisses rechnet man $5 \cdot 21 \bmod 26 = 1$.
Alice rechnet somit:

Nachricht:	H	C	P	W	C	V
numerische Darstellung:	7	2	15	22	2	21
Subtraktion von $d = 8$:	25	20	7	14	20	13
Multipl. mit $k^{-1} = 21$:	5	4	17	8	4	13
Ergebnis:	f	e	r	i	e	n

A 4.3 (M2) Alice sendet Bob die verschlüsselte Nachricht CHJF. Die beiden haben die Verwendung der 26 lateinischen Buchstaben mit den Positionen im Alphabet als numerische Codierung (also A=0 bis Z=25) vereinbart und sie benutzen zur Verschlüsselung den multiplikativen Schlüssel $k = 3$ und den additiven Schlüssel $s = 5$. Wie bestimmt Alice aus der empfangenen verschlüsselten Nachricht den Klartext und wie lautet dieser?
Cleo hat die Nachricht CHJF abgefangen und außerdem Bobs Abfalleimer durchwühlt. Dort hat sie einen Papierschnipsel mit dem Text ha gefunden, in welchem sie den zu CH gehörenden Klartext vermutet. Berechnen Sie daraus (unter der Annahme, dass ein entsprechendes Verschlüsselungsverfahren verwendet wurde) den additiven Schlüssel s und den multiplikativen Schlüssel k.

Lösung

Für die Entschlüsselung muss Alice rechnen:

$$x = (y - s) \cdot k^{-1} \bmod n.$$

mit $k = 3$, $s = 5$ und $n = 26$. Dabei ist k^{-1} die modulare Inverse von k. Diese lässt sich z. B. aus dem Satz von Euler bestimmen:

$$k^{-1} = k^{\Phi(n)-1} \bmod n \quad .$$

Mit $\Phi(26) = \Phi(2)\Phi(13) = 1 \cdot 12 = 12$ erhält man $k^{-1} = 3^{11} \bmod 26 = 9$. Zur Prüfung des Ergebnisses rechnet man $3 \cdot 9 \bmod 26 = 1$.
Alice rechnet somit:

Nachricht:	C	H	J	F
numerische Darstellung:	2	7	9	5
Subtraktion von $s = 5$:	23	2	4	0
Multipl. mit $k^{-1} = 9$:	25	18	10	0
Ergebnis:	z	s	k	a

HINWEIS: Für die folgende Rechnung ist leider die Angabe in Auflage 5 falsch. Es muss heißen: „Cleo hat die Nachricht CHJF abgefangen und außerdem Bobs Abfalleimer durchwühlt. Dort hat sie einen Papierschnipsel mit dem Text **zs** gefunden …“.
Cleo vermutet, dass der auf dem Papierschnipsel gefundene Text zs zu den verschlüsselten Zeichen CH gehört. Daraus berechnet sie den additiven Schlüssel s und den multiplikativen Schlüssel k wie folgt:
Die numerischen Darstellungen der Zeichen gemäß ihrer Position im Alphabet lauten:
$C : y_1 = 2, H : y_2 = 7, z : x_1 = 25, s : x_2 = 18$.
Mit $y_i = x_i k + s$ ergeben sich daraus die beiden Gleichungen für k und s (mod 26):

$$(1) \quad 2 = 25 \cdot k + s$$
$$(2) \quad 7 = 18 \cdot k + s.$$

Aus $(2) - (1)$ folgt:

$$5 = -7k \quad \rightarrow \quad 5 = 19k \quad \rightarrow k = 5 \cdot 19^{-1} = 5 \cdot 11 = 3 \quad (\bmod 26).$$

Einsetzen von $k = 3$ in (1) liefert: $2 = 75 + s \rightarrow s = -73 \rightarrow s = 5$.
Alternativ ergibt sich aus (2): $7 = 54 + s \rightarrow s = -47 \rightarrow s = 5$.

A 4.4 (M1) Bob sendet die folgende verschlüsselte Nachricht an seine Freundin Alice:
HVOBVXVEXCYVHRUXYSTIJLZOJXVLZAX. Cleo fängt die Nachricht ab. Sie weiß, dass Bob und Alice einen Vigenère-Code verwenden. Als ersten Schritt der Entschlüsselung muss Cleo daher die Schlüssellänge bestimmen. Ermitteln Sie die wahrscheinlichste Schlüssellänge für die verschlüsselte Nachricht.

Lösung

Cleo wendet den Kasiski-Test an, d. h. sie bestimmt die die Primfaktoren der Abstände der häufigsten Zeichen. Die am häufigsten auftretenden Primfaktoren bzw. deren häufigste Produkte sind dann Kandidaten für die Schlüssellänge.
Die häufigsten Buchstaben sind V und X, die beide 5 Mal auftreten.
Die Abstände zwischen den Vs sind:

$$3,5,10 = 2\cdot 5, 25 = 5\cdot 5, \quad 2,7,22 = 2\cdot 11, \quad 5,20 = 2\cdot 2\cdot 5, \quad 15 = 3\cdot 5.$$

Die Abstände zwischen den Xen sind:

$$3,10 = 2\cdot 5, 20 = 2\cdot 2\cdot 5, 25 = 5\cdot 5, \quad 7,17,22 = 2\cdot 11, \quad 10 = 2\cdot 5, 15 = 3\cdot 5, \quad 5.$$

Die Häufigkeiten der Abstände lauten also: 14 mal 5, 10 mal 2, 4 mal 3. Die wahrscheinlichste Schlüssellänge ist also 5.

A 4.5 (L1) Der Text MARIA wurde unter Verwendung des 7-Bit ASCII-Codes mit einem One-Time-Pad verschlüsselt. Das Ergebnis ist JOSEF. Wie lautet der One-Time-Pad?

Lösung

Die Verschlüsselung erfolgt durch bitweise XOR-Verknüpfung des im ASCII-Code dargestellten Klartextes mit einer nur einmal verwendeten Bit-Maske $m_1m_2m_3m_4m_5$, die ebenso lang ist wie der Klartext (One-Time-Pad). Die Wahrheitstabelle der XOR-Verknüpfung lautet:

$$\begin{aligned} 1\,\text{XOR}\,1 &= 0,\\ 1\,\text{XOR}\,0 &= 1,\\ 0\,\text{XOR}\,1 &= 1,\\ 0\,\text{XOR}\,0 &= 0 \end{aligned}$$

Die XOR-Verknüpfung ist involutorisch, es gilt also: $a\,\text{XOR}\,m = b$ und $b\,\text{XOR}\,m = a$. Für dieses Beispiel rechnet man dementsprechend:

$$\begin{aligned} M = J\,\text{XOR}\,m_1 \rightarrow 1001101 &= 1001010\,\text{XOR}\,m_1 \rightarrow m_1 = 0000101\\ A = O\,\text{XOR}\,m_2 \rightarrow 1000001 &= 1001111\,\text{XOR}\,m_2 \rightarrow m_2 = 0001110\\ R = S\,\text{XOR}\,m_3 \rightarrow 1010010 &= 1010011\,\text{XOR}\,m_3 \rightarrow m_3 = 0000001\\ I = E\,\text{XOR}\,m_4 \rightarrow 1001001 &= 1000101\,\text{XOR}\,m_4 \rightarrow m_4 = 0001100\\ A = F\,\text{XOR}\,m_5 \rightarrow 1000001 &= 1000110\,\text{XOR}\,m_5 \rightarrow m_5 = 0000111 \end{aligned}$$

Das One-Time-Pad lautet also: 0000101000111000000010001100000111.

A 4.6 (T1) Was sind Falltürfunktionen und Einwegfunktion?

Lösung

Einwegfunktionen sind injektive Funktionen $f : X \rightarrow Y$, für die $y = f(x)$ für alle $x \in X$ effizient berechenbar ist, für die aber $x = f^{-1}(y)$ aus der Kenntnis von y nicht effizient (also nur mit exponentieller Komplexität) berechenbar ist.
Falltürfunktionen sind spezielle Einwegfunktionen, bei denen ebenfalls $y = f(x)$ effizient berechenbar ist, aber auch $x = f^{-1}(y)$, wenn eine Zusatzinformation (ein Schlüssel) bekannt ist.

A 4.7 (M2) Berechnen Sie $\mathrm{ggT}(16269,693)$ und $31^{4123} \bmod 40251$.

Lösung

$\mathrm{ggT}(16269,693)$:
$\mathrm{ggT}(x,y) = \mathrm{ggT}(y, x \bmod y)$ für $y > 0$, $\mathrm{ggT}(x,0) = x$.
$\mathrm{ggT}(16269,693) = \mathrm{ggT}(693,330) = \mathrm{ggT}(330,33) = \mathrm{ggT}(33,0) = 33$.

Berechnung von $31^{4123} \bmod 40251$:
Es gilt: $a \cdot b \bmod n = (a \bmod n)(b \bmod n) \bmod n$
oder für $a = b$: $a^2 \bmod n = (a \bmod n)(a \bmod n) \bmod n$.
Daraus folgt durch wiederholte Anwendung der Regel für beliebige Potenzen:
$a^m \bmod n = (a^k \bmod n)^j \cdot (a^{m-kj} \bmod n) \bmod n$.
Geschickt ist eine Zerlegung des Exponenten in Zweierpotenzen und Klammerung nach Hornerschema:

$$4123 = 4096 + 27 = 4096 + 16 + 8 + 2 + 1,$$

also

$$\begin{aligned} 31^{4123} &= 31^{4096} \cdot 31^{16} \cdot 31^8 \cdot 31^2 \cdot 31^1 \quad , \\ &= (31^{2048} \cdot 31^8 \cdot 31^4 \cdot 31)^2 \cdot 31 \\ &= ((31^{512} \cdot 31^2 \cdot 31)^{2^2} \cdot 31)^2 \cdot 31 \\ &= (((31^{256} \cdot 31)^2 \cdot 31)^{2^2} \cdot 31)^2 \cdot 31 \\ &= ((((31^{2^2 2^2 2^2 2^2} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \end{aligned}$$

Die Klammern werden nun von innen nach außen ausgewertet. Dabei wird nach jeder Operation sofort das Ergebnis modulo 40251 reduziert. Auf diese Weise entstehen niemals Zahlen, die größer als das Quadrat des

Moduls 40251 sind:

$$
\begin{aligned}
&= ((((31^{2^{2^{2^{2^{2^2}}}}} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((37999^{2^{2^{2^{2^2}}}} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((40129^{2^{2^{2^2}}} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((14884^{2^{2^2}} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((32203^{2^2} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((6445^{2^2} \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((39244^2 \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((((7774 \cdot 31)^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= (((39739^2 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= (((20638 \cdot 31)^2)^2 \cdot 31)^2 \cdot 31 \\
&= ((36013^2)^2 \cdot 31)^2 \cdot 31 \\
&= (8698^2 \cdot 31)^2 \cdot 31 \\
&= (23575 \cdot 31)^2 \cdot 31 \\
&= 6307^2 \cdot 31 \\
&= 10261 \cdot 31 \\
&= 36334 \pmod{40251}
\end{aligned}
$$

A 4.8 (M2) Beim RSA-Verfahren wird im Schlüsselverzeichnis eine Zahl n veröffentlicht, die das Produkt zweier großer Primzahlen p und q ist, sowie für jeden Teilnehmer ein öffentlicher Schlüssel c, für den $\mathrm{ggT}(c, \Phi(n)) = 1$ gilt. Jeder Teilnehmer erhält ferner als geheimen Schlüssel d die modulare Inverse von c bezüglich $\Phi(n)$. Zur Verschlüsselung einer Nachricht x dient die Funktion $y = x^c \bmod n$ und zur Entschlüsselung die Funktion $x = y^d \bmod n$.

a) Es sei $p = 3$ und $q = 11$. Ermitteln Sie die kleinste für Bob als öffentlicher Schlüssel c in Frage kommende Zahl. Berechnen Sie nun Bobs geheimen Schlüssel d.

b) Alice möchte an Bob die Nachricht „ei“ senden. Berechnen Sie die buchstabenweise verschlüsselte Nachricht y. Dabei wird als numerische Codierung der Buchstaben deren Position im Alphabet, beginnend mit a=1 verwendet.

c) Bob hat die Nachricht 26, 3 empfangen und möchte sie entschlüsseln. Wie rechnet er?

Lösung

a) $n = 3 \cdot 11 = 33$,
$\Phi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20 = 2 \cdot 2 \cdot 5$.
Für c muss gelten:

$1 < c < \Phi(n)$, $\text{ggT}(c, \Phi(n)) = 1$.
Daraus folgt, dass $c = 3$ der kleinste mögliche öffentliche Schlüssel ist.

Man ermittelt nun aus der Bedingung $c \cdot d \bmod \Phi(n) = 1$ den geheimen Schlüssel d als die modulare Inverse zu c, z. B. aus dem Satz von Euler:

$$d = c^{-1} = c^{\Phi(\Phi(n))-1} \bmod \Phi(n) \quad .$$

also

$$d = 3^{-1} = 3^{\Phi(20)-1} \bmod 20 \quad .$$

Es gilt $\Phi(20) = \Phi(2 \cdot 2 \cdot 5) = 2 \cdot 4 = 8$ und damit

$$d = 3^7 \bmod 20 = 7 \quad .$$

b) e entspricht $x_e = 5$, i entspricht $x_i = 9$. Alice entnimmt dem öffentlichen Schlüsselverzeichnis Bobs öffentlichen Schlüssel (n, c) und rechnet:
$y_e = x_e^c \bmod n = 5^3 \bmod 33 = 26$,
$y_i = x_i^c \bmod n = 9^3 \bmod 33 = 3$.

c) Bob empfängt die Nachricht 26, 3 und rechnet mit seinem geheimen Schlüssel $d = 7$:
$x_e = y_e^d \bmod n = 26^7 \bmod 33 = 5$, entsprechend „c“,
$x_i = y_i^d \bmod n = 3^7 \bmod 33 = 9$, entsprechend „i“.

5 Lösungen zu Kapitel 5 – Computerhardware und Maschinensprache

Übungsaufgaben zu Kapitel 5.2

A 5.1 (T1) Beantworten Sie die folgenden Fragen:

a) Was wird durch die Aussagenlogik beschrieben?

b) Was ist eine logische Verknüpfung?

c) Geben Sie die Absorptionsgesetze an.

d) Was ist ein boolescher Verband?

e) Was ist ein Venn-Diagramm?

f) Was ist eine binäre Schaltfunktion?

g) Beschreiben Sie das boolesche Normalformtheorem.

h) Was sind Maxterme und Minterme?

i) Was sind benachbarte Terme?

Lösung

a) In der Aussagenlogik studiert man als Wahrheitsfunktionen bezeichnete Verknüpfungen von Aussagen durch logische Operatoren, deren Ergebnisse wiederum Aussagen sind. Unter Aussagen versteht man in der Aussagenlogik Elemente $a, b, c, \ldots$ einer Menge, wobei diese Elemente – neben anderen, in diesem Zusammenhang nicht relevanten Eigenschaften – einen Wahrheitswert besitzen, der nur die beiden Zustände „wahr“ oder „falsch“ annehmen kann.

b) Zur Verknüpfung Aussagen werden die folgenden logischen Grundverknüpfungen ver- wendet:

Verknüpfung	Name	gebräuchliche Schreibweisen
nicht a	Negation	$\neg a, \overline{a}$
a und b	Konjunktion	$a \wedge b, a\&b, a \cdot b, ab$
a oder b	Disjunktion	$a \vee b, a + b$
wenn a dann b	Implikation	$a \rightarrow b, a \Rightarrow b$
a genau dann wenn b	Äquivalenz	$a \leftrightarrow b, a \Leftrightarrow b$

Der Wahrheitswert des Ergebnisses einer Wahrheitsfunktion hängt nur von den Wahrheitswerten der Argumente der Wahrheitsfunktion ab. Da es nur zwei Wahrheitswerte gibt, kann man sämtliche Wahrheitsfunktionen durch endliche Tabellen eindeutig definieren. Für die oben genannten Verknüpfungen lauten die zugehörigen Tabellen:

Zweistellige Verknüpfungen

a	b	$a \vee b$	$a \wedge b$	$a \Rightarrow b$	$a \Leftrightarrow b$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	1

Einstellige Verknüpfung

a	$\neg a$
0	1
1	0

Da Aussagen nur die beiden Wahrheitswerte „wahr“ bzw. 1 und „falsch“ bzw. 0 annehmen können, kann es genau $2^2 = 4$ einstellige und $2^4 = 16$ zweistellige logische Verknüpfungen geben. Man kann zeigen dass man alle diese Verknüpfungen durch oben bereits definierten Grundverknüpfungen Konjunktion, Disjunktion und Negation ausgedrückt kann.

c) Die Absorptionsgesetze lauten: $a \wedge (a \vee b) = a$ und $a \vee (a \wedge b) = a$

d) Zur Definition eines booleschen Verbands siehe Buch, Kap. 5.2.2, S. 195.

e) Venn-Diagramme dienen in der Mengenlehre zur anschaulichen, grafischen Darstellung der Verknüpfung von Mengen. Es gibt eine strukturelle Übereinstimmung (Isomorphie) der Mengenlehre mit der Aussagenlogik: „oder“ entspricht „Vereinigung“, „und“ entspricht „Durchschnitt“ und „nicht“ entspricht „Komplementbildung“. Deswegen hat man auch die an die Symbole der Mengenoperationen erinnernde Schreibweise und für die logischen Verknüpfungen „oder“ und „und“ eingeführt. Wegen dieser Isomorphie lassen sich auch logische Verknüpfungen durch Venn-Diagramme anschaulich darstellen, wie im Buch in Abb. 5.10 auf S. 196 gezeigt.

f) siehe Buch Kap. 5.2.2, S. 196, Abschnitt „Schaltfunktionen“.

g) Das boolesche Normalform-Theorem liefert eine einfache Möglichkeit, aus der Wahrheitstabelle einer Schaltfunktion die Schaltfunktion selbst zu konstruieren. Zur Herleitung siehe Buch, Kap. 5.2.3, S. 197.

h) Minterme und Maxterme sind Bestandteile der disjunktiven bzw. konjunktiven Normalform. Details hierzu siehe Buch, Kap. 5.2.3, S. 197.

i) Zur Beschreibung benachbarter Terme siehe Buch, Kap. 5.2.4, S. 198.

A 5.2 (L1) Stellen Sie die zweistelligen logischen Verknüpfungen Implikation, NOR, NAND, Äquivalenz und XOR unter ausschließlicher Verwendung von Konjunktion, Disjunktion und Negation dar.

Lösung

NOR: a NOR $b = \neg(a \vee b)$
NAND: a NAND $b = \neg(a \wedge b)$
Äquivalenz: $a \Leftrightarrow b = (a \wedge b) \vee (\neg a \vee \neg b)$
XOR: a XOR $b = \neg(a \Leftrightarrow b) = (\neg a \vee \neg b) \wedge (a \vee b)$

A 5.3 (L2) Vereinfachen Sie den folgende booleschen Ausdruck so weit wie möglich:
$(a \wedge b \wedge d) \vee (a \wedge \overline{b} \wedge c) \vee (a \wedge b \wedge \overline{d})$.

Lösung

$$\begin{aligned}(a \wedge b \wedge d) \vee (a \wedge \overline{b} \wedge c) \vee (a \wedge b \wedge \overline{d}) &= (a \wedge b) \vee (a \wedge \overline{b} \wedge c) \\ &= a \wedge (b \vee \overline{b} \wedge c) \\ &= a \wedge (b \vee c) \\ &= a \wedge b \vee a \wedge c\end{aligned}$$

A 5.4 (L2) Vereinfachen Sie die folgende Schaltfunktion so weit wie möglich und geben Sie das Ergebnis unter ausschließlicher Verwendung von NAND an:
$f(x_1,x_2,x_3) = (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3)$.

Lösung

Terme, die sich nur in einer Variablen unterscheide, werden zusammengefasst:

$$\begin{aligned} f(x_1,x_2,x_3) &= (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \\ &= (x_1 \wedge x_3) \vee (x_1 \wedge \bar{x}_2) \vee (\bar{x}_1 \wedge \bar{x}_2) \\ &= (x_1 \wedge x_3) \vee \bar{x}_2 \\ &= \overline{\overline{(x_1 \wedge x_3) \vee \bar{x}_2}} \\ &= \overline{\overline{(x_1 \wedge x_3)} \wedge x_2} \\ &= \mathrm{NAND}(\mathrm{NAND}(x_1,x_3),x_2) \end{aligned}$$

A 5.5 (L3) Geben Sie die konjunktive und die disjunktive Normalform der folgenden Schaltfunktion an:
$f(x_1,x_2,x_3) = (x_1 \wedge x_2)\,\mathrm{XOR}\,(x_1 \wedge x_3)\,\mathrm{XOR}\,(x_2 \wedge x_3)\,\mathrm{XOR}\,\overline{(x_1 \wedge x_3)}$.

Lösung

Für XOR gilt: $a\,\mathrm{XOR}\,b = (\bar{a} \wedge b) \vee (a \wedge \bar{b})$.

$$\begin{aligned} f(x_1,x_2,x_3) &= (x_1 \wedge x_2)\,\mathrm{XOR}\,(x_1 \wedge x_3)\,\mathrm{XOR}\,(x_2 \wedge x_3)\,\mathrm{XOR}\,\overline{(x_1 \wedge x_3)} \\ &= (x_1 \wedge x_2)\,\mathrm{XOR}\,(x_2 \wedge x_3)\,\mathrm{XOR}\,1, \quad \text{da } (x_1 \wedge x_3)\,\mathrm{XOR}\,\overline{(x_1 \wedge x_3)} = 1 \\ &= (\overline{(x_1 \wedge x_2)} \wedge (x_2 \wedge x_3) \vee (x_1 \wedge x_2) \wedge \overline{(x_2 \wedge x_3)})\,\mathrm{XOR}\,1 \\ &= ((\bar{x}_1 \vee \bar{x}_2) \wedge (x_2 \wedge x_3) \vee (x_1 \wedge x_2) \wedge (\bar{x}_2 \vee \bar{x}_3))\,\mathrm{XOR}\,1 \\ &= ((\bar{x}_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_2 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \bar{x}_2) \vee (x_1 \wedge x_2 \wedge \bar{x}_3))\,\mathrm{XOR}\,1 \\ &= ((\bar{x}_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \bar{x}_3))\,\mathrm{XOR}\,1 \\ &= (\overline{(\bar{x}_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \bar{x}_3)} \wedge 1) \vee (((\bar{x}_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \bar{x}_3)) \wedge 0) \\ &= (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \quad \text{konjunktive Normalform} \\ &= (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3) \vee (\bar{x}_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) \\ &\quad \text{disjunktive Normalform} \end{aligned}$$

A 5.6 (L3) Es seien f und g zwei n-stellige Schaltfunktionen. Dann sind auch logische Verknüpfungen wie $f \wedge g$ etc. ebenfalls Schaltfunktionen. Zeigen Sie:

a) $f \vee (\neg f \wedge g) = f \vee g$,

b) $f \vee (f \wedge g) = f$,

c) $f = g$ gilt genau dann, wenn $(\neg f \wedge g) \vee (f \wedge \neg g) = 0$ gilt.

Lösung

a) $f \vee (\neg f \wedge g) = (f \vee \neg f) \wedge (f \vee g) = f \vee g$

b) $f \vee (f \wedge g) = (f \wedge 1) \vee (f \wedge g) = f \wedge (1 \vee g) = f \wedge 1 = f$

c) $f = g \Leftrightarrow (\neg f \wedge g) \vee (f \wedge \neg g) = 0$ ist zu zeigen.
Man nimmt zunächst an, $f = g$ sei richtig und beweist, dass daraus $(\neg f \wedge g) \vee (f \wedge \neg g) = 0$ folgt. Man beweist also die Richtung $\Rightarrow$:
$(\neg g \wedge g) \vee (g \wedge \neg g) = 0 \vee 0 = 0$.
Nun nimmt man an, $(\neg f \wedge g) \vee (f \wedge \neg g) = 0$ sei richtig und beweist, dass daraus $f = g$ folgt. Man beweist also die Richtung $\Leftarrow$:
1) $0 = \neg g \wedge 0 = \neg g \wedge ((\neg f \wedge g) \vee (f \wedge \neg g)) = (\neg g \wedge \neg f \wedge g) \vee (\neg g \wedge f \wedge \neg g) = 0 \vee (\neg g \wedge f \wedge \neg g) = f \wedge \neg g$.
2) $1 = \neg 0 = \neg((\neg f \wedge g) \vee (f \wedge \neg g)) = (f \vee \neg g) \wedge (\neg f \vee g) = ((f \vee \neg g) \wedge \neg f) \vee ((f \vee \neg g) \wedge g) = ((f \wedge \neg f) \vee (\neg g \wedge \neg f)) \vee ((f \wedge g) \vee (\neg g \wedge g)) = (\neg g \wedge \neg f) \vee (f \wedge g)$.
3) $g = g \wedge 1 = g \wedge ((\neg g \wedge \neg f) \vee (f \wedge g)) = (g \wedge \neg g \wedge \neg f) \vee (g \wedge f \wedge g) = 0 \vee (g \wedge f \wedge g) = g \wedge f$.
4) $g = g \vee 0 = (g \wedge f) \vee 0 = (g \wedge f) \vee (f \wedge \neg g) = f \wedge (g \vee \neg g) = f$.

A 5.7 (L3) Eine Schaltfunktion y mit drei Eingängen x_1, x_2, x_3 sei durch folgende Funktionstabelle gegeben:

x_1	0 0 0 0 1 1 1 1
x_2	0 0 1 1 0 0 1 1
x_3	0 1 0 1 0 1 0 1
$y = f(x_1, x_2, x_3)$	0 1 1 0 0 1 1 1

Geben Sie die Schaltfunktion in disjunktiver Normalform an, erstellen Sie das zugehörige KV-Diagramm und vereinfachen Sie die Funktion so weit wie möglich.

Lösung

Das zugehörige allgemeine Karnaugh-Veitch-Diagramm für drei Variablen lautet:

	x_2		$\bar{x}_2$	
x_1	$x_1x_2\bar{x}_3$	$x_1x_2x_3$	$x_1\bar{x}_2x_3$	$x_1\bar{x}_2\bar{x}_3$
$\bar{x}_1$	$\bar{x}_1x_2\bar{x}_3$	$\bar{x}_1x_2x_3$	$\bar{x}_1\bar{x}_2x_3$	$\bar{x}_1\bar{x}_2\bar{x}_3$
	$\bar{x}_3$	x_3		$\bar{x}_3$

$=$

	x_2		$\bar{x}_2$	
x_1	m_6	m_7	m_5	m_4
$\bar{x}_1$	m_2	m_3	m_1	m_0
	$\bar{x}_3$	x_3		$\bar{x}_3$

In die den Mintermen entsprechenden Felder trägt man nun je nach dem gegebenen Wert $y = 0$ oder $y = 1$ der Schaltfunktion eine 0 oder eine 1 ein. Man erhält:

	x_2		$\bar{x}_2$	
x_1	1	1	1	0
$\bar{x}_1$	1	0	1	0
	$\bar{x}_3$	x_3		$\bar{x}_3$

Aus den 5 Feldern mit den Einträgen folgt die disjunktive Normalform mit den Mintermen:

$$y = m_6 \vee m_7 \vee m_5 \vee m_2 \vee m_1 = x_1x_2\bar{x}_3 \vee x_1x_2x_3 \vee x_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3 \quad .$$

Aus dem KV-Diagramm ist ersichtlich, dass beispielsweise m_2 und m_6 sowie m_1 und m_5 benachbart sind und daher zusammen gefasst werden können. Dies ergibt die Lösung:

$$y = x_2\bar{x}_3 \vee \bar{x}_2x_3 \vee x_1x_2x_3 \quad .$$

Alternativ dazu könnte man die Schaltfunktion auch durch die den Einträgen 0 entsprechende konjunktive Normalform mit den Maxtermen darstellen:

$$y = M_0 \wedge M_3 \vee M_4 = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \quad .$$

Übungsaufgaben zu Kapitel 5.3

A 5.8 (T1) Beantworten Sie die folgenden Fragen:

a) Erläutern Sie den Unterschied zwischen Schaltnetzen und Schaltwerken.

b) Was versteht man unter einer kombinatorischen Schaltung?

c) Beschreiben Sie die Aufgabe eines Dekodierers.

d) Was ist eine Pseudo-Tetrade?

e) Grenzen Sie die Begriffe Halb-, Voll- und Serienaddierer gegeneinander ab.

f) Wozu werden Flip-Flops in erster Linie eingesetzt?

g) Nennen Sie vier unterschiedliche Flip-Flop-Typen und charakterisieren Sie diese.

Lösung

a) Schaltnetz: technische Realisierung einer Schaltfunktion; siehe Buch Kap. 5.3, S. 201.

Schaltwerk: Schaltnetz mit Verzögerungs- und Rückkopplungsgliedern; siehe Buch Kap. 5.3, S. 206.

b) kombinatorische Schaltung: Zustand der Ausgänge hängt ausschließlich vom Zustand der Eingänge ab. Die Abhängigkeit der Ausgangssignale von den Eingangssignalen wird dabei in der Regel durch eine Schaltfunktion definiert, die wiederum als Wahrheitstabelle angegeben wird.

c) Aufgabe eines Dekodierers: Transformation einer Anzahl von Eingangssignalen in die Ausgangssignale nach einer vorgegebenen Regel; siehe z. B. im Buch Abb. 5.15, S. 203.

d) Pseudo-Tetrade: In manchen Fällen spielen nicht alle Kombinationen der Eingangsvariablen für die Schaltfunktion eine Rolle. Ein Beispiel dafür ist die Ansteuerung der sieben Elemente einer 7-Segmentanzeige zur Darstellung der Ziffern 0 bis 9 (siehe Buch Abb. 5.16, S. 203). Die 7 Segmente a bis g können durch Decodierung von vier Eingängen x_0 bis x_3 an- und ausgeschaltet werden. Dazu wird eine negative Logik verwendet, d. h. „0“ bedeutet „Segment an“ und „1“ bedeutet „Segment aus“. Da nur die 10 Ziffern 0 bis 9 anzuzeigen sind, mit vier Eingängen aber 16 Kombinationen existieren, spielen offenbar sechs prinzipiell mögliche Kombinationen keine Rolle. Man bezeichnet diese als Pseudo-Tetraden.

e) Halbaddierer: dient zur Addition von zwei binären Stellen a und b. Das Ergebnis wird in zwei Ausgängen angegeben, nämlich der Summe s und dem Übertrag (Carry) c.

Volladdierer: Aus zwei Halbaddierern lässt sich ein Volladdierer aufbauen. Ein Volladdierer hat drei Eingänge: a, b und c, wobei c der Übertrag aus der Addition der vorhergehenden binären Stelle ist. Wie der Halbaddierer, so hat auch der Volladdierer zwei Ausgänge, nämlich die Summe S und den Übertrag C.

Serienaddierer: Volladdierer, bei dem der Übertrag über ein Verzögerungsglied in den Addierer zurückgekoppelt wird.

f) Einsatz von Flip-Flops in erster Linie als schneller Speicher in CPU-Registern oder statischem Speicher.

g) Flip-Flop-Typen: RS-FF, JK-FF, T-FF, D-FF; siehe Buch Kap. 5.3.3, S. 190.

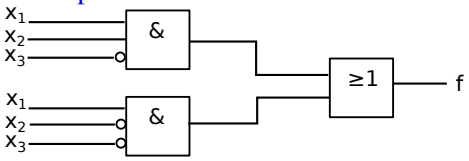
A 5.9 (L1) Erstellen Sie eine Wertetabelle und einen Schaltplan mit möglichst wenig Gattern für die folgende Schaltfunktion: $f(x_1,x_2,x_3) = (x_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3)$.

Lösung

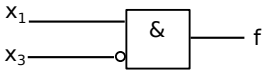
$f(x_1,x_2,x_3) = (x_1 \wedge x_2 \wedge \bar{x}_3) \vee (x_1 \wedge \bar{x}_2 \wedge \bar{x}_3) = x_1 \wedge \bar{x}_3$.
Die Vereinfachung ergibt, dass die Schaltfunktion lediglich die Verknüpfung $x_1 \wedge \bar{x}_3$ darstellt.

x_1	0 0 0 0 1 1 1 1
x_2	0 0 1 1 0 0 1 1
x_3	0 1 0 1 0 1 0 1
$y = f(x_1,x_2,x_3)$	0 0 0 0 1 0 1 0

Schaltpläne:



Direkte Übertragung der Schaltfunktion



reduzierte Form

A 5.10 (L3) Gesucht ist eine Schaltung derart, dass eine Lampe von drei verschiedenen Schaltern ein- bzw. ausgeschaltet werden kann (eine sog. Kreuzschaltung). Erstellen Sie eine Wertetabelle, finden und minimieren Sie eine Schaltfunktion und geben Sie eine Schaltung an, die ausschließlich NOR-Gatter verwendet. Betrachten Sie dabei drei Varianten:

a) Die Lampe soll genau dann brennen, wenn genau ein Schalter geschlossen ist.

b) Die Lampe soll genau dann brennen, wenn mindestens ein Schalter geschlossen ist.

c) Die Lampe soll genau dann nicht brennen, wenn genau ein Schalter geöffnet ist.

Lösung

a) Bei der gesuchten Schaltung muss die Lampe eingeschaltet sein, wenn entweder Schalter s_1 oder s_2 oder s_3 eingeschaltet ist, also den logischen Wert 1 annimmt. Die Bedingung $f(0,0,0) = 0$ bedeutet, dass die Lampe ausgeschaltet ist, wenn alle drei Schalter ausgeschaltet sind. Die Wertetabelle, die Schaltfunktion und die Schaltung lauten damit:

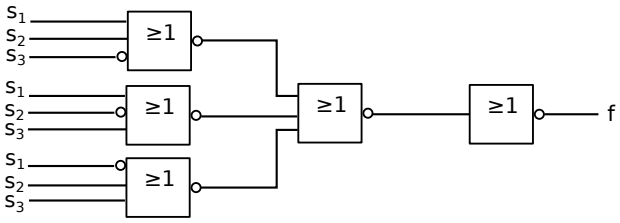
s_1	0 0 0 0 1 1 1 1
s_2	0 0 1 1 0 0 1 1
s_3	0 1 0 1 0 1 0 1
$y = f(s_1,s_2,s_3)$	0 1 1 0 1 0 0 0

Daraus ergibt sich (3 Minterme):

$$f(s_1,s_2,s_3) = (\bar{s}_1 \wedge \bar{s}_2 \wedge s_3) \vee (\bar{s}_1 \wedge s_2 \wedge \bar{s}_3) \vee (s_1 \wedge \bar{s}_2 \wedge \bar{s}_3).$$

Umwandlung in NOR-Gatter:

$$\begin{aligned}(\overline{s}_1 \wedge \overline{s}_2 \wedge s_3) \vee (\overline{s}_1 \wedge s_2 \wedge \overline{s}_3) \vee (s_1 \wedge \overline{s}_2 \wedge \overline{s}_3) &= \overline{\overline{(\overline{s}_1 \wedge \overline{s}_2 \wedge s_3)} \vee \overline{(\overline{s}_1 \wedge s_2 \wedge \overline{s}_3)} \vee \overline{(s_1 \wedge \overline{s}_2 \wedge \overline{s}_3)}} \\ &= \overline{(s_1 \vee s_2 \vee \overline{s}_3)} \vee \overline{(s_1 \vee \overline{s}_2 \vee s_3)} \vee \overline{(\overline{s}_1 \vee s_2 \vee s_3)} \\ &= \overline{\overline{(s_1 \vee s_2 \vee \overline{s}_3)} \vee \overline{(s_1 \vee \overline{s}_2 \vee s_3)} \vee \overline{(\overline{s}_1 \vee s_2 \vee s_3)}}\end{aligned}$$



b) Bei der gesuchten Schaltung muss die Lampe eingeschaltet sein, wenn mindestens einer der drei Schalter geschlossen ist. Das Ergebnis lautet damit:

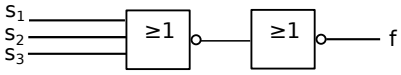
s_1	0 0 0 0 1 1 1 1
s_2	0 0 1 1 0 0 1 1
s_3	0 1 0 1 0 1 0 1
$y = f(s_1, s_2, s_3)$	0 1 1 1 1 1 1 1

Daraus ergibt sich (1 Maxterm):

$$f(s_1, s_2, s_3) = s_1 \vee s_2 \vee s_3.$$

Umwandlung in NOR-Gatter:

$$f(s_1, s_2, s_3) = s_1 \vee s_2 \vee s_3 = \overline{\overline{s_1 \vee s_2 \vee s_3}}.$$



c) Wertetabelle:

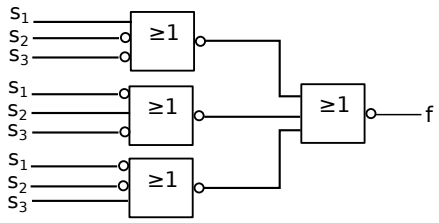
s_1	0 0 0 0 1 1 1 1
s_2	0 0 1 1 0 0 1 1
s_3	0 1 0 1 0 1 0 1
$y = f(s_1, s_2, s_3)$	1 1 1 0 1 0 0 1

Daraus ergibt sich (3 Maxterme):

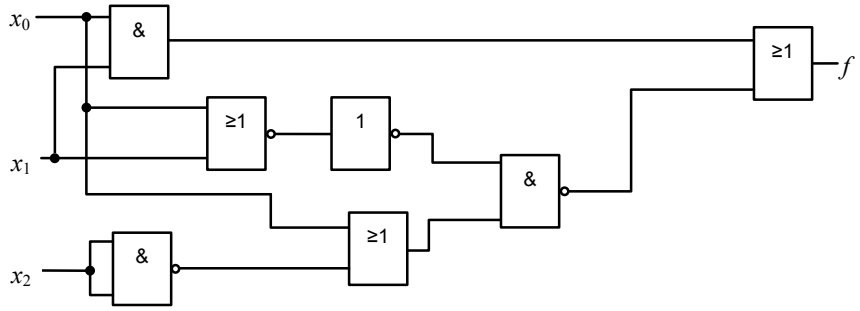
$$f(s_1, s_2, s_3) = (s_1 \vee \overline{s}_2 \vee \overline{s}_3) \wedge (\overline{s}_1 \vee s_2 \vee \overline{s}_3) \wedge (\overline{s}_1 \vee \overline{s}_2 \vee s_3).$$

Umwandlung in NOR-Gatter:

$$\begin{aligned}(s_1 \vee \overline{s}_2 \vee \overline{s}_3) \wedge (\overline{s}_1 \vee s_2 \vee \overline{s}_3) \wedge (\overline{s}_1 \vee \overline{s}_2 \vee s_3) &= \overline{\overline{(s_1 \vee \overline{s}_2 \vee \overline{s}_3) \wedge (\overline{s}_1 \vee s_2 \vee \overline{s}_3) \wedge (\overline{s}_1 \vee \overline{s}_2 \vee s_3)}} \\ &= \overline{\overline{(s_1 \vee \overline{s}_2 \vee \overline{s}_3)} \vee \overline{(\overline{s}_1 \vee s_2 \vee \overline{s}_3)} \vee \overline{(\overline{s}_1 \vee \overline{s}_2 \vee s_3)}}\end{aligned}$$



A 5.11 (L2) Geben Sie für die nachfolgende Schaltung eine Schaltfunktion in konjunktiver Normalform an.



Lösung

$$
\begin{aligned}
f(x_0,x_1,x_2) &= (x_0 \wedge x_1) \vee \overline{\overline{\overline{(x_0 \vee x_1)} \wedge (x_0 \vee \bar{x}_2)}} \\
&= (x_0 \wedge x_1) \vee \overline{(x_0 \vee x_1) \wedge (x_0 \vee \bar{x}_2)} \\
&= (x_0 \wedge x_1) \vee (\bar{x}_0 \wedge \bar{x}_1) \vee (\bar{x}_0 \wedge x_2) \\
&= (x_0 \wedge x_1 \wedge x_2) \vee (x_0 \wedge x_1 \wedge \bar{x}_2) \vee (\bar{x}_0 \wedge \bar{x}_1 \wedge x_2) \vee (\bar{x}_0 \wedge \bar{x}_1 \wedge \bar{x}_2) \vee (\bar{x}_0 \wedge x_1 \wedge x_2) \vee (\bar{x}_0 \wedge \bar{x}_1 \wedge x_2) \\
&= m_7 \vee m_6 \vee m_1 \vee m_0 \vee m_3 \quad \text{Minterme, disjunktive Normalform} \\
&= \overline{m_5 \vee m_4 \vee m_2} \quad \text{verbleibende Minterme mit } f = 0 \\
&= \overline{(x_0 \wedge \bar{x}_1 \wedge x_2) \vee (x_0 \wedge \bar{x}_1 \wedge \bar{x}_2) \vee (\bar{x}_0 \wedge x_1 \wedge \bar{x}_2)} \\
&= M_5 \wedge M_4 \wedge M_2 \quad \text{Maxterme, konjunktive Normalform} \\
&= (\bar{x}_0 \vee x_1 \vee \bar{x}_2) \wedge (\bar{x}_0 \vee x_1 \vee x_2) \wedge (x_0 \vee \bar{x}_1 \vee x_2)
\end{aligned}
$$

A 5.12 (L3) Geben Sie ein Schaltnetz mit vier Eingängen und vier Ausgängen an, welches das Produkt aus zwei zweistelligen Dualzahlen liefert. In der Schaltung dürfen Halbaddierer verwendet werden.

Lösung

Es seien a und b zwei zweistellige Dualzahlen:

$$a = a_1 a_0, \quad b = b_1 b_0 \quad .$$

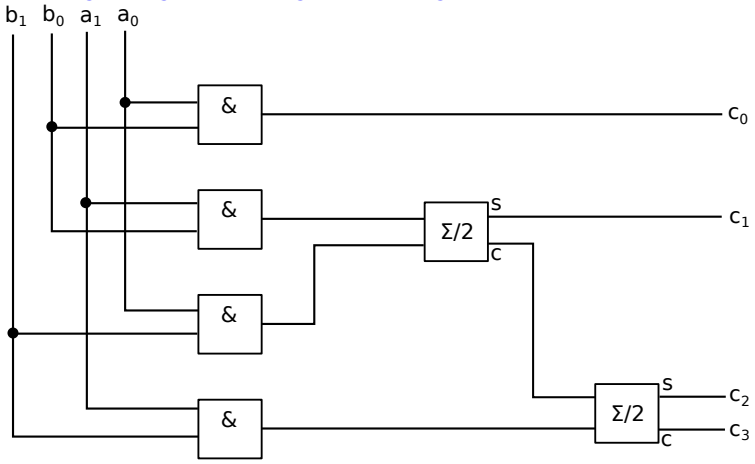
Für das Ergebnis der Multiplikation $c = a \cdot b$ Man erhält man:

$$c = c_3 c_2 c_1 c_0 = b_1 b_0 \cdot a_1 a_0 \quad ,$$

entsprechend:

$$\begin{array}{cccccc} & 0 & a_1 \wedge b_1 & a_0 \wedge b_1 & & 0 \\ + & 0 & 0 & a_1 \wedge b_0 & a_0 \wedge b_0 & \\ \hline & c_3 & c_2 & c_1 & c_0 & \end{array}$$

Die zugehörige Schaltung hat die folgende Form:



Übungsaufgaben zu Kapitel 5.5

A 5.13 (T1) Beantworten Sie die folgenden Fragen:

a) Beschreiben Sie kurz die Vor- und Nachteile von memory-mapped und isolated I/O.

b) Erläutern Sie die Wirkungsweise der Adressierungsart ARI und erklären Sie, warum gerade diese Adressierungsart in der Praxis besonders wichtig ist.

c) Erläutern Sie den Unterschied zwischen arithmetischer und logischer Verschiebung.

d) Grenzen Sie die Bedeutung des C-Flag und des X-Flag gegeneinander ab.

e) Welche Adressierungsarten werden vor allem bei Stack-Operationen angewendet.

Lösung

a) Beschreibung siehe Buch, Kap. 5.4.4, S. 209.
Memory-mapped I/O ist weniger komplex, weshalb die CPU einfacher (und billiger) aufgebaut werden kann. Es benötigt keine separaten Befehle/Register. Es gibt keinen Unterschied zu normalen Speicherzugriffen, alle Register und Adressierungsarten sind nutzbar.
Allerdings muss ein Teil des Adressbereichs für I/O reserviert werden, was den nutzbaren Speicherplatz reduziert. In heute üblichen 32 bzw. 64 Bit Systemen ist dies jedoch kein großes Problem.

b) Die Adresse eines Operanden ist bei der indirekten Adressierung (Address Register Indirect, ARI) in einem Adressregister abgelegt. Siehe hierzu Buch, Kap. 5.4.4, S. 228.
ARI ermöglicht relative Adressierung zu einer Basisadresse. Typische Beispiele sind Stack-Operation (Push, Pop).

c) Siehe hierzu Buch, Kap. 5.5.2, S. 233.

d) Siehe hierzu Buch, Kap. 5.4.2, S. 219.

e) ARI, insbesondere Adressregister indirekt mit Predekrement und Adressregister indirekt mit Postinkrement. Siehe hierzu Buch, Kap. 5.4.4, S. 228.

A 5.14 (L2) Geben Sie für die folgenden Assembler-Befehle die Adressierungsarten für Quelle und Ziel an: a) `MOVE D0,A0` b) `MOVE #$123,(A1)` c) `MOVE (A7)+,D3` d) `MOVE D1,$12345`

Lösung

Befehl	Adressierungsart für Quelle	Adressierungsart für Ziel
`MOVE D0,A0`	Datenregister direkt	Adressreg. direkt (implizit)
`MOVE #$123,(A1)`	Konstantenadr. (immediate)	Adressregister indirekt, ARI
`MOVE (A7)+,D3`	ARI mit Postinkrement (Stack)	Datenregister direkt
`MOVE D1,$12345`	Datenregister direkt	Absolut lang

A 5.15 (P3) Ein einfacher Mikroprozessor soll Befehle der Formate `OP adr` und `OP` verarbeiten können. Dabei soll der Befehlsteil `OP` jeweils durch ein Byte codiert sein, die Adressen `adr` sollen zwei Byte lang sein und die Speicherzellen sollen ein Byte speichern können. Es sind folgende Befehle möglich, wobei AC den Akkumulator bezeichnet:

OP	Befehl	Bedeutung
`A9`	`LDA a`	Lade ein Byte von Speicherzelle a in AC
`8D`	`STA a`	Speichere das in AC enthaltene Byte in Speicherzelle a
`C7`	`SUB a`	Subtrahiere das Byte in Speicherzelle a von dem Inhalt des AC
`C8`	`ADD a`	Addiere das Byte in Speicherzelle a zu dem Inhalt von AC
`76`	`HLT`	Programmende

a) Im Hauptspeicher sei ab Speicherzelle 0100_{hex} bis 0112_{hex} ein Programm gespeichert. Ferner sollen die Speicherzellen 0113_{hex} bis 0116_{hex} Daten enthalten. Der relevante Speicherausschnitt habe folgenden Inhalt:

0100_{hex}: `A9 01 14 C7 01 13 8D 01`
0108_{hex}: `13 A9 01 15 C7 01 13 8D`
0110_{hex}: `01 16 76 F0 FF 0F AB 01`

Der Befehlszähler soll nun mit 0100_{hex} initialisiert werden und das Programm soll ausgeführt werden. Geben sie für jeden Einzelschritt den Inhalt des Akkumulators an. Was steht nach Beendigung des Programms in den Speicherzellen 0113_{hex} bis 0116_{hex}?

b) Schreiben Sie unter Verwendung der oben angegebenen Befehle ein Programm, das die Inhalte der Speicherzellen 0113_{hex} und 0114_{hex} vertauscht.

Lösung

a) Ablauf:

Adresse	Befehl / Daten	Was passiert?	AC
0100	A9 0114	(0114) → AC	FF
0103	C7 0113	AC−(0114) → AC	0F = FF−F0
0106	8D 0113	AC → (0113)	0F
0109	A9 0115	(0115) → AC	0F
010C	C7 0113	AC−(0113) → AC	00 = 0F−0F
010F	8D 0116	AC → 0116	00
0112	76	HLT	

Speicherinhalt:

0113: 0F	0114: FF	0115: 0F	0116: 00

b) A9 0113
8D 0115
A9 0114
8D 0113
A9 0115
8D 0114
76

6 Lösungen zu Kapitel 6 – Rechnerarchitektur

A 6.1 (T1) Informieren Sie sich im Internet oder in einem Fachgeschäft, aus welchen Komponenten aktuelle Laptops oder PCs bestehen. Typischerweise finden Sie dort größere Tabellen mit dem CPU-Typ, dem RAM-Typ und anderen Informationen. Welche Mikroarchitektur hat die CPU? Von welchem Hersteller ist diese? Gibt es auch eine GPU, wenn ja welche?

Lösung

A 6.2 (T1) Vergleichen Sie die Ausstattung eines günstigen Desktop-Computers für das Büro mit einem Gaming-PC. Was macht einen Gaming-PC aus?

Lösung

Grafikkarte (GPU) Hochleistungs-Grafikkarten sind das Schlüsselmerkmal von Gaming-PCs. Sie bieten die Rechenleistung für grafikintensive Spiele und unterstützen hohe Auflösungen, hohe Frame-Raten sowie Grafikeffekte.

Prozessor (CPU) Gaming-PCs sind typischerweise mit Mehrkern-Prozessoren ausgestattet.

Kühlung Um die erhöhte Wärmeentwicklung insbesondere der Grafikkarte zu bewältigen, verfügen Gaming-PCs über umfangreichere Kühlungssysteme: beispielsweise mehrere Lüfter, spezielle Kühlkörper oder Wasserkühlungssysteme.

Speicher Gaming-PCs haben in der Regel einen großzügig bemessenen, schnellen RAM, ab 16 GB. Sie haben typischerweise SSDs als Sekundärspeicher, die über aktuelle Schnittstellen angebunden sind (M.2/PCIe, NVMe).

Erweiterbarkeit Gaming-PCs sind in der Regel so konzipiert, dass sie leicht aufgerüstet werden können, um neue oder leistungsstärkere Komponenten einzubauen.

A 6.3 (T1) Beschaffen Sie sich einen alten, nicht mehr benötigten (!) Computer und schrauben Sie diesen auf (Achtung, trennen Sie den Computer vorher vom Stromnetz!). Identifizieren Sie die CPU und den Hauptspeicher. Wie sind die Festplatten angeschlossen, schon seriell z. B. über SATA oder noch parallel mit einem Flachbandkabel? Welche Busse finden sich in dem Computer (ISA, EISA, PCI, PCIe)? Sehen sie irgendwo das BIOS / UEFI?

Lösung

A 6.4 (T1) Beantworten Sie die folgenden Fragen:

a) Nennen Sie mindestens drei prinzipiell unterschiedliche Rechnerarchitekturen.

b) Was versteht man unter dem Bus-Bottleneck?

c) Grenzen Sie die Begriffe RISC und CISC gegeneinander ab.

d) Erläutern Sie das EVA-Prinzip.

e) Was sind: RAM, ROM, EPROM, EEPROM?

f) Was ist der Unterschied zwischen einer Festplatte und einer Solid-State-Disk?

Lösung

a) Von-Neumann-Artchitektur (serielle Verarbeitung); Parallelverarbeitung (Pipeline, Vektorrechner, Prozessor-Arrays etc.); Neuronale-Prozessoren; Analog- und Hybridrechner; Quantencomputer

b) Bei der von-Neumann Architektur und damit verwandten Architekturen müssen Adressen, Daten und Befehle über denselben Bus transportiert werden, sie befinden sich in demselben Speicher. Diese als Von-Neumann-Flaschenhals oder Bus-Bottleneck beannte Engstelle bewirkt eine Geschwindigkeitseinbuße.

c) In Prozessoren mit RISC-Architektur (von Reduced Instruction Set Computer) beschränkt man sich auf einen eingeschränkten Befehlssatz. Die dementsprechend kurzen Befehls-Codes und die geringe Anzahl der zur Ausführung benötigten Maschinenzyklen – meist genügt ein Zyklus - erlauben daher in Verbindung mit einer größeren Anzahl von Registern modernere (superskalare) CPU-Architekturen mit Pipelining. Die einfachere Struktur ermöglicht zudem den Bau effizienterer Compiler.

CISC steht für Complex Instruction Set Computer. Hier gibt es teilweise weit über 1000 verschiedene Befehle und eine Vielzahl von Adressierungsarten. In modernen PC werden häufig Mischtypen verwendet (z. B. RISC-Kern in Hardware und komplexere Befehle in Microcode), so dass die Grenze zwischen RISC und CISC fließend ist.

d) Jede Form der Datenverarbeitung beinhaltet immer einen Ablauf der Art
Eingabe → Verarbeitung → Ausgabe

Man bezeichnet dies als EVA-Prinzip, wobei die Ein-/Ausgabegeräte die Schnittstelle zwischen Mensch und Maschine darstellen. Der Ablauf geschieht nach einem festen Schema (Programm), das über eine Eingabeeinheit (z. B. Tastatur oder externer Speicher) der Verarbeitungseinheit zugeführt wird.

e) **RAM** Random Access Menory. Speicher mit wahlfreiem Schreib- und Lesezugriff, der meist als Arbeitsspeicher während der Programmausführung verwendet wird. Er ist in der Regel volatil, d. h. sein ohne Betriebsspannung geht der Inhalt verloren.

ROM Read Only Memory. Ein Speicher, auf den nur lesend zugegriffen werden kann. Er ist nicht volatil, d. h. sein Inhalt bleibt auch ohne Betriebsspannung erhalten. Ein ROM wird als Festwertspeicher verwendet, insbesondere für Grundfunktionen (BIOS, Basic Input-Output System) des Betriebssystems.

PROM Programmable Read Only Memory. Wie ROM, aber durch ein Zusatzgerät einmal frei programmierbar.

EPROM Electrical Programmable Read Only Memory. Wie ROM, aber durch ein Zusatzgerät frei programmierbar, wobei Inhalte auch gelöscht und überschrieben werden können.

f) **Funktionsweise:** Eine Festplatte verwendet rotierende Magnetscheiben und bewegliche Leseköpfe, um Daten zu speichern und abzurufen. Die SSD speichert Daten in Flash-Speicherbausteinen ohne bewegliche Teile.

Geschwindigkeit: SSDs sind in der Regel schneller als Festplatten, da sie keine mechanischen Komponenten haben und Daten schneller finden und breitbandiger lesen und schreiben können. Eine Festplatte ist gezwungen die Daten durch die Rotation der Magnetscheiben und die Bewegung des Lesekopfes zu finden.

Robustheit: SSDs sind robuster und stoßunempfindlicher als Festplatten, da sie keine beweglichen Teile haben, ein sog. Plattenchrash, wo der Lesekopf die Magnetplatte zerkratzt ist nicht möglich.

Speicherkapazität und Preis: Festplatten bieten in der Regel größere Speicherkapazität zu günstigeren Preisen als SSDs.

Energieverbrauch und Lautstärke: SSDs verbrauchen weniger Energie und sind im Betrieb nahezu geräuschlos, die rotierenden Magnetscheiben erzeugen ein gewisses Betriebsgeräusch.

Lebensdauer und Datensicherheit: Die Festplatte hat theoretisch unbegrenzte Schreibzyklen, Datenrettung bei Defekten ist oft möglich. Sie ist aber mechanischem Verschleiß unterworfen. Die SSD bietet nur eine begrenzte Anzahl an Schreibzyklen, Datenrettung bei Defekten schwieriger.

A 6.5 (T1) Was ist der Unterschied zwischen Mikrocontrollern, DSPs und SoC?

Lösung

Mikrocontroller Microcontroller sind einfache, integrierte Systeme für spezifische Aufgaben, häufig in der Automatisierung (Steuerung, Regelung) und im Internet of Things eingesetzt. Sie enthalten CPU, Speicher und Peripherie auf einem Chip und sind für einfache Steuerungsaufgaben optimiert, auf niedrige Stückkosten und geringen Stromverbrauch.

Digitale Signalprozessoren (DSP) DSP sind spezialisiert auf die schnelle Verarbeitung digitaler Signale. Sie sind optimiert für sich wiederholende Multiply-Accumulate (MAC) Operationen. Häufig verwendet in digitaler Filterung und Fourier-Transformationen unter anderem für Audio- und Bildverarbeitung sowie beim Mobilfunk.

System-on-a-Chip (SoC) Ein SoC integriert alle Komponenten eines kompletten Computersystems auf einem Chip. Diese sind in der Regel die Hauptkomponente von Smartphones, z. B. der A17 von Apple oder der Snapdragon von Qualcomm, inzwischen folgen Intel und AMD mit ihren Desktop/Laptop-CPU auch dieser Architektur. Das SoC kombiniert Prozessor, Speicher, I/O-Schnittstellen und oft zusätzliche Funktionen wie GPU oder DSP.

A 6.6 (T2) Diskutieren Sie Vorteile und Nachteile der wesentlichen Merkmale der von-Neumann-Architektur, grenzen sie diese von der Harvard-Architektur ab.

Lösung

Die Von-Neumann-Architektur ist vor allem durch die serielle Abarbeitung von Algorithmen nach dem EVA-Prinzip (Eingabe-Verarbeitung-Ausgabe) gekennzeichnet. Im einfachsten Fall besteht ein von-Neumann-Rechner aus der Eingabeeinheit, einem Arbeitsspeicher, einem Zentralprozessor (CPU), der im wesentlichen Steuerwerk, Rechenwerk (ALU) und Register enthält, einer Ausgabeeinheit sowie Verbindungseinrichtungen zwischen diesen Komponenten. Nach der Flynn-Klassifikation gehören von-Neumann-Rechner zum Typ Instruction Single Data (SISD). Dieses vergleichsweise einfache Grundprinzip macht die Konstruktion einfacher, effizienter und preiswerter Rechner möglich. Ein wesentliches Designmerkmal ist ferner, dass für Programme und Daten nur ein gemeinsamer Arbeitsspeicher vorgesehen ist, der über einen Bus mit der Zentraleinheit gekoppelt ist. Gerade dies ist aber auch ein Nachteil, nämlich ein als Von-Neumann-Flaschenhals bezeichneter Engpass bei der Transferierung von Daten über den Bus. Im Gegensatz dazu hat die Harvard-Architektur getrennte Speicher für Programme und Daten.

A 6.7 (T2) Erläutern Sie die wesentlichen Merkmale der Flynn-Klassifikation. Was ist eine Schwäche dieser Klassifikation?

Lösung

Die Flynn-Notation definiert folgende Klassifikationen von Rechnern:

SISD Beim Typ Single Instruction Single Data (SISD) wird ein Datenstrom gemäß einer seriellen Befehlsfolge verarbeitet. Dazu gehören auch die von-Neumann-Rechner.

SIMD Beim Typ Single Instruction Multiple Data (SIMD) werden mehrere Datenströme gleichzeitig gemäß einem für alle Prozessoren identischen Befehlsstrom verarbeitet. In diese Klasse gehören Array-

Prozessoren und GPUs oder auch die Vektor-Befehle moderner Prozessoren.

MIMD Der Typ Multiple Instruction Multiple Data (MIMD) ist der allgemeinste und am wenigsten spezifische: Mehrere Datenströme werden durch mehrere Befehlsströme verarbeitet. Hier finden sich die meisten moderen Rechner wieder (Multicore-, Multiprozessorsysteme, ...).

MISD Die Variante Multiple Instruction Single Data (MISD) beinhaltet ein Parallel-Konzept auf Befehlsebene, wobei jedoch nur ein Datenstrom bearbeitet wird. Dieses interne Befehls-Pipelining begründet an sich aber keine eigenständige Prozessorarchitektur. Daher ist diese Klasse in der Praxis nicht relevant und viele Autoren lassen sie leer.

Eine Schwäche der Flynn-Klassifikation ist, dass gerade der Typ MIMD, der zahlreiche Varianten und Realisierungsmöglichkeiten beinhaltet, nicht weiter unterteilt wird.

A 6.8 (T2) Beschäftigen Sie sich mit GPUs und erstellen eine Übersicht über wichtige Konzepte. Was versteht man unter CUDA? Was ist ein CUDA-Core bzw. ein Stream-Prozessor?

Lösung

Was ist eine GPU?

Eine GPU (Graphics Processing Unit) ist ein spezialisierter Prozessor, der ursprünglich entworfen wurde, um Grafikberechnungen zu beschleunigen. Derzeit werden GPUs auch für wissenschaftliche Berechnungen, Bitconin-Mining und maschinelles Lernen eingesetzt. Für Laptops und Gaming-PCs gibt es derzeit zwei Hersteller: Nvidia und AMD.

Parallele Verarbeitung

GPUs sind für massiv parallele Verarbeitung ausgelegt. Sie enthalten Hunderte oder Tausende von kleineren spezialisierten Kernen bzw. einfacheren ALUs. Dies unterscheidet sich von einer traditionellen CPU (Central Processing Unit), die weniger aber allgemeinere Kerne hat. Eine aktuelle Grafikkarte, die NVIDIA GeForce RTX 4090, Listenpreis ca. 1800 Euro (Stand März 2023) hat 76. 3 Mrd. Transistoren und 16 384 CUDA-Kerne verteilt auf 128 Ray-Tracing Cores. Darin enthalten sind auch 512 Tensor-Cores (also 128 CUDA-Kerne und 4 Tensor-Kerne pro Ray-Trace-Core). Als Grafikspeicher stehen 24GB G-DDR6X zur Verfügung. Nvidia bezeichnet die Architektur mit dem Namen der Computer Pionierin Ada Lovelace.

Stream-Prozessoren, CUDA-Kerne und Shader-Kerne

Ein CUDA-Kern[1] ist eigentlich eine einfache ALU (Arithmetische Logische Einheit). Diese beherrscht die Grundrechenarten und logische Operationen. Jede Operation wird jeweils in einem Taktzyklus durchgeführt. Mithilfe der Bündelung von vielen solcher Kerne können umfangreichere Berechnungen durchgeführt werden, besonders mit Vektoren. Die vielen kleinen Rechenkerne verarbeiten eher einen kontinuierlich anfallenden Datenstrom. Daher ist die Bezeichnung Streamprozessor auch passend.
Nvidia bündelt in ihrer aktuellen Ada-Lovelace-Architektur 128 CUDA-Kerne zu einem SM (Streaming Multiprocessor). Dieser dient im Wesentlichen zur Verarbeitung von Datenvektoren (SIMD). In der oben schon erwähnten GPU (AD102-301) der Grafikkarte GeForce RTX 4090 finden sich insgesamt 128 dieser SMs. Jede SM enthält noch weitere spezialisierte Rechenkerne, sog. Tensor-Cores.

[1] bei NVIDIA

Speicherarchitektur

GPUs haben eine hierarchische Speicherarchitektur, die schnellen Zugriff auf Daten ermöglicht, die von den Kernen verarbeitet werden. Dazu gehören dedizierter Grafikspeicher (oft G-DDR), Caches auf verschiedenen Ebenen und gemeinsam genutzter Speicher, der von Gruppen von Kernen für die Koordination und Datenübertragung genutzt wird.

Pipeline für Grafikverarbeitung

Die traditionelle Rolle einer GPU besteht darin, die Grafikpipeline zu beschleunigen, ein Prozess, der 3D-Modelle in die 2D-Bilder umwandelt, die auf einem Bildschirm angezeigt werden.

API-Unterstützung

Zur Steuerung der GPU-Funktionen und zur Entwicklung von Anwendungen, die GPUs nutzen, gibt es mehrere Programmierschnittstellen (APIs) und Entwicklungsumgebungen. Beliebt ist unter anderem CUDA (Compute Unified Device Architecture), eine von Nvidia entwickelte Plattform, die speziell für ihre GPUs optimiert ist.

A 6.9 (T2) Die Unterstützung neuronaler Netze in Hardware ist gerade unübersichtlich, Hersteller sprechen von TPU (Google), VPU (Intel) oder APU (AMD). Sind diese Produkte vergleichbar? Wofür werden sie eingesetzt?

Lösung

Die Unterstützung neuronaler Netze durch spezialisierte Hardware ist ein Bereich mit vielen Akteuren und unterschiedlichen Technologien für mobile Geräte, Desktop-Rechner oder Rechenzentren in der Cloud, für das Training oder die Anwendung neuronaler Netze (Inferenz). Jeder Hersteller entwickelt Hardware, die auf bestimmte Aspekte der KI- und Maschinenlern-Workflows abzielt, was zu einer Vielzahl von Begriffen und Produkten führt. Die am häufigsten diskutierten sind TPUs (Tensor Processing Units) von Google, VPUs (Vision Processing Units) von Intel und APUs (Accelerated Processing Units) von AMD. Hier ist ein Überblick über diese Technologien und wie sie sich unterscheiden:

TPU (Tensor Processing Unit, Google)

Entwickelt für maschinelles Lernen und insbesondere für den Einsatz mit TensorFlow, Googles Framework für maschinelles Lernen. TPUs sind darauf optimiert, große Mengen an Matrix- und Vektorberechnungen effizient durchzuführen, die häufig in neuronalen Netzwerken vorkommen.
TPUs werden hauptsächlich in Rechenzentren verwendet, um maschinelles Lernen und Deep Learning-Anwendungen zu beschleunigen, insbesondere solche, die mit TensorFlow arbeiten. Sie eignen sich besonders gut für Inferenz- und Trainingsaufgaben in neuronalen Netzwerken.

VPU (Vision Processing Unit), Intel

Optimiert für Computer Vision-Aufgaben. VPUs sind darauf ausgelegt, Bildverarbeitung und Computer Vision in Echtzeit zu ermöglichen, oft mit einem Fokus auf Energieeffizienz für den Einsatz in Edge-Geräten. VPUs finden Anwendung in einer Vielzahl von Geräten, von Drohnen und Überwachungskameras bis hin zu Smartphones und autonomen Fahrzeugen, wo sie Aufgaben wie Objekterkennung, -klassifizierung und -tracking übernehmen.

APU (Accelerated Processing Unit, AMD)

Eine APU ist eigentlich eine Integration eines herkömmlichen CPU-Kerns mit einem leistungsstarken Grafikprozessor (GPU) auf einem einzigen Chip. Obwohl APUs nicht speziell für maschinelles Lernen oder neuronale Netze entwickelt wurden, können ihre GPU-Komponenten für solche Aufgaben genutzt werden. APUs werden hauptsächlich in Verbrauchergeräten wie Laptops, Desktops und Spielekonsolen eingesetzt. Ihre Stärke sind grafikintensive Aufgaben und allgemeine Berechnungen, dies ist nutzbar für maschinelles Lernen und Computer Vision.

7 Lösungen zu Kapitel 7 – Rechnernetze

A 7.1 (L1) Machen Sie sich mit Ihrem Laptop/PC oder Smartphone vertraut:

- Welche Netzwerkschnittstellen hat Ihr eigener Laptop, Tablet-Computer bzw. Smartphone? Welche WLAN-Standards werden unterstützt? Welche Übertragungsraten können Sie damit erreichen? Wenn ein Gerät über eine Ethernet-Schnittstelle verfügt, um welches Ethernet handelt es sich genau? Welche Bluetooth-Version wird unterstützt und was genau bedeutet das?
- Finden Sie die MAC-Adresse Ihres Laptops, Tablet-Computers, Smartphones in Ihrem eigenen LAN oder WLAN heraus! Was genau ist eine MAC-Adresse? Rufen Sie beispielsweise unter Windows das Kommando `ipconfig /all` auf. Was genau sehen Sie dort noch?
- Finden Sie die IP-Adresse Ihres Laptops/PCs/Smartphones in Ihrem eigenen LAN oder WLAN heraus! Ist das eine IPv4 oder eine IPv6 Adresse? Unter Windows können Sie dazu beispielsweise das Kommando `netstat` in Ihrer Konsole verwenden. Was genau ist die Ausgabe von `netstat`? Warum taucht dort die Adresse 127.0.0.1 so oft auf?
- Wie sind Sie mit dem Internet verbunden? Welchen Gateway verwenden Sie dazu und welche IP-Adresse hat dieser (`ipconfig /all`)?
- Auf welchen DNS-Server greifen Sie zu? Wozu brauchen Sie DNS?

Lösung

Die Übungsaufgaben dienen dazu, dass Sie Ihren eigenen Rechner bzw. weitere Geräte wie Smartphones oder Tablet-Computer genauer kennenlernen. Nachfolgend untersuchen wir gemeinsam Ihr Netzwerk zuhause bzw. das Netzwerk in Ihrer Arbeitsumgebung. Sie benötigen dafür die Windows-Konsole oder ein Terminalfenster auf Linux oder MacOS.
Auf einem Windows-System gelangen Sie zur Konsole, indem Sie das Programm `cmd.exe` ausführen. Die Abb. 7.1 zeigt, wie das Programm auf Windows 10 ausgeführt werden kann. Eventuell müssen Sie unter Administrator-Rechten laufen lassen (Rechte Maustaste und '*als Administrator ausführen*' aufrufen.
Entsprechend sieht das Konsolen-Fenster auf Ihrem Desktop aus. Ein Beispiel gibt die Abb. 7.2. Hier können Sie die unten dargestellten Kommandos eingeben, unten ist nur noch die textuelle Ausgabe des jeweiligen Kommandos dargestellt, auf einen Screenshot wurde jeweils verzichtet.

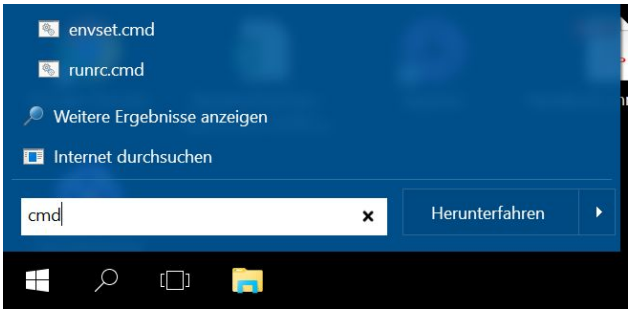


Abb. 7.1 Screenshot aus Windows 10 (Klick auf das Windows-Symbol), dann CMD eingeben.

Abb. 7.2 Konsolen-Fenster unter Windows 10.

Netzwerkschnittstellen

Auf der Konsole: `ipconfig /all`

Auf einer Windows-Maschine haben Sie verschiedene Möglichkeiten, Informationen zu ihren Netzwerkschnittstellen zu bekommen: Rufen Sie die Konsole (Shell-Fenster) auf und geben Sie `ipconfig /all` ein. Dort werden alle Netzwerkadapter mit samt ihrer Konfiguration genannt. Die nachfolgende Ausgabe von `ipconfig /all` auf einem Rechner der Autoren ist stark gekürzt. Sie soll einen Eindruck vermitteln. Sie erkennen die MAC-Adressen der jeweiligen Netzwerkadapter für LAN und WLAN.

```
C:\>ipconfig /all

Windows-IP-Konfiguration

   Hostname  . . . . . . . . . . . . : LTB12345
*gekuerzt*

Ethernet-Adapter Ethernet 2:

   Medienstatus. . . . . . . . . . . : Medium getrennt
   Verbindungsspezifisches DNS-Suffix:
   Beschreibung. . . . . . . . . . . : Intel(R) Ethernet Connection I219-V
   Physische Adresse . . . . . . . . : 12-34-AB-CD-56-78
   DHCP aktiviert. . . . . . . . . . : Ja
*gekuerzt*

Drahtlos-LAN-Adapter WLAN:

   Verbindungsspezifisches DNS-Suffix: Speedport_W_723V
   Beschreibung. . . . . . . . . . . : Intel(R) Dual Band Wireless-AC 8260
   Physische Adresse . . . . . . . . : 90-34-AB-CD-56-78
   DHCP aktiviert. . . . . . . . . . : Ja
   IPv4-Adresse  . . . . . . . . . . : 192.168.0.108(Bevorzugt)
   Subnetzmaske  . . . . . . . . . . : 255.255.255.0
   Standardgateway . . . . . . . . . : 192.168.0.1
   DHCP-Server . . . . . . . . . . . : 192.168.0.1
   DNS-Server  . . . . . . . . . . . : 192.168.0.1
```

Auf dem Rechner des Autors ist gut zu erkennen, dass die IP-Adressen über DHCP (siehe unten) vergeben werden. Im Rechner selbst ist keine IP-Adresse konfiguriert. Der Rechner *LTB12345* ist gerade über seinen WLAN-Adapter im Internet und hat im lokaken Netz die IP-Adresse `192.168.0.108`. An der Adresse ist leicht zu erkennen, dass es sich um eine private Andresse handelt. Die IANA, welche die Internet-Adresse weltweit verteilt, hat bestimmte Adressbereiche nicht vergeben. Das sind *private* IP-Adressen. Unter anderem `192.168.0.0/16`. Damit können in lokalen Netzen die Adressen `192.168.0.0` bis `192.168.255.255` frei genutzt werden. Zwei weitere Adressbereiche stehen zur Verfügung, das sind `172.16.0.0` bis `172.31.255.255` und `10.0.0.0` bis `10.255.255.255`.
Der Rechner befindet sich in einem Subnetz mit der Subnetzmaske `255.255.255.0`. Daran ist erkennbar, welche Rechner sich in demselben Subnetz befinden. Die drei 255-Werte zeigen an, dass alle Rechner in

denen die ersten drei Blöcke der IP-Adresse übereinstimmen im selben Subnetz sind. Im Beispiel sind das die Rechner mit den IP-Adressen von `192.168.0.1` bis `192.168.0.254`. Broadcast-Nachrichten werden nur an Rechner in dem selben Subnetz verteilt. Die Subnetzmaske definiert damit auch den Namen des Subnetzes selbst, das ist die erste mögliche IP-Adresse. Das Subnetz heißt also `192.168.0.0`. Die letzte mögliche IP-Adresse dient dazu, alle Rechner des Subnetzes zu adressieren, in unserem Fall ist `192.168.0.255` die Broadcast-Adresse.
Auch die MAC-Adressen beider Netzwerk-Adapter (=Physische Adresse) werden dargestellt. Der WLAN-Adapter hat die MAC-Adresse `90-34-AB-CD-56-78`. Diese Adresse kennzeichnet den WLAN-Adapter weltweit eindeutig. MAC steht für Media Access Controll und ist das Protokoll, das im *lokalen Netz* dafür sorgt, dass jeder Teilnehmer Zugriff auf das Kommunikationsmedium bekommt. Also die Luft-Schnittstelle im WLAN bzw. das Ethernet-Kabel. MAC liegt in der Schicht 2 des ISO/OSI-Modells, während das oben diskutierte Internet Protocol auf Schicht 3 liegt.
Der Router *Speedport_W_723V* wurde von der Telekom bereitgestellt und übersetzt lokale IP-Adresse per NAT-Protokoll (siehe unten) in eine im ganzen Internet sichtbare IP-Adresse, diese ist hier nicht zu sehen. Der Router hat die lokale IP-Adresse `192.168.0.1`, das ist typisch. Auf ihm laufen alle für das lokale Netz relevanten Dienste wie DHCP und DNS. Diese wird vom Internet-Provider vergeben. Im lokalen Netz wird noch das Internet-Protokoll IPv4 verwendet.

Windows-Systemsteuerung

In Windows können Informationen zur Konfiguration des Rechners über die Systemsteuerung gefunden werden: im '`Netzwerk- und Freigabecenter`'. Im Dialog *Grundlegende Informationen zum Netzwerk anzeigen und Verbindungen einrichten* (Windows10), wird das aktuell verwendete Netzwerk dargestellt, von dort aus kann man sich bis zu den Netzwerkverbindungsdetails durchklicken.

Internet zuhause untersuchen

Im Internet finden sich viele Werkzeuge, um Ihren Anschluss zuhause zu untersuchen. Ein Beispiel ist `https://www.telekom.de/netz/speedtest`. Dort wird überprüft, welche Übertragungsgeschwindigkeiten Ihnen beim Upload und Download von Daten aus dem Internet zur Verfügung stehen. Weiterhin werden Netzwerklatenzen gemessen. Die Messung eines Autors ist in Abb. 7.3 zu sehen.

Welche Verbindungen sind von Ihrem Windows Rechner gerade offen?

Der Aufruf von `netstat -n` auf einer Konsole liefert sehr interessante Erkenntnisse: Es werden die aktuellen TCP und UDP Verbindungen auf Ihrem Rechner angezeigt. Man ist überrascht, wie viele Programme gerade ohne Wissen des Benutzers mit anderen Servern aus dem Internet kommunizieren. Das folgende Listing zeigt einen Ausschnitt der Konsolenausgabe des Autors.

```
TCP    192.168.0.108:21974    40.77.229.72:443       HERGESTELLT
TCP    192.168.0.108:22587    172.217.22.206:443     HERGESTELLT
TCP    192.168.0.108:22663    216.58.205.100:443     HERGESTELLT
```

Zu sehen sind hier drei Verbindungen des Rechners mit der Adresse `192.168.0.108` mit verschiedenen Hosts im Internet, beispielsweise gehört der Rechner `40.77.229.72` zu Microsoft. Hinter den IP-Adressen wird jeweils die Port-Nummer genannt, diese ist per Doppelpunkt getrennt angegeben. Die erste Zeile zeigt, dass beim Kommunikationspartner der Port 443 verwendet wird. Das deutet auf die Verwendung von HTTPS bzw. TLS hin. Dieser Port wird beim Verbindungsaufbau vom Programm, das die Verbindung benötigt, mit angegeben. Dieser Port gibt an, welcher Dienst auf dem angesprochenen Rechner genutzt werden soll. Der Port 21974 auf dem lokalen Rechner (outbound) wurde vom lokalen Betriebssystem zugewiesen.

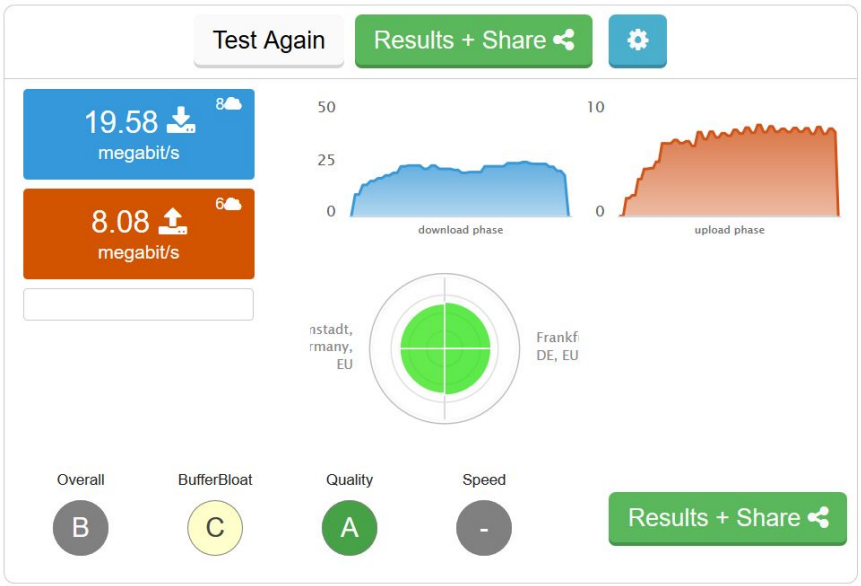


Abb. 7.3 Screenshot einer Geschwindigkeitsmessung der Internetverbindung

A 7.2 (L1) Machen Sie sich mit dem Netzwerk in Ihrer Universität, Hochschule oder Firma vertraut:

- Welche IP-Adresse hat der Webserver Ihrer Universität, Hochschule oder Firma? Sie können das beispielsweise mithilfe des `ping`-Kommandos herausfinden.
- Untersuchen Sie, in welchem Subnetz sich Ihre Universität, Hochschule oder Firma befindet. Verwenden Sie dazu beispielsweise Ihren Browser mit `https://www.whois.com/whois/`. Welche Subnetzmaske ist hier erforderlich? Wie viele Bit umfasst Ihre Subnetzadresse?

Lösung

Ein `ping www.fh-rosenheim.de`, das ist der Webserver der Hochschule Rosenheim, liefert beispielsweise folgende Ausgabe:

```
C:\>ping www.fh-rosenheim.de

Ping wird ausgefuehrt fuer www.fh-rosenheim.de
[141.60.160.196] mit 32 Bytes Daten:
Antwort von 141.60.160.196: Bytes=32 Zeit=29ms TTL=63
Antwort von 141.60.160.196: Bytes=32 Zeit=29ms TTL=63
Antwort von 141.60.160.196: Bytes=32 Zeit=29ms TTL=63
Antwort von 141.60.160.196: Bytes=32 Zeit=29ms TTL=63

Ping-Statistik fuer 141.60.160.196:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0
    (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 29ms, Maximum = 29ms, Mittelwert = 29ms
```

Der DNS-Server hat offenbar die Adresse `www.fh-rosenheim.de` in die IPv4 Adresse `141.60.160.196` übersetzt. Das `ping`-Kommando liefert auch Informationen über die Güte der Verbindung zum angesprochenen Server, also die Verlustrate und Latenzeiten.
Interessant ist Ausgabe von Whois (siehe Aufgabenstellung). Für eine Firma oder Hochschule mit der Endung `.de` können Sie die Seite der `denic.de` verwenden. Dort erkennen Sie, wer Ihre Firma bzw. Hochschule vertritt und wie Sie diese Person erreichen können.

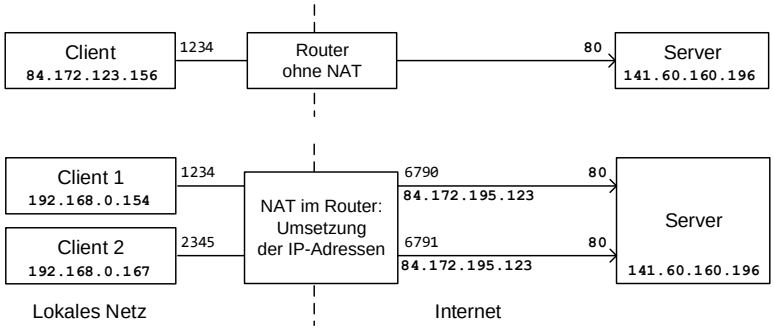


Abb. 7.4 Darstellung der Network Address Translation an einem Beispiel. Hier verwendet ein Router NAT. Im öffentlichen Internet haben alle Rechner aus dem Intranet alle dieselbe IP-Adresse.

A 7.3 (L1) Was genau macht das DHCP-Protokoll? Ermitteln Sie dessen Eigenschaften über eine Internetrecherche. Wo findet sich das ARP-Protokoll und wofür ist dieses zuständig?

Lösung

NAT: Network Address Translation

NAT steht für *Network Address Translation*. Ein Router oder Gateway übersetzt IP-Adressen zwischen zwei verbundenen Netzwerken.
Typischerweise ist dies das Netzwerk des Internet-Providers bzw. das Internet sowie das lokale Netz daheim oder in der Firma. Wenn die IP-Adresse Ihres Rechners oder Smartphones mit `192.168.`, mit `172.` oder mit `10.` beginnt, dann ist NAT mit Sicherheit im Spiel. Diese Adressen sind private IP-Adressen, die im öffentlichen Internet nicht vergeben sind. Der Router bzw. das Gateway daheim, in Ihrer Firma oder Hochschule muss diese IP-Adresse noch in eine IP-Adresse übersetzen, die im öffentlichen Internet gültig ist. Beispielsweise gehören der deutschen Telekom die IP-Adressen `91.32.0.0` bis `91.63.255.255` (Auskunft über Whois). Die im Internet sichtbare Adresse des Rechners (es handelt sich eigentlich um die IP-Adresse des Telekom-Routers) des Autors lautet beispielsweise im Telekom-Netz `91.51.55.87`, im lokalen Netz hat derselbe Rechner die IP-Adresse `192.168.0.108`. Im öffentlichen Netz ist nur der Router sichtbar und nicht der Rechner des Autors.
Für das Weiterleiten der Datenpakete zwischen beiden Netzen wird für jede Verbindung zwischen zwei Rechnern jeweils eine eigene Sitzung erzeugt. Das ist notwendig damit der Router die übersetzung der Sender- und Empfänger-IP-Adressen durchführen kann.

NAT und IPv4

NAT hat in den letzten Jahren sehr an Bedeutung gewonnen, denn die IPv4 Adressen sind inzwischen vollständig vergeben worden. Längst sind mehr als die eigentlich maximal möglichen 4 Milliarden ($= 2^{32}$) Geräte im Internet unterwegs. Diese Zahl wird bereits durch die Verbreitung der Smartphones überschritten[1].
NAT ermöglicht es, mit mehreren Geräten gleichzeitig dieselbe IP-Adresse (IPv4) zu nutzen. Typischerweise teilen sich alle Geräte hinter einem Router (in privaten Haushalten) dessen im Internet sichtbare IP-Adresse. Der Router ordnet über NAT für jede Verbindung die richtige Maschine im lokalen Netz zu.
Ein weiterer Vorteil dieses Verfahrens ist auch, dass Angreifer aus dem Internet nicht einfach auf einen Rechner im lokalen Netz zugreifen können, da dieser nur über den Router und die darin in der Regel

[1] `https://de.statista.com/statistik/daten/studie/309656/umfrage/prognose-zur-anzahl-der-smartphone-nutzer-weltweit/`

enthaltene Firewall zugreifen kann.

NAT und Port-Weiterleitung in IPv4

Wenn ein Dienst auf einem Rechner aus dem lokalen Netz im öffentlichen Internet sichtbar sein soll, ist die Port Weiterleitung eine Option. Im Router wird ein NAT-Weiterleitungs-Port konfiguriert, beispielsweise für typische Dienste wie HTTP (Port 80 oder 443) oder SSH (Port 22). Diesem Port wird im Router ein Rechner (private IP-Adresse) aus dem lokalen Netz sowie der Port des dort installierten Dienstes zugeordnet. Wenn aus dem Internet auf die IP-Adresse des Routers über den konfigurierten Port zugegriffen wird, wird dieser Zugriff auf den Rechner im lokalen Netz auf dem entsprechenden Port weitergeleitet.

NAT und IPv6

NAT erhält IPv4 noch am Leben. Mit der Nachfolgeversion des Protokolls IPv6 wurde der Adressraum so vergrößert, dass man jedem Fussel auf diesem Planenten eine eigene IP-Adresse (v6) zuordnen kann. Es stehen dort bis zu $= 2^{128}$ Adressen zur Verfügung. Der oben dargestellte Anwendungsfall für NAT entfällt also mit dem Umstieg auf IPv6.

DHCP: Dynamic Host Configuration Protocol

In einem lokalen Netz ist es aufwendig, jedem Rechner manuell eine IP-Adresse zuzuordnen. Es ist wahrscheinlich, dass versehentlich IP-Adressen mehrfach vergeben werden und so Konflikte entstehen. Viele Rechner sind inzwischen mobil, der klassische Desktop-PC stirbt gerade zugunsten von Laptops und Tablet-Computern aus. Damit bewegen sich diese mobilen Rechner in verschiedenen lokalen Netzen. Eine dem Rechner fest zugeordnete Adresse führt auch in diesem Szenario zu Problemen.
Dieses Problem löst DHCP, das Dynamic Host Configuration Protocol (RFC 2131). Ein DHCP-Server wird im Netzwerk daheim meistens auf dem Router bereitgestellt, in größeren lokalen Netzwerken gibt es dafür einge Server.
Wenn ein Rechner in einem lokalen Netzwerk gestartet wird (und das entsprechend konfiguriert ist), kann er sich über den DHCP-Server eine gültige IP-Adresse beschaffen. Dazu macht der Rechner zunächst den DHCP-Server über eine Broadcast-Nachricht ausfindig. Er kann eventuell zwischen mehreren verfügbaren DHCP-Servern wählen. Er erhält nach dem Verbindungsaufbau vom ausgewählten DHCP-Server eine Adresse zugeordnet und es werden weitere Informationen, beispielsweise die Adresse des DNS-Servers übermittelt. Die Abbildung 7.5 zeigt einen Screenshot aus der Systemsteuerung eines Windows-Rechners. Das DHCP-Protokoll ist aktiviert, sodass der Rechner automatisch die IP-Adresse bezieht. Mit Administrationsrechten könnte man auch eine feste IP-Adresse einstellen.

ARP: Address Resolution Protocol in IPv4

Im öffentlichen Internet werden Rechner über deren IP-Adressen angesprochen. Daten werden TCP/IP-Pakete bzw. UDP/IP-Pakete ausgetauscht. Sender und Empfänger werden über ihre IP-Adressen identifiziert. Diese Kommunikation läuft auf Schicht 3 des ISO/OSI-Modells.
In einem Subnetz kommunizieren die Rechner über ein gemeinsames Netzwerk, häufig ein gemeinsames Ethernet-LAN oder WLAN. Hier können die Rechner ohne weiteres Routing direkt miteinander kommunizieren. Beide Kommunikationspartner müssen dazu die physische Adresse des jeweils anderen kennen, das ist die MAC-Adresse.
Das ARP (Address Resolution Protocol) bildet die IP-Adresse (v4) aus Schicht 3 des ISO/OSI-Modells auf die MAC-Adresse (Schicht 2) der ensprechenden Rechner bzw. Geräte innerhalb eines lokalen Netzes ab.

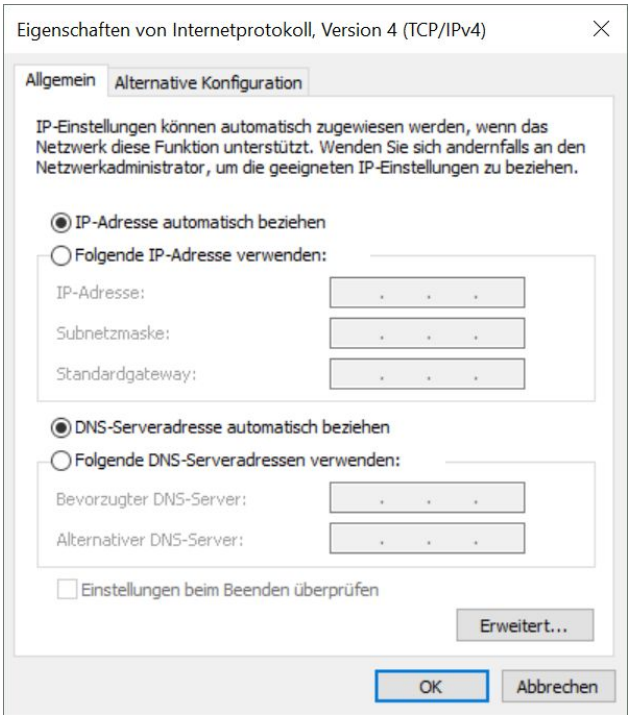


Abb. 7.5 Screenshot des Dialogs aus der Systsemsteuerung zur Konfiguraiton von IPv4

ARP ist im Betriebssystem implementiert. Es wird in IPv6 durch das NDS (Neighbor Discovery Protocol, RFC4861) ersetzt.
Wenn ein Rechner einen anderen Rechner in demselben Subnetz erreichen will und die IP-Adresse noch unbekannt ist, wird ein ARP-Request ausgeführt. Der Rechner schickt eine Broadcast-Nachricht an alle Rechner im Subnetz. Die Nachricht enthält die IP-Adresse und die MAC-Adresse des Senders sowie die IP-Adresse des gesuchten Rechners. Befindet sich der Rechner mit der gesuchten IP-Adresse im Subnetz, schickt er die Nachricht inklusive seiner eigenen MAC-Adresse zurück. Beide Rechner können nun im lokalen Netz miteinander kommunizieren, da sie gegenseitig die physischen Adressen kennen. ARP speichert die Zuordnung MAC-Adresse zu IP-Adresse lokal in einem Cache. Es wird empfohlen, diesen nach 300 Sekunden wieder zu löschen.
Sie können die Funktionsweise des ARP selbst testen. Sie benötigen mindestens einen weiteren Rechner, dessen IP-Addresse Sie kennen in Ihrem Subnetz. Das Konsolen-Kommando `arp -a` zeigt Ihnen die aktuelle ARP-Tabelle auf Ihrem Windows-Rechner an. Sie sehen die Zuordnung von IP-Adressen zu den MAC-Adressen der jeweilgen Geräte. Wenn Sie mit PING ein Gerät ansprechen, beispielsweise Ihr Smartphone, das Ihr Windows-Rechner noch nicht adressiert hat, können Sie sehen, wie die MAC- zu IP-Auflösung über ARP verwaltet wird.

```
C:\>arp -a

Schnittstelle: 192.168.0.108 --- 0x2
  Internetadresse       Physische Adresse     Typ
  192.168.0.1           7d-4e-b5-8c-d8-60     dynamisch
  192.168.0.255         ff-ff-ff-ff-ff-ff     statisch
*gekuerzt*

C:\>ping 192.168.0.104
C:\>arp -a

Schnittstelle: 192.168.0.108 --- 0x2
  Internetadresse       Physische Adresse     Typ
```

```
  192.168.0.1           7d-4e-b5-8c-d8-60     dynamisch
  192.168.0.104         99-ec-62-16-b4-8d     dynamisch
  192.168.0.255         ff-ff-ff-ff-ff-ff     statisch
*gekuerzt*
```

Gut zu erkennen ist in dem Beispiel auch die Broadcast-Adresse. In unserem Subnetz ist das die letzte verfügbare IP-Adresse `192.168.0.255`. Ihr ist standardmäßig die Broadcast-MAC-Adresse `ff-ff-ff-ff-ff-ff` zugeordnet.

A 7.4 (L2) Untersuchen Sie Ihr lokales Netz genauer: Installieren Sie dazu Wireshark (`https://www.wireshark.org/`) auf Ihrem Rechner. Nehmen Sie nun den Netzwerkverkehr Ihres Rechners mit Wireshark für einige Minuten auf. Versuchen Sie Nachrichten zu Protokollen wie TCP oder UDP zu finden. Welche Details können Sie erkennen?

Lösung

A 7.5 (L2) Untersuchen Sie das HTTP-Protokoll: Was bedeuten die Status-Codes 200, 201, 204, 400, 404, 418 sowie 500? Machen Sie sich mit dem Kommando `curl` vertraut (eventuell müssen Sie dieses nachinstallieren). Was können Sie damit tun? Setzen Sie mindestens einen GET-Request und einen POST-Request ab. Wozu dienen die HTTP-Requests DELETE und PUT?

Lösung

HTTP-Verben

In den letzten Jahren werden Server und Dienste im Internet zunehmend mit RESTful Schnittstellen versehen. REST stellt eine bestimmte Art und Weise dar, eine entfernt aufrufbare Schnittstelle anzubieten. REST arbeitet mit Ressourcen, das können Hypertext-Seiten genauso wie Vertrags- oder Kundendaten sein. Ressourcen werden über ihren URI (Uniform Ressource Identifier, RFC 3986) eindeutig identifiziert. Das HTTP-Protokoll erlaubt es mit seinen Request-Typen (den Verben), das Anlegen (POST-Request), Ändern (PUT-Request, PATCH-Request), Löschen (DELETE-Request) und Suchen (GET-Request) dieser Ressourcen durchzuführen.
Ihr Browser kann typischerweise nur GET und POST. Mit entsprechenden Plugins, wie etwa Postman[2] kann man alle Arten von Requests absetzen. In der Konsole wird auf Linux und Windows-Systemen auch der Kommandozeilen-Browser `CURL` verwendet.

```
curl -d '{"keyA":"valueA",␣"keyB":"valueB"}'
    -H "Content-Type:␣application/json"
   -X POST http://customer.example.com/customers
```

Der Aufruf von `CURL` im Beispiel verwendet einen POST-Request um eine neue Ressource anzulegen. Die Daten werden im JSON-Format übermittelt, das ist im Wesentlichen ein Schlüssel/Wert-Paar Datenformat (z. B. `{"keyA":"valueA", "keyB":"valueB"}`), urpspünglich aus JavaScript. Die neue Ressource wird unterhalb der URI `http://customer.example.com/customers` angelegt.

HTTP-Status-Codes

Weiteres Bordmittel sind die HTTP-Status-Codes. Diese können nachgelesen werden in der HTTP-Spezifikation[3]. Nachfolgend einige ausgewählte Beispiele:

[2] siehe `https://www.getpostman.com/`
[3] `https://tools.ietf.org/html/rfc7231`

200 OK Der Request war erfolgreich. Alle Status-Codes mit einer 2 am Anfang zeigen erfolgreiche Requests an.

201 Created Eine oder mehrere Ressouren wurden erfolgreich angelegt.

204 No Content Der Request wurde erfolgreich verarbeitet, hat aber kein Ergebnis erzeugt.

400 Bad Request Der Server weigert sich, den Request zu verarbeiten, da vermutlich ein Fehler auf Client-Seite vorliegt. Beispielsweise Fehler in der Syntax des Aufrufs oder in den übermittelten Daten.

403 Forbidden Der Server verweitert den Zugriff, da dem Client die Zugriffsrechte fehlen. Diese Antwort führt bei Browsern häufig dazu, dass ein Login-Fenster erzeugt wird (HTTP-Basic Auth).

404 Not Found DIE klassische Fehlermeldung. Die im Request genannte Ressource (URI) gibt es nicht (mehr).

500 Internal Server Error Zeigt einen Fehler innerhalb des Servers an. Deswegen konnte der Request nicht verarbeitet werden.

8 Lösungen zu Kapitel 8 – Betriebssysteme

A 8.1 (T1) Welches Betriebssystem befindet sich in welcher Version auf Ihrem Smartphone? Welche wesentlichen Eigenschaften hat dieses Betriebssystem?

Lösung

Auf Smartphones sind im wesentlichen zwei Betriebssysteme verbreitet, das sind iOS ca. 36% und Android ca. 64% (Stand: Dezember 2023). Unterschiedliche Versionen dieser Systeme sind verbreitet. Ein Betriebssystem auf einem Smartphone hat folgende Aufgaben, vergleichbar mit den Aufgaben auf einem Laptop oder Desktop PC:

Hardware-Abstraktion und -Steuerung Ein Betriebssystem kapselt die Hardware vor dem Nutzer und den Anwendungen häufig über einen HAL (Hardware Abstraction Layer). Die Hardware von Smartphones ist bei Android recht heterogen. Das Betriebssystem bietet einheitliche Schnittstellen (APIs), über die Software auf Hardwarefunktionen wie Kamera, GPS, Display, Sensoren und Netzwerkverbindungen zugreifen kann, ohne die spezifischen Details der Gerätehardware kennen zu müssen.

Prozessverwaltung Das Betriebssystem ist für das Starten, Ausführen und Beenden von Apps zuständig. Es verwaltet Apps, Prozesse und Threads, indem es CPU-Zeit zuweist, Prioritäten setzt und sicherstellt, dass Anwendungen effizient und ohne Konflikte miteinander laufen.

Speichermanagement Es überwacht und steuert die Nutzung des physischen Speichers (RAM) und des Speicherplatzes. Dazu gehört das Allokieren von Speicher für Programme und Daten, das Verwalten des virtuellen Speichers sowie das Freigeben und Recycling von Speicherplatz, der nicht mehr benötigt wird. Apps, die gerade nicht verwendet werden, werden bei Bedarf in den Sekundärspeicher verschoben.

Datenspeicherung und -verwaltung Das Betriebssystem verwaltet das Dateisystem und eventuelle externe Speichermedien. Es organisiert Daten, ermöglicht das Speichern, Abrufen und Aktualisieren von Dateien und kontrolliert den Zugriff auf diese Daten, um Sicherheit und Datenintegrität zu gewährleisten.

Netzwerkmanagement Es steuert die Kommunikation des Smartphones mit Netzwerken über verschiedene Technologien wie WLAN, Bluetooth und Mobilfunk. Das Betriebssystem verwaltet Netzwerkverbindungen, Datenübertragungen und implementiert Netzwerkprotokolle.

Sicherheit und Datenschutz Das Betriebssystem implementiert Sicherheitsmaßnahmen, um das Gerät und seine Daten vor unautorisiertem Zugriff und Malware zu schützen. Dazu gehören Benutzer-Authentifizierung, Verschlüsselung, App-Berechtigungen und die Ausführung von Apps in isolierten Umgebungen.

Touch-basierte Benutzeroberfläche Es bietet eine grafische Benutzeroberfläche, die für Touchscreens optimiert ist, und verwaltet die Benutzerinteraktion mit dem Gerät durch Gesten, Berührungen und Spracheingaben. Das Betriebssystem sorgt für eine konsistente und intuitive Bedienung über verschiedene Anwendungen hinweg.

Systemdienste und -updates Es bietet grundlegende Dienste wie Uhren, Alarme, Benachrichtigungen und Systemaktualisierungen. Das Betriebssystem ist zudem verantwortlich für das Herunterladen und Installieren von Updates, um neue Funktionen hinzuzufügen, die Leistung zu verbessern und Sicherheitslücken zu schließen.

Management des Stromverbrauchs Das Betriebssystem auf mobilen Geräten versucht, den Stromverbrauch des Geräts z. B. durch Ausschalten bestimmter Prozessorkerne, Ausschalten des Bildschirmes und andere Maßnahmen zu vermindern. Ziel ist eine möglichst lange Akkulaufzeit.

A 8.2 (P1) Öffnen Sie auf einem Rechner mit dem Betriebssystem Windows oder Linux eine Shell.

- Was ist Ihr aktuelles Arbeitsverzeichnis?
- Wechseln Sie mit der Shell zum Wurzelverzeichnis (`cd`) und zeigen dort alle Dateien und Verzeichnisse an (`ls` bzw. `dir`)!
- Erstellen Sie automatisch eine Textdatei, welche den Inhalt des Wurzelverzeichnisses enthält z. B. mit `ls > inhalt.txt`. Lassen Sie sich den Inhalt der entstandenen Textdatei anzeigen. Sie haben vermutlich im Wurzelverzeichnis kein Schreibrecht, wo muss die Textdatei daher liegen?
- Wem gehören die Dateien und Verzeichnisse im Wurzelverzeichnis bzw. in Ihrem Arbeitsverzeichnis? Finden Sie das heraus!
- Beschäftigen Sie sich mit den in Unix bzw. Linux verfügbaren Kommandos `grep`, `find` und `sed`! Was können Sie damit machen?

Lösung

Auf einem Windows-Rechner können Sie eine Shell öffnen, indem Sie `cmd` unten links im Eingabefeld zum Suchen eingeben. Dann startet ein Fenster (genauer das Programm cmd.exe), das auch Eingabeaufforderung, Kommandozeile oder Command Prompt genannt wird. Hier funktionieren noch die Kommandozeilen-Befehle aus der MS-DOS-Zeit, wie im Kapitel über Betriebssysteme beschrieben. Alternativ können Sie auch Powershell eingeben und starten damit eine Kommandozeilen-Schnittstelle, die auf der Common-Language-Runtime aus . NET basiert und neben den MS-DOS Kommandos das Erstellen von Skripten mit einer eigenen Skriptsprache vereinfacht. Auf Linux- und MAC-Systemen wird häufig die BASH als Shell, Kommandozeilen-Schnittstelle verwendet.
Typischerweise befindet man sich auf allen Betriebssystemen nach dem Start des Shellfensters im sog. Home-Directory des jeweils angemeldeten Benutzers. Bei mir ist dies auf Windows beispielsweise `C:\Users\be` und auf Ubuntu Linux ist es `/home/be`.
Eine umfassende Einführung in die Möglichkeiten der jeweiligen Shells würde hier zu weit führen.

A 8.3 (P3) Versuchen Sie auf ihrem Computer Docker-Desktop zu installieren. Starten Sie ein einfaches Image mit Ubuntu-Linux führen Sie darin die `bash` aus. Wiederholen Sie dann die Aufgabe zum Thema Shell.

Lösung

A 8.4 (T2) Beschreiben Sie den Unterschied zwischen einem Mikrokernel und einem monolithischen Kernel. Untersuchen Sie dabei die Rolle der Gerätetreiber. Kann ein Gerätetreiber vollständig außerhalb des Kernels liegen?

Lösung

Monolithischer Kernel

Ein monolithischer Kernel ist eine Betriebssystemarchitektur, bei der der gesamte Kernel und seine Dienste, einschließlich Gerätetreiber, Dateisysteme und Netzwerkprotokolle, in einem einzigen großen Block von

Code laufen. Dieser Block operiert im Kernel-Modus, was ihm volle Kontrolle über das System gibt.
In einem monolithischen System sind Gerätetreiber typischerweise Teil des Kernels selbst. Sie laufen im Kernel-Modus, was ihnen direkten Zugang zur Hardware und zu privilegierten Systemressourcen ermöglicht (Speicher, IO, Prozesse). Diese Integration kann zu besserer Performance führen, da die Kommunikation zwischen Treibern und dem Kernel effizienter ist. Allerdings birgt dies Risiken für die Stabilität und Sicherheit: Ein fehlerhafter Treiber kann das gesamte System zum Absturz bringen oder Sicherheitslücken öffnen.

Mikrokernel

Ein Mikrokernel ist ein Ansatz, bei dem der Kernel auf die Verwaltung der grundlegenden Funktionen beschränkt ist, wie beispielsweise die Kommunikation zwischen Prozessen (IPC) und grundlegende Scheduling-Aufgaben. Andere Systemdienste, einschließlich Gerätetreibern, laufen im Benutzermodus außerhalb des Kerns.
Gerätetreiber laufen typischerweise im Benutzermodus außerhalb des Kerns. Dies erhöht die Stabilität und Sicherheit, da ein fehlerhafter Treiber den Kern nicht direkt beeinträchtigen kann. Stattdessen kommunizieren Treiber über definierte IPC-Mechanismen mit dem Kern. Diese Trennung kann allerdings zu Lasten der Performance gehen, wegen der Kommunikation zwischen dem Kernel und den Treibern. Dennoch muss der Kern natürlich eine grundlegende API für die Hardware bereitstellen.

A 8.5 (P1) Beschäftigen Sie sich mit dem Schlüsselwort `synchronized` in der Programmiersprache Java. Hat eine Methode diesen Modifizierer wird bei ihrem Aufruf durch einen Thread etwas gesperrt, was genau wird gesperrt? Was genau passiert beim Sperren, wenn andere Threads dieselbe Methode aufrufen wollen? Was hat das mit dem in diesem Kapitel vorgestellten Konzept der Semaphore zu tun?

Lösung

Was wird gesperrt?

Wenn eine Methode in Java mit dem synchronized-Modifizierer versehen ist, wird das Objekt, zu dem die Methode gehört (bei einer Instanzmethode) oder die Klasse selbst (bei einer statischen Methode) gesperrt. Dies geschieht für den Zeitraum des Methodenaufrufs durch den ausführenden Thread.
Der erste Thread erhält eine Sperre (auch als Monitor bekannt) auf dem Objekt (Instanz-Methode) oder der Klasse (statische Methode). Wenn ein anderer Thread dieselbe oder eine andere synchronized-Methode desselben Objekts bzw. derselben Klasse aufruft, muss dieser warten, bis die Sperre auf dem Objekt oder der Klasse wieder freigegeben wird. Also der erste Thread die synchronized-Methode wieder verlässt. Achtung: Andere Threads können alle Instanz- bzw. statischen Methoden, die nicht über synchronized geschützt, sind weiter verwenden, denn nur synchronized fordert eine Sperre an, blockiert den eigenen Thread, wenn die Sperre gerade nicht verfügbar ist und gibt sie nach Verlassen der Methode wieder frei.

Beziehung zu Semaphoren

Das Konzept der synchronized-Methoden (Monitore) in Java ist eng verwandt mit dem Konzept der Semaphore. Eine Semaphore ist ein Synchronisationswerkzeug, das die Anzahl der gleichzeitigen Zugriffe auf eine Ressource kontrollieren kann. Es gibt binäre Semaphore (auch bekannt als Mutex für „Mutial Exclusion“) und zählende Semaphore.
Semaphore erlauben eine feinere Kontrolle über den Zugriff, indem explizit angegeben wird, wie viele Threads gleichzeitig auf einen bestimmten Abschnitt des Codes zugreifen dürfen. Semaphor-Implementierungen finden sich eher innerhalb des Betriebssystems, während synchronized (Monitor) ein Konzept aus einer

Programmiersprache ist. Synchronized verwendet die dort vorhandenen Abstraktionen wie Objekte und Klassen und ein Monitor kapselt das Beschaffen und das Freigeben der Sperren für den Programmierer über den synchronized-Modifzierer.

9 Lösungen zu Kapitel 9 – Höhere Programmiersprachen und C

Lösungen zu Kapitel 9.1

A 9.1 (T0) Beantworten Sie die folgenden Fragen:

a) Nennen Sie nach dem Zeitpunkt ihrer Entstehung geordnet vier objektorientierte Programmiersprachen.

b) Nennen Sie die drei ältesten höheren Programmiersprachen.

c) Nennen Sie eine Programmiersprache, in der es keine Laufanweisungen (`FOR` oder ähnliche Anweisungen) gibt.

d) Welche Anforderungen muss eine Programmiersprache erfüllen, damit in dieser Sprache geschriebene Programme „portierbar“ sind?

Lösung

a) Erste OO-Sprachen: Simula (1960er), Smalltalk (1970er), C++ (1980er), Java (1990er)

b) Erste Hochsprachen: Fortran (1953-56), Algol (1958-1960), Cobol (1959 - 60), LISP (1959)

c) Sprachen ohne ’for’: Viele funktionale Sprachen haben keine Konstrukte für Schleifen, sondern empfehlen stattdessen (End-)Rekursion. Weiteres Beispiel ist die Sprache PROLOG aus der Logik-Programmierung.

d) Portierbar sind Sprachen mit Eigenschaften wie

Plattformunabhängigkeit Der geschriebene Code kann auf verschiedenen Betriebssystemen und Hardwareplattformen laufen, ohne dass der Code geändert werden muss. Klassisches Beispiel hierfür ist Java, dessen Programme auf jeder Maschine mit einer Java Virtual Machine (JVM) laufen können.

Standardisierte Sprachspezifikation Ein klar definierter Standard für die Sprachspezifikation hilft dabei, Unterschiede in der Implementierung zwischen verschiedenen Compilern oder Laufzeitumgebungen zu minimieren. Dies gewährleistet, dass Programme konsistent über verschiedene Umgebungen hinweg funktionieren. Ein Beispiel hierfür ist die Sprache C++ mit ihren fortlaufend aktualisierten Standards, aktuell ist Version 20 (`https://isocpp.org/std/the-standard`) an der Version 2023 wird gerade gearbeitet.

Standardbibliotheken Die Bereitstellung von umfangreichen, plattformübergreifenden Standardbibliotheken innerhalb der Sprache ermöglicht es Entwicklern, auf eine gemeinsame Sammlung von Funktionalitäten zuzugreifen, ohne sich auf externe, plattformspezifische Bibliotheken verlassen zu müssen.

Abstraktion von der Hardware Die Sprache sollte eine Abstraktion der der zugrunde liegenden Hardware anbieten z. B. in Form einer Virtuellen Maschine wie in Java, sodass Entwickler sich nicht um hardware- oder betriebssystemspezifische Details kümmern müssen. Dies vereinfacht die Portierung erheblich.

Konsistente Behandlung von Datentypen Um Probleme mit der Portierbarkeit zu vermeiden, sollte die Sprache eine konsistente Größe und Darstellung von Datentypen über verschiedene Plattformen hinweg sicherstellen. Unterschiede in der Behandlung von Datentypen, wie z.B. Ganzzahlen

unterschiedlicher Größe oder verschiedene Byte-Reihenfolgen für Datentypen, die durch mehr als ein Byte repräsentiert werden (Little- und Big-Endian), können sonst zu Fehlern führen.

A 9.2 (L1) Beantworten Sie die folgenden Fragen:

a) Bilden Sie aus Selektionen eine Auswahlanweisung mit vier Ausgängen.

b) Wie wird eine Netzstruktur in der erweiterten D-Struktur abgebildet?

c) Was unterscheidet die D-Struktur von der erweiterten D-Struktur?

d) Lässt sich unter Verwendung der Strukturen Sequenz, Selektion und Iteration für jedes berechenbare Problem ein Algorithmus angeben?

Lösung

A 9.3 (T1) Beantworten Sie die folgenden Fragen:

a) Was ist der Unterschied zwischen Syntax und Semantik?

b) Wodurch unterscheiden sich funktionale und imperative Programmiersprachen?

c) Grenzen Sie die Begriffe Assembler, Compiler und Interpreter ab.

d) Wodurch unterscheiden sich die Zeigerkonzepte in C und Pascal?

Lösung

a) Syntax und Semantik sind zwei grundlegende Aspekte in Sprachen, sowohl in natürlichen Sprachen als auch in Programmiersprachen:

Syntax Die Syntax definiert Regeln und Strukturen, wie Wörter und Symbole kombiniert werden können, um gültige Sätze oder Ausdrücke in einer Sprache zu bilden. In der Programmierung definiert die Syntax, wie Befehle, Funktionen und andere Anweisungen formatiert und organisiert werden müssen, damit der Compiler oder Interpreter sie verarbeiten kann. Syntaxfehler treten auf, wenn der Code nicht den definierten Regeln der Programmiersprache folgt, was zu Kompilierungs- oder Interpretationsfehlern führt.

Semantik Die Semantik definiert die Bedeutung der Ausdrücke, Sätze und Anweisungen, die durch die Syntax korrekt geformt sind. In der Programmierung bestimmt die Semantik, was ein syntaktisch korrekter Ausdruck bewirkt, wenn er ausgeführt wird. Semantische Fehler treten auf, wenn der Code zwar syntaktisch korrekt ist, aber nicht das tut, was der Entwickler beabsichtigt hat, oft weil er die Bedeutung (oder die Konsequenzen) der verwendeten Konstrukte nicht vollständig versteht.

b) Funktionale und imperative Programmiersprachen repräsentieren zwei unterschiedliche Paradigmen oder Ansätze in der Softwareentwicklung, die sich in ihrer Herangehensweise an die Programmierung und Problemlösung unterscheiden.

Imperative Programmiersprachen In imperativen Sprachen wird ein Programm als eine Reihe von Anweisungen verstanden, die der Computer Schritt für Schritt ausführt. Diese Anweisungen ändern den Zustand des Programms durch Zuweisungen, Schleifen oder Verzweigungen. Imperative Sprachen beschreiben wie Aufgaben ausgeführt werden, einschließlich der expliziten Manipulation von Speicherzuständen und der Kontrolle über die Programmablaufstruktur.

Funktionale Programmiersprachen Funktionale Sprachen sind eher deklarativ, sie konzentrieren sich darauf, was berechnet werden soll, anstatt zu beschreiben, wie. Programme sind eine Menge von Funktionen: Funktionen können als Argumente an andere Funktionen übergeben, von Funktionen zurückgegeben und Variablen zugewiesen werden. Dies fördert eine starke Abstraktion und

Wiederverwendbarkeit von Code. Funktionale Programmierung nutzt oft rekursive Funktionen und Konzepte wie Map, Reduce und Filter für Operationen auf Datenstrukturen. In rein funktionalen Sprachen gibt es keine Zustandsänderungen (Mutationen), und Daten sind unveränderlich. Dies erleichtert das Verständnis und die Vorhersage des Verhaltens in parallelen und verteilten Systemen.

c) Assembler, Compiler und Interpreter sind Werkzeuge, um Programme von einer Hochsprache in Maschinensprache zu übersetzen. Sie unterscheiden sich jedoch in der Art und Weise und dem Zeitpunkt, wie und wann sie diese Übersetzungen durchführen.

Assembler Ein Assembler ist ein Programm, das Assemblersprache in diese Maschinensprache übersetzt. Die Assemblersprache ist eine für Menschen lesbare Form der Maschinensprache des jeweiligen Prozessors (ISA-Architektur). Jede Assembleranweisung entspricht einem Maschinenbefehl. Assembler werden verwendet, um hardwarenahe Software zu schreiben oder wenn Kontrolle über die Ausführung des Programms auf der CPU erforderlich ist, wie bei Betriebssystemen oder eingebetteten Systemen.

Compiler Ein Compiler übersetzt Quellcode einer Hochsprache (wie C oder Java) in Maschinensprache oder in den Bytecode einer virtuellen Maschine, bevor das Programm ausgeführt wird. Diese Übersetzung erfolgt in einem Schritt, und das resultierende ausführbare Programm kann dann ohne den Compiler ausgeführt werden.

Interpreter Ein Interpreter führt den Quellcode eines Programms direkt auf der darunterliegenden Hardware aus, indem er diesen Zeile für Zeile oder Block für Block zur Laufzeit interpretiert. Es ist keine separate Übersetzungsphase erforderlich, bevor das Programm ausgeführt wird. Interpreter sind nützlich für Skriptsprachen (wie Python oder Ruby) und Situationen, in denen eine sofortige Programmausführung erwünscht ist. Sie erleichtern auch die Entwicklung, da Änderungen am Code sofort getestet werden können, ohne den Übersetzungsprozess zu durchlaufen.

d) Wodurch unterscheiden sich die Zeigerkonzepte in C und Pascal?

C In C sind Zeiger ein zentrales Konzept, es gibt Zeiger auf Daten und auch Funktionen. Syntaktisch werden sie durch `*` dargestellt, z. B. `int *ptr;` Sie erlauben, direkt auf Speicheradressen zuzugreifen. Zeiger können auf beliebige Datentypen zeigen und können arithmetisch manipuliert werden (Zeigerarithmetik), um auf benachbarte Speicherzellen zuzugreifen. Dies ermöglicht die effiziente Implementierung von Datenstrukturen wie Listen, Bäumen und Graphen. Der direkte Zugriff auf Speicheradressen macht C sehr mächtig, aber auch fehleranfällig, da Zeigerfehler zu schwer zu findenden Fehlern führen können, der Zugriff ist alles andere als typsicher, da die Speicherinhalte durch Casting beliebig interpretiert werden können.

Pascal Zeiger in Pascal sind weniger mächtig und flexibel als in C. Sie werden durch `^` gekennzeichnet, beispielsweise `var ptr: ^Integer;`. Sie können nicht arithmetisch manipuliert werden. Dies macht Pascal sicherer und einfacher zu verwenden, da viele potenzielle Fehlerquellen entfallen, die durch Zeigerarithmetik entstehen können. Pascal bietet auch spezielle Konstrukte wie Referenzen, die sicherstellen, dass der Zugriff auf Speicherbereiche sicher und typsicher ist.

Lösungen zu Kapitel 9.2

A 9.4 (T1) Beantworten Sie die folgenden Fragen:

a) Was versteht man unter einer Metasprache?
b) Erläutern Sie die Aufgabe eines Parsers.
c) Was ist ein Syntaxbaum?
d) Was ist eine Top-Down-Analyse und was ist eine Bottom-Up-Analyse?

Lösung

A 9.5 (L2) Geben Sie die Syntax für eine Gleitpunktzahl als BNF-Produktion, als EBNF-Produktion und als Syntaxgraphen an. Eine Gleitpunktzahl kann eine Ganze Zahl mit oder ohne Exponenten in der in C üblichen Schreibweise sein oder ein Dezimalbruch mit mindestens einer Vorkommastelle und mindestens einer Nachkommastelle. Beispiele: 123, -25, 3.14159, 1E$-$111, -0.21E3.

Lösung

A 9.6 (L3) Geben Sie Syntaxgraphen, BNF- und EBNF-Produktionen für konjunktive Terme und disjunktive Ausdrücke an. Ein disjunktiver Ausdruck ist eine Oder-Verkettung von Variablen, negierten Variablen und eingeklammerten konjunktiven Termen, die ebenfalls auch negiert sein dürfen. Ein konjunktiver Term ist eine Und-Verkettung von Variablen oder negierten Variablen. Variablen können als bekannt vorausgesetzt werden, verwenden Sie dafür das Symbol *var* .
Beispiele für konjunktive Terme: var, $\neg var$, $var \wedge var$, $var \wedge \neg var$, $var \wedge var \wedge var$
Beispiele für disjunktive Ausdrücke:
var, $var \vee (var \wedge var)$, $(var \wedge var) \vee var \vee \neg(var \wedge \neg var \wedge var) \vee \neg var$

Lösung

10 Lösungen zu Kapitel 11 – Automatentheorie und formale Sprachen

Übungsaufgaben zu Kapitel 11.1

A 11.1 (T1) Beantworten Sie die folgenden Fragen:

a) Was versteht man unter dem kartesischen Mengenprodukt?
b) Grenzen Sie die Begriffe Mealy- und Moore-Automat gegeneinander ab.
c) Definieren Sie den Begriff Mächtigkeit im Zusammenhang mit Automaten.
d) Was ist ein Akzeptor? Was ist ein endlicher Übersetzer?

Lösung

a) Das kartesische Mengenprodukt $A \times B$ zweier Mengen A und B ist die Menge aller geordneten Paare der Art (a,b) mit $a \in A$ und $b \in B$. Ist beispielsweise $A = \{x,y\}$ und $B = \{1,2,3\}$, dann ist $A \times B = \{(x,1),(x,2),(x,3),(y,1),(y,2),(y,3)\}$.

b) Hängt die Ausgabe vom Eingabezeichen und vom aktuellen Zustand des Automaten ab, so spricht man von einem *Mealy-Automaten*: $g : Q \times \Sigma \to Y$.

Hängt die Ausgabe nur vom aktuellen Zustand ab, so spricht man von einem *Moore-Automaten*: $g : Q \to Y$.

Trotz der unterschiedlichen Definitionen sind beide Arten äquivalent: Jeder Mealy-Automat lässt sich in einen Moore-Automaten mit gleicher Funktionalität überführen und umgekehrt.

c) Bei einem Automaten mit akzeptierter Sprache L wird die Anzahl der Wörter der Sprache L als deren Mächtigkeit |L| bezeichnet. Ein Sprache mit unendlich vielen Wörtern hat die Mächtigkeit „abzählbar unendlich“. Allgemein bezeichnet man als die Mächtigkeit einer Menge die Anzahl der Elemente der Menge.

d) Automaten mit einer Sprache L sind in der Lage, von einem Wort $w \in \Sigma^*$ zu erkennen, ob es zu L gehört oder nicht. Man lässt dazu die einzelnen Zeichen von w als Eingabezeichen auf den zu Beginn im Anfangszustand q_s befindlichen Automaten wirken. Wenn dieser mit dem letzten Zeichen von w in einen Endzustand gelangt, so ist $w \in$ L, andernfalls nicht. Man bezeichnet derartige Automaten als *erkennende* Automaten oder *Akzeptoren*.

Diese sind von übersetzenden Automaten zu unterscheiden, die ein Eingabewort in ein Ausgabewort transformieren. Dies sind dann Automaten mit Ausgabe (siehe Aufgabe b). Sind Q, Σ und Y des übersetzenden Automaten außerdem endlich, so wird dieser auch als *endlicher Übersetzer* bezeichnet.

A 11.2 (L2) Gegeben sei der Automat mit $\Sigma = \{$a, b, c$\}$ und $Q = \{q_s, q_1, q_2, q_e\}$, dessen Übergangsfunktion durch die Tabelle rechts definiert ist. q_s ist der Anfangs- q_e der Endzustand.

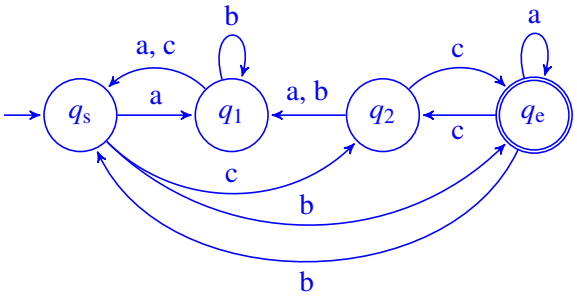
a) Zeichnen Sie den Übergangsgraphen für diesen Automaten.

b) Welche der folgenden Wörter gehören zur akzeptierten Sprache dieses Automaten: abc, a^3bc^3, $a^2b^2c^2$, $a^3b^2c^2$?

σ_i	q_s	q_1	q_2	q_e
a	q_1	q_s	q_1	q_e
b	q_e	q_1	q_1	q_s
c	q_2	q_s	q_e	q_2

Lösung

a) Übergangsdiagramm



b) Der Automat befindet sich im Anfangszustand q_s. Die Zeichen des Wortes abc werden nun von links nach rechts zeichenweise eingegeben. Dabei ergeben sich folgende Übergänge:

$q_s, a \to q_1, \quad q_1, b \to q_1, \quad q_1, c \to q_s$.

Insgesamt findet man also $q_s, abc \to q_s$. Der Endzustand wird offenbar nicht erreicht, dementsprechend gehört abc nicht zur akzeptierten Sprache: abc $\notin$ L.

Für a^3bc^3 findet man: $q_s, a^3bc^3 \to q_e$, also gilt $a^3bc^3 \in L$.

Für $a^2b^2c^2$ findet man: $q_s, a^2b^2c^2 \to q_e$, also gilt $a^2b^2c^2 \in L$.

Für $a^3b^2c^2$ findet man: $q_s, a^3b^2c^2 \to q_2$, also gilt $a^3b^2c^2 \notin L$.

A 11.3 (L2) Konstruieren Sie einen DEA mit $\Sigma = \{a, b\}$, dessen akzeptierte Sprache L aus der Menge aller Worte aus Σ^* besteht, die mit a beginnen und bb nicht als Teilstring enthalten. Geben Sie L in der üblichen Mengenschreibweise an.

Lösung

Diagramm:

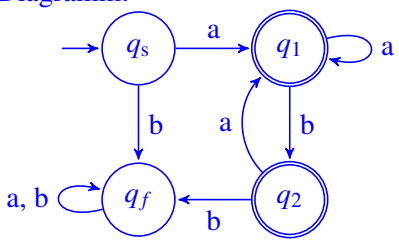


Tabelle:

σ_i	q_s	q_1	q_2	q_f
a	q_1	q_1	q_1	q_f
b	q_f	q_2	q_f	q_f

einfache Lösung: $L = \{ax \mid x \in \Sigma^*, x \text{ enthält nicht } bb\}$.

besser: $L = \{a(a|ba)^*(a|b)^n \mid n \in \{0,1\}\}$

oder $L = \{a(a|ba)^*\} \cup \{a(a|ba)^*(a|b)\}$.

A 11.4 (L3) Geben Sie einen DEA als Übergangstabelle und als Übergangsdiagramm an, der alle aus den Ziffern 1 bis 4 gebildeten natürlichen Zahlen akzeptiert, deren Stellen monoton wachsen. Jede folgende Ziffer ist also größer oder gleich der vorangehenden, ein Beispiel dafür ist etwa die Zahl 112444.

Lösung

Diagramm (unvollständiger Automat, Fangzustand nicht eingezeichnet):

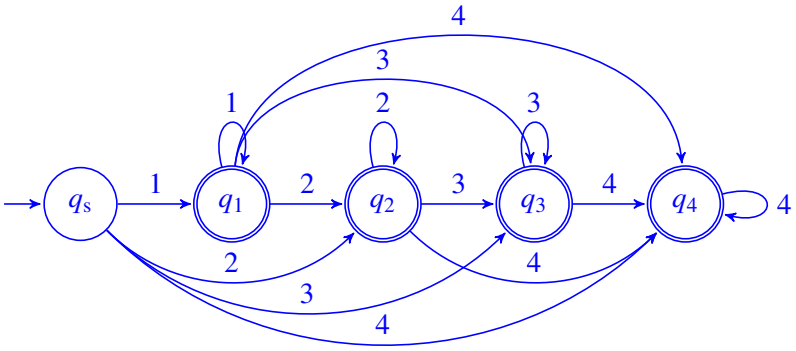


Tabelle:

σ_i	q_s	q_1	q_2	q_3	q_4	q_f
1	q_1	q_1	q_f	q_f	q_f	q_f
2	q_2	q_2	q_2	q_f	q_f	q_f
3	q_3	q_3	q_3	q_3	q_f	q_f
4	q_4	q_4	q_4	q_4	q_4	q_f

Die Zustände q_1 bis q_4 sind alle Endzustände.

A 11.5 (L3) Gegeben sei der Automat $A(\Sigma, Q, \delta)$ mit $\Sigma = \{a, b\}$, $Q = \{q_s, q_1, q_e\}$, Anfangszustand q_s, Endzustand q_e und δ definiert durch die unten stehende Übergangstabelle.

σ_i	q_s	q_1	q_e
a	q_s, q_e	q_e	q_1
b	–	q_e	q_e

a) Wie interpretieren Sie, dass in der Übergangstabelle in der Spalte für q_s für die Eingabe a zwei Folgezustände q_s, q_e eingetragen sind?

b) Zeichnen Sie das zum Automaten A gehörige Übergangsdiagramm.

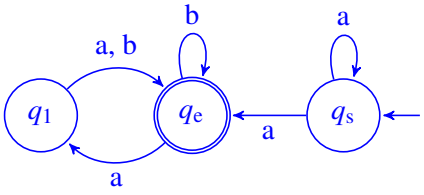
c) Geben Sie die akzeptierte Sprache L(A) von A als regulären Ausdruck an.

d) Konstruieren Sie einen zu A äquivalenten deterministischen Automaten und zeichnen Sie das zugehörige Übergangsdiagramm.

Lösung

a) Es handelt sich um einen nichtdeterministischen Automaten, da vom Zustand q_s mit dem Eingabezeichen a zwei Folgezustände erreicht werden können, nämlich q_s und q_e. Die Übergangstabelle beschreibt also keine eindeutige Funktion sondern eine Relation.

b) Übergangsdiagramm



c) $L(A) = a^+(b^*(a(a \mid b))^*)^*$

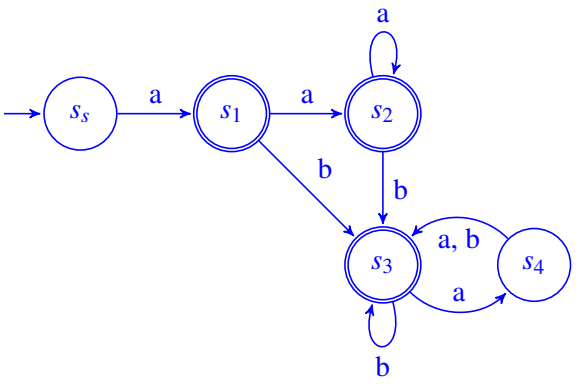
d) Konstruktion eines äquivalenten DEA:

σ_i	$\{q_s\}$	$\{q_s, q_e\}$	$\{q_s, q_e, q_1\}$	$\{q_e\}$	$\{q_1\}$
a	$\{q_s, q_e\}$	$\{q_s, q_e, q_1\}$	$\{q_s, q_e, q_1\}$	$\{q_1\}$	$\{q_e\}$
b	–	$\{q_e\}$	$\{q_e\}$	$\{q_e\}$	$\{q_e\}$

Umbenennung der Zustände (übersichtlicher):

σ_i	s_s	s_1	s_2	s_3	s_4
a	s_1	s_2	s_2	s_4	s_3
b	–	s_3	s_3	s_3	s_3

Startzustand ist s_s, Endzustände sind s_1, s_2, s_3. Das Diagramm sind dann wie folgt aus:



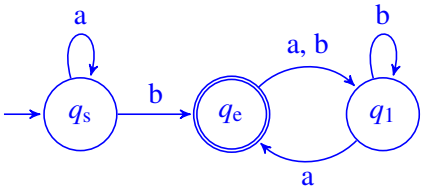
A 11.6 (L2) Gegeben sei der Automat A$(Q, \Sigma, \delta, q_s, q_e)$ mit $\Sigma = \{a, b\}$, $Q = \{q_s, q_1, q_e\}$ und der durch die nebenstehende Tabelle definierten Zustandsübergangsfunktion.

σ_i	q_s	q_1	q_e
a	q_s	q_e	q_1
b	q_e	q_1	q_1

a) Zeichnen Sie das zu A gehörige Übergangsdiagramm.
b) Wie lautet die durch A akzeptierte Sprache?

Lösung

a) Übergangsdiagramm

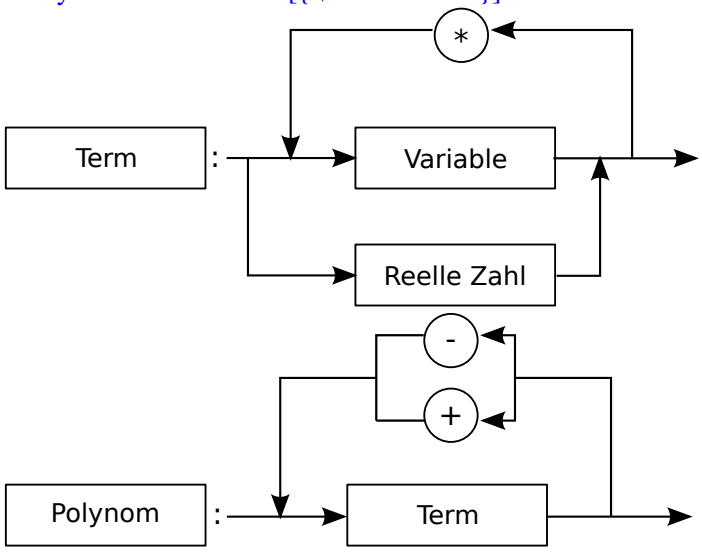


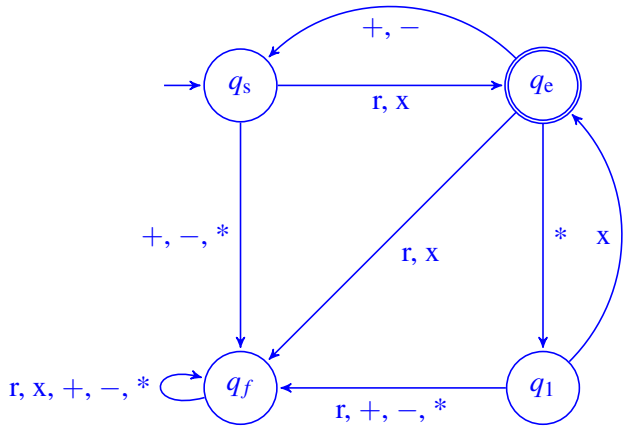
b) L(A) $= a^*b((a \mid b)b^*a)^*$

A 11.7 (L3) Beschreiben Sie ein Polynom als BNF-Produktion, als Syntaxgraph und als Automat. Ein Polynom besteht aus durch die Operatoren $+$ und $-$ verknüpften Termen. Ein Term besteht aus einer optionalen reellen Zahl, optional multipliziert mit einer beliebig langen Folge von multiplikativ verknüpften Variablen x. Ein Term darf nicht leer sein.
Beispiele für Terme: 3,5, x, $-5*x*x$, $x*x*x$.
Beispiel für ein Polynom: $2+4*x+x*x*x-3{,}5*x*x$.
Die syntaktischen Variablen <Reelle Zahl> und <Variable> dürfen für BNF-Produktionen und Syntaxgraphen als bekannt vorausgesetzt werden. Das Alphabet des Automaten sei $\Sigma = \{r, x, *, +, -\}$, wobei r für eine reelle Zahl und x für eine Variable steht.

Lösung

<Term> ::= <Reelle Zahl> | <Variable> [{*<Variable>}]
<Polynom> ::= <Term> [{+ | − <Term>}]





A 11.8 (T1) Beantworten Sie die folgenden Fragen:

a) Was haben Turing-Maschinen mit Berechenbarkeit zu tun?

b) Woran ist Alan Turing gestorben?

c) Was ist ein linear beschränkter Automat?

d) Grenzen Sie die Begriffe Automat, Kellerautomat und Turing-Maschine gegeneinander ab.

Lösung

a) Eine Funktion $f(x) = y$ mit $x, y \in \Sigma^*$ ist Turing-berechenbar, wenn es eine Turing-Maschine gibt, welche die auf dem Band gespeicherte Eingabe x in die Ausgabe y transformiert, die dann auf dem Band abgelesen werden kann. Turing-Maschinen sind Modelle für einen abstrakten Computer: alles was man prinzipiell mit einer Turing-Maschine berechnen kann, kann man auch mit einem Computer berechnen und umgekehrt.

b) Alan Turing ist vermutlich durch Selbstmord durch einen mit Cyanid vergifteten Apfel gestorben. Dies steht wohl in Zusammenhang mit einer Hormonbehandlung, der er sich wegen seiner offenkundig gewordenen Homosexualität (die damals strafbar war) unterziehen musste.

c) Bei linear beschränkten Automaten handelt es sich um Turing-Maschinen, bei denen nur ein durch die Länge des Eingabewortes beschränkter Bereich des Bandes verwendet wird. Linear beschränkte Automaten sind weniger mächtig als Turing-Maschinen. Auch bei linear beschränkten Automaten kann man deterministische und nichtdeterministische Varianten betrachten. Ob nichtdeterministische linear beschränkten Automaten äquivalent zu deterministischen sind, ist nicht bekannt.

   Die durch nichtdeterministische linear beschränkte Automaten akzeptierten Sprachen sind zu den in Kap. 11.2 eingeführten kontextsensitiven Sprachen (Chomsky Typ 1) äquivalent.

d) Ein Automat hat außer den internen Zuständen keinen Speicher, er erhält von außen Eingabezeichen.

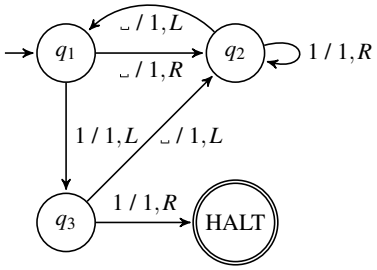
   Ein Kellerautomat hat einen einseitig unbegrenzten Speicher, der als Stack für Kellerzeichen verwendet wird, d. h. Zeichen dürfen nur oben auf den Stack gelegt oder oben von ihm entnommen werden. Ein Kellerautomat enthält ebenfalls von außen Eingabezeichen.

   Eine Turing-Maschine hat ein beidseitig unbegrenztes lineares Speicherband, auf dem vor Start der Turing-Maschine die Eingabezeichen vorhanden sein müssen. Die Turing-Maschine kann dann die Zeichen vom Band lesen, Zeichen auf das Band schreiben und schrittweise nach rechts und links den Schreib-/Lesekopf auf dem Band bewegen.

A 11.9 (L2) Es sei die nebenstehende Turing-Maschine mit den Bandzeichen $\{␣, 1\}$ als Übergangsdiagramm gegeben.

a) Geben Sie das dazugehörige tabellarische Turing-Programm an.

b) Der Schreib-/Lesekopf stehe auf einem mit Blank vorbesetzten Band. Was bewirkt diese Turing-Maschine?



Lösung

a)

$$1\begin{cases} ␣\ 1\ R\ 2 \\ 1\ 1\ L\ 3 \end{cases} \quad 2\begin{cases} ␣\ 1\ L\ 1 \\ 1\ 1\ R\ 2 \end{cases} \quad 3\begin{cases} ␣\ 1\ L\ \ 2 \\ 1\ 1\ R\ \text{HALT} \end{cases}$$

b) Diese Turing-Maschine schreibt 6 1en auf ein anfänglich mit Blank vorbesetztes Band. Sie stoppt nach 13 Schritten unter der zweiten 1 von rechts.

Es handelt sich hierbei um die Busy-Beaver-Funktion $B(3)$, d. h. um diejenige aus genau drei Anweisungen bestehende Turing-Maschine, welche die größtmögliche Anzahl von aufeinander folgenden 1en auf ein anfänglich leeres Band schreibt und danach anhält.

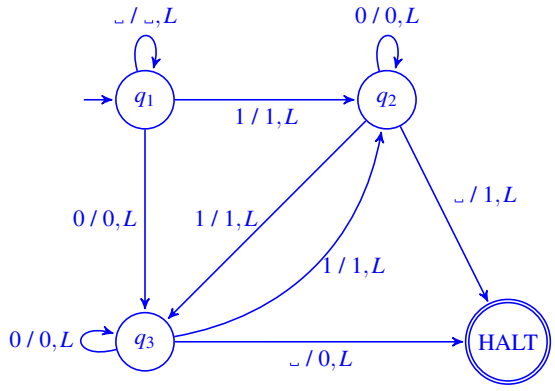
A 11.10 (L3) Konstruieren Sie eine Turing-Maschine mit den Bandzeichen $\Gamma = \{␣, 0, 1\}$, welche für eine zusammenhängende aus Nullen und Einsen bestehende Zeichenfolge auf einem mit ␣ vorbesetztem Band die Anzahl der Einsen auf gerade Parität ergänzt. Dazu wird am linken Ende der Zeichenfolge eine 0 angefügt, wenn die Anzahl der Einsen gerade ist und eine 1, wenn die Anzahl der Einsen ungerade ist. Der Schreib-/Lesekopf soll vor der Operation rechts neben der Zeichenfolge stehen. Beispiel: aus ␣␣␣␣10101␣␣␣␣ wird also ␣␣␣110101␣␣␣␣ und aus ␣␣␣␣1001␣␣␣␣ wird ␣␣␣01001␣␣␣␣.

Lösung

Strategie:

1. In Zustand 1 rechts neben dem String starten. In Zustand 1 nach links gehen, solange ␣ gelesen wird. Wird in Zustand 1 eine 0 gelesen, nach Zustand 3 gehen. (Erstes Stringzeichen ist 0) Wird in Zustand 1 eine 1 gelesen, nach Zustand 2 gehen. (Erstes Stringzeichen ist 1)
2. In Zustand 2 solange nach links gehen, wie 0en gelesen werden. Wird in Zustand 2 eine 1 gelesen, nach Zustand 3 gehen. Wird in Zustand 2 ein ␣ gelesen, ist das Stringende erreicht, eine 1 schreiben, HALT.
3. In Zustand 3 solange nach links gehen, wie 0en gelesen werden. Wird in Zustand 3 eine 1 gelesen, nach Zustand 2 gehen. Wird in Zustand 3 ein ␣ gelesen, ist das Stringende erreicht, eine 0 schreiben, HALT. Ist die Turing-Maschine in Zustand 2, so ist die aktuelle Anzahl der 1en ungerade, ist sie in Zustand 3, so ist die Anzahl der 1en gerade.

$$1\begin{cases} ␣\ ␣\ L\ 1 \\ 0\ 0\ L\ 3 \\ 1\ 1\ L\ 2 \end{cases} \quad 2\begin{cases} ␣\ 1\ L\ \text{HALT} \\ 0\ 0\ L\ \ 2 \\ 1\ 1\ L\ \ 3 \end{cases} \quad 3\begin{cases} ␣\ 0\ L\ \text{HALT} \\ 0\ 0\ L\ \ 3 \\ 1\ 1\ L\ \ 2 \end{cases}$$



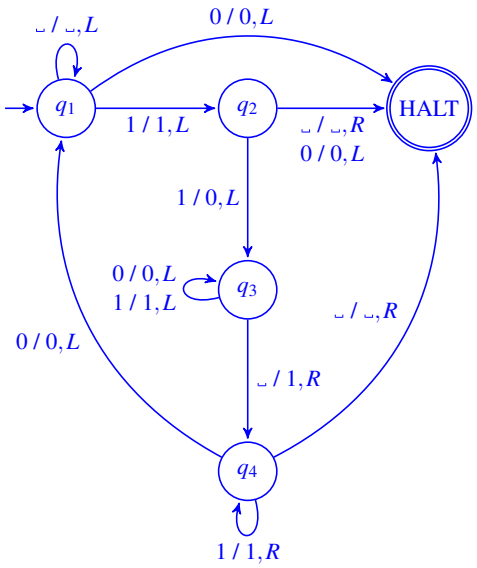
A 11.11 (L3) Konstruieren Sie eine Turing-Maschine mit den Bandzeichen $\Gamma = \{␣, 0, 1\}$, die in einer zusammenhängenden Gruppe von Einsen auf einem mit Blank vorbesetzten Band zwischen je zwei Einsen eine Null einfügt. Zu Beginn soll der Schreib-/Lesekopf rechts von der Einsergruppe stehen. Beispiel: aus ␣␣␣␣11111␣␣␣␣ wird also ␣␣␣␣101010101␣␣␣␣.

Lösung

Strategie:

1. In Zustand 1 rechts neben dem String starten. In Zustand 1 nach links gehen, bis zur ersten 1, also zum Stringanfang. Wurde die erste 1 gelesen, nach Zustand 2 gehen.
2. In Zustand 2 einen Schritt nach links gehen. Wird dort eine 1 gelesen, diese durch 0 ersetzen und nach Zustand 3 gehen. Wird ein ␣ oder eine 0 gelesen, HALT.
3. In Zustand 3 solange nach links gehen, bis ein ␣ gelesen wird. Den ␣ am Stringende durch eine 1 ersetzen und nach Zustand 4 gehen.
4. In Zustand 4 nach rechts gehen bis eine 0 gelesen wird. Jetzt weiter bei Punkt 1 in Zustand 1.

$$1\begin{cases} ␣\ ␣\ L & 1 \\ 0\ 0\ L & \text{HALT} \\ 1\ 1\ L & 2 \end{cases} \quad 2\begin{cases} ␣\ ␣\ R & \text{HALT} \\ 0\ 0\ L & \text{HALT} \\ 1\ 0\ L & 3 \end{cases} \quad 3\begin{cases} ␣\ 1\ R & 4 \\ 0\ 0\ L & 3 \\ 1\ 1\ L & 3 \end{cases} \quad 4\begin{cases} ␣\ ␣\ R & \text{HALT} \\ 0\ 0\ L & 1 \\ 1\ 1\ R & 4 \end{cases}$$



Übungsaufgaben zu Kapitel 11.2

A 11.12 (T1) Beantworten Sie die folgenden Fragen:

a) Wie kann man endliche von zyklischen Sackgassen unterscheiden?

b) Was ist ein Palindrom?

c) Was versteht man unter einer kontextfreien Sprache?

d) Was ist das Wortproblem?

Lösung

a) Kommt man bei der Analyse eines Wortes nach endlich vielen Schritten zu einem Wort, auf das keine Produktionen mehr angewendet werden können, so ist man in eine endliche Sackgasse geraten. Man kann dann die Analyse bei einer vorangehenden Verzweigungsmöglichkeit wieder aufnehmen. Bei einer zyklischen Sackgasse führt die Analyse nach einer endlichen Anzahl von Schritten wieder auf ein Wort, das in einem früheren Schritt bereits aufgetreten ist. Man kann die Sackgasse erkennen und wieder verlassen, wenn man alle Analyseschritte zwischenspeichert und immer wieder mit dem aktuellen Schritt vergleicht.

b) Unter einem Palindrom versteht man ein um seinen Mittelpunkt symmetrisches Wort. Es ist also vorwärts und rückwärts gelesen identisch. Beispiel: otto.

c) Eine *Grammatik* heißt Chomsky-2-Grammatik oder kontextfreie Grammatik wenn die Produktionen nicht von einem Kontext abhängen. Alle Produktionen haben dann die Form:

$$A \to u \quad \text{mit } A \in V, u \in (V \cup \Sigma)^+ \quad .$$

Die Menge der durch kontextfreie Grammatiken erzeugten Sprachen ist mit der Menge der durch nichtdeterministische Kellerautomaten akzeptierten Sprachen identisch.

Eine *Sprache* L ist kontextfrei, wenn eine kontextfreie Grammatik existiert, die die Sprache L erzeugt.

d) Unter dem Wortproblem versteht man die Aufgabe, von einem gegebenen Wort zu entscheiden, ob es zu einer bestimmten Sprache gehört oder nicht. Es kann sich dabei beispielsweise um die von einer formalen Grammatik erzeugte Sprache oder um die akzeptierte Sprache eines Automaten handeln.

A 11.13 (T1) Beantworten Sie die folgenden Fragen:

a) Was versteht man unter dem Nachbereich einer formalen Sprache?

b) Unter welcher Bedingung heißt eine Produktion terminal?

c) Was versteht man unter dem Kern einer formalen Sprache?

d) Welche Klasse von formalen Sprachen der Chomsky-Hierarchie lässt sich durch endliche Automaten darstellen?

e) Welcher Typ von formalen Sprachen ist zu nichtdeterministischen Kellerautomaten äquivalent?

Lösung

a) Als *Nachbereich* bezeichnet man alle überhaupt aus dem Startsymbol S ableitbaren Wörter aus $(V \cup \Sigma)^*$. Er umfasst also auch Wörter, die nicht zur von der Grammatik G erzeugten Sprache L(G) gehören.

b) Eine Produktion wird als *terminal* bezeichnet, wenn das Ergebnis nur aus terminalen Zeichen besteht, also:

$$A \to u \quad \text{mit } A \in V, u \in \Sigma^+ \quad .$$

c) Die Gesamtheit aller in den Ableitungen der Wörter der Sprache vorkommenden Wörter aus $(V \cup \Sigma)^*$ bilden den *Kern K* der formalen Sprache. Offenbar gilt $\mathrm{L(G)} = K \cap \Sigma^*$.

Zusätzlich zur Sprache umfasst der Kern also auch Worte, in denen auch syntaktische Variablen (Nichtterminalsymbole) enthalten sind.

d) Reguläre Sprachen (Typ 3), also solche mit terminalen und linkslinearen oder rechtslinearen Produktionen, lassen sich durch endliche Automaten darstellen.

e) Kontextfreie Sprachen, also durch Chomsky-2-Grammatiken erzeugte, sind zur akzeptierten Sprache von nichtdeterministischen Kellerautomaten äquivalent.

A 11.14 (L3) Gegeben sei die folgende formale Sprache:
$\Sigma = \{\mathrm{u,v,w}\}, V = \{S,X\}, P = \{S \to \mathrm{u}S\mathrm{v}|X, X \to \mathrm{v}X\mathrm{u}|\mathrm{v}|\mathrm{w}\}$

a) Von welchem Typ ist die Grammatik?

b) Geben Sie die zugehörige Sprache an.

c) Leiten Sie das Wort $\mathrm{uv^2wu^2v}$ ab. Falls es bei der Ableitung Sackgassen gibt, geben Sie bitte ein Beispiel an.

Lösung

a) Die Grammatik ist wortlängenmonoton, da keine Produktion verkürzend wirkt. Sie ist außerdem kontextfrei und daher vom Typ Chomsky-2. Da es nichtlineare Produktionen gibt, ist die Grammatik nicht vom Typ Chomsky-3.

b) Die zugehörige Sprache lautet: $\mathrm{L} = \{u^n v^m x u^m v^n \mid n,m \in \mathbb{N}_0, x \in \{v,w\}\}$.
Die kürzesten Wörter sind: v, w, uwv, uvv, vvu, vwu, uvwuv, uvvuv.

c) $S \Rightarrow uSv \Rightarrow uXv \Rightarrow uvXuv \Rightarrow uvvXuuv = uv^2wu^2v$.
Es gibt bei der Ableitung von Wörtern (also vorwärts ausgehend vom Startsymbol) keine Sackgassen.

A 11.15 (L3) Konstruieren Sie eine Formale Sprache L für die Menge aller korrekten arithmetischen Ausdrücke mit natürlichen Zahlen *n* unter Verwendung der üblichen Klammerung mit den Operationen „+“ und „·“.

Lösung

$\Sigma = \{0,1,2,3,4,5,6,7,8,9,+,\cdot,(,)\}$,
$V = \{S,N,T\}$,
$P = \{S \to T, T \to T+T, T \to T \cdot T, T \to (T), T \to N, N \to NN, N \to 0|1|\ldots|9\}$.
Anmerkung: Die beiden Operationen Multiplikation und Addition sind hier gleichberechtigt, „Punkt vor Strich“ wird also nicht beachtet. Zum Erzeugen von arithmetischen Ausdrücken ist dies auch nicht erforderlich; für eine Analyse in einem Compiler allerdings schon, hierfür müsste die Grammatik angepasst werden.

A 11.16 (L3) Die Zusammenstellung eines Intercity-Zuges möge nach folgenden Regeln erfolgen: Der erste Wagen des Zugs ist ein Triebwagen, es folgen $n \geq 1$ Wagen der ersten Klasse, danach folgt ein Speisewagen und danach $2n$ Wagen der zweiten Klasse.

a) Geben Sie die Wagenfolge des kürzestmöglichen Zuges an.

b) Konstruieren Sie eine formale Sprache für die Zusammenstellung von Intercity-Zügen. Verwenden Sie dazu die Menge $\Sigma = \{t, 1, s, 2\}$ von terminalen Zeichen in ihrer offensichtlichen Bedeutung sowie die Menge $V = \{S, W\}$ von syntaktischen Variablen, wobei W für „Wagen“ steht.

c) Von welchem Chomsky-Typ ist die von Ihnen definierte Grammatik? Von welchem die Sprache? Bitte begründen Sie Ihre Antwort.

d) Wie muss man die Regeln für die Zugzusammenstellung ändern, damit die entsprechende formale Sprache als endlicher Automat darstellbar ist?

Lösung

a) Die Wagenfolge des kürzestmöglichen Zuges lautet: $t1s22$.

b) $\Sigma = \{t, 1, s, 2\}$,
$V = \{S, W\}$,
$P = \{S \to tW, W \to 1s22, W \to 1W22\}$.

c) Nicht alle verwendeten Produktionen sind terminal oder rechts-/linkslinear, daher ist die Grammatik nicht regulär (Typ 3). Die Grammatik ist wortlängenmonoton und kontextfrei und daher vom Typ 2.

Die Sprache ist daher ebenfalls mindestens Typ 2, könnte aber auch Typ 3 sein. Voraussetzung wäre, dass man eine äquivalente Typ 3 Grammatik angeben könnte. Dies ist hier nicht möglich (siehe dazu auch die folgende Aufgabe). Ein Beweis kann durch das Pumping Theorem erfolgen.

d) Endliche Automaten akzeptieren Typ 3 (reguläre) Sprachen. Da die Sprache hier Typ 2 ist, kann sie nicht als endlicher Automat modelliert werden.

Problematisch ist, dass die Anzahl der Wagen zweiter Klasse doppelt so hoch sein soll wie die der ersten Klasse und dass die Anzahl der Wagen nicht beschränkt ist. Beschränkt man die maximale Anzahl der Wagen, so ist eine Darstellung als endlicher Automat möglich.

Auch ohne Beschränkung gelingt dies, wenn die Anzahl der Wagen erster Klasse unabhängig von der Anzahl der Wagen zweiter Klasse ist.

Man kann zwar eine Mengen von Produktionen finden, bei der alle Produktionen terminal oder linear sind, beispielsweise
$P = \{S \to tW, W \to 1s22, W \to V22, V \to 1W\}$.
Es treten jedoch linkslineare *und* rechtslineare Produktionen gleichzeitig auf, so dass die resultierende Grammatik nicht regulär ist.

Man könnte die Aufgabe jedoch ohne weiteres mit Hilfe eines Kellerautomaten lösen.

11 Lösungen zu Kapitel 12 – Algorithmen – Berechenbarkeit und Komplexität

A 12.1 (T1) Beantworten Sie die folgenden Fragen:

a) Wann ist ein Algorithmus statisch finit und wann dynamisch finit?

b) Erläutern Sie den Unterschied zwischen Berechenbarkeit und Komplexität.

c) Was besagt die Church-Turing-These?

d) Was ist der Unterschied zwischen primitiv-rekursiven und μ-rekursiven Funktionen?

e) Was sind NP-vollständige Probleme?

f) Wann ist ein Algorithmus effektiv, wann effizient?

Lösung

a) Ein Algorithmus muss immer statisch finit und dynamisch finit sein.

statisch finit: Es gibt nur eine endliche Anzahl an Aktionen, die man in einem Algorithmus ausführen kann. Statisch bedeutet hier, dass der Algorithmus nicht laufen muss, um dies zu prüfen.

dynamisch finit: Ressourcen (Speicher, Rechenzeit) sind endlich, daher muss ein Algorithmus mit endlichen Ressourcen auskommen.

b) Im Grunde hat das eine mit dem anderen nicht wirklich etwas zu tun.

Berechenbarkeit: Hier geht es um die Frage, ob ein Problem grundsätzlich berechenbar, also als Algorithmus formulierbar ist. Nur wenn dies der Fall ist, kann man das Problem auf einem Computer überhaupt lösen.

Komplexität: Für berechenbare Probleme ist die Zeit- oder Speicherkomplexität interessant. Eine zentrale Aufgabe der Informatik im Zusammenhang mit der Komplexitätstheorie ist es, nicht nur irgendeinen Algorithmus zur Lösung eines Problems zu finden, sondern einen möglichst effizienten, wobei man die Effizienz vor allem an der benötigten Ausführungszeit und am Speicherbedarf misst.

Die Komplexitätstheorie stellt Mittel bereit, um solche Untersuchungen unabhängig von konkreter Hardware nur anhand der Algorithmen durchzuführen.

c) siehe Buch, Kap. 11.1.1, S. 418

d) Bei den μ-rekursiven Funktionen wird zusätzlich noch der μ-Operator eingeführt (siehe Buch, Kap. 12.1.5, S. 532). Sie sind identisch mit den WHILE-berechenbaren Funktionen, und damit nach der Church-Turing These mit dem Berechenbarkeitsbegriff überhaupt.

Die primitiv rekursiven Funktionen sind eine echte Untermenge, äquivalent zu den LOOP-berechenbaren Funktionen.

e) siehe Buch, Kap. 12.2.4, S. 547.

f) Effektivität bedeutet nur, dass einzelne Anweisungen grundlegend genug sind, so dass sie prinzipiell exakt und in endlicher Zeit ausgeführt werden können. Davon abzugrenzen ist die *Effizienz* bzgl. Zeit und Speicherplatz, die praktisch natürlich eine Rolle spielt, aber keine grundlegende Anforderung für einen Algorithmus ist.

Ein Problem heißt *effizient lösbar*, wenn es einen Algorithmus mit Zeitkomplexität $\mathcal{O}(p(n))$ gibt, wobei $p(n)$ ein Polynom beliebigen Grades ist, der Algorithmus hat polynomielle Laufzeit. Das Problem liegt dann in der Komplexitätsklasse P.

A 12.2 (T1) Beantworten Sie die folgenden Fragen:

a) Was ist eine rekursive Relation?

b) Was sind probabilistische Algorithmen?

c) Erläutern Sie das Prinzip des Backtracking.

d) Erläutern Sie das Prinzip Teile und Herrsche.

Lösung

a) Rekursion = eine Funktion/Relation ist dadurch definiert, dass sie sich selbst mit anderen Parametern direkt oder indirekt aufruft.

b) siehe Buch, Kap. 12.3, S. 555.

c) siehe Buch, Kap. 12.4.3, S. 571.

d) siehe Buch, Kap. 12.2.2, S. 543.

A 12.3 (M3) Zur Ackermannfunktion $A(x,y)$:

a) Wie ist die Ackermannfunktion definiert?

b) Ist die Ackermannfunktion primitiv-rekursiv?

c) Berechnen Sie $A(3,2)$.

d) Für welches $x \in \mathbb{N}$ gilt $A(3,A(0,y)) = A(x,A(3,y))$?

e) Zeigen Sie: $A(p,q+1) > A(p,q)$.

Lösung

a) Die Ackermannfunktion $A(x,y)$ ist wie folgt definiert:

$$\begin{aligned} A(0,y) &= y+1, \\ A(x+1,0) &= A(x,1), \\ A(x+1,y+1) &= A(x,A(x+1,y)). \end{aligned}$$

b) Nein. Die Ackermann-Funktion ist keine primitiv-rekursive Funktion, sondern eine μ-rekursive Funktion, da man zeigen kann, dass sie schneller wächst als jede primitiv rekursive Funktion.

Man kann auch als Begründung angeben, dass man bei der iterativen Programmierung der Ackermann-Funktion nicht mit einer klassischen FOR-Schleife auskommt. Man benötigt eine GOTO- oder eine WHILE-Schleife.

c) m Folgenden wird statt $A(a,b)$ vereinfachend (a,b) geschrieben. Nach dem Aufschreiben der ersten Terme erkennt man das Bildungsgesetz, so dass sich die Lösung schnell finden lässt. Siehe auch die

folgende Teilaufgabe.

$$
\begin{aligned}
&(3,2) = (2,(3,1)) = (2,(2,(3,0))) = (2,(2,(2,1))) = \\
&(2,(2,(1,(2,0)))) = (2,(2,(1,(1,1)))) = (2,(2,(1,(0,(1,0))))) = \\
&(2,(2,(1,(0,(0,1))))) = (2,(2,(1,(0,2)))) = (2,(2,(1,3))) = \\
&(2,(2,(0,(1,2)))) = (2,(2,(0,(0,(1,1))))) = (2,(2,(0,(0,(0,(1,0)))))) = \\
&(2,(2,(0,(0,(0,2))))) = (2,(2,(0,(0,3)))) = (2,(2,(0,4))) = (2,(2,5)) = \\
&(2,(1,(2,4))) = (2,(1,(1,(2,3)))) = (2,(1,(1,(1,(2,2))))) = (2,(1,(1,(1,(1,(2,1)))))) = \\
&(2,(1,(1,(1,(1,(1,(2,0))))))) = \\
&(2,(1,(1,(1,(1,(1,(1,1))))))) = \\
&(2,(1,(1,(1,(1,(1,(0,(1,0)))))))) = \\
&(2,(1,(1,(1,(1,(1,(0,(0,1)))))))) = \\
&(2,(1,(1,(1,(1,(1,(0,2)))))))) = \\
&(2,(1,(1,(1,(1,(1,3)))))) = \\
&(2,(1,(1,(1,(1,(0,(1,2)))))))) = \\
&(2,(1,(1,(1,(1,(0,(0,(1,1)))))))) = \\
&(2,(1,(1,(1,(1,(0,(0,(0,(1,0))))))))) = \\
&(2,(1,(1,(1,(1,(0,(0,(0,(0,1))))))))) = \\
&(2,(1,(1,(1,(1,(0,(0,(0,2)))))))) = \\
&(2,(1,(1,(1,(1,(0,(0,3)))))))) = \\
&(2,(1,(1,(1,(1,(0,4)))))) = \\
&(2,(1,(1,(1,(1,5))))) = \\
&(2,(1,(1,(1,(0,(1,4)))))) = \\
&(2,(1,(1,(1,(0,(0,(1,3)))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(1,2)))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(0,(1,1))))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(0,(0,(1,0)))))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(0,(0,(0,1)))))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(0,(0,2))))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(0,(0,3)))))))) = \\
&(2,(1,(1,(1,(0,(0,(0,(0,4)))))))) = \\
&(2,(1,(1,(1,(0,(0,5)))))) = \\
&(2,(1,(1,(1,(0,6))))) = \\
&(2,(1,(1,(1,7)))) = \\
&(2,(1,(1,(0,(1,6))))) =
\end{aligned}
$$

...
(2,(1,(1,(0,(0,(0,(0,(0,(0,(0,(0,(1,0)))))))))))) =
(2,(1,(1,(0,(0,(0,(0,(0,(0,(0,(0,(0,1)))))))))))) =
...
(2,(1,(1,(0,8)))) =
(2,(1,(1,9))) =
(2,(1,(0,(1,8)))) =
...
(2,(1,(0,(0,(0,(0,(0,(0,(0,(0,(0,(1,0)))))))))))) =
(2,(1,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,1)))))))))))) =
...
(2,(1,(0,10))) =
(2,(1,11)) =
(2,(0,(1,10))) =
...
(2,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(1,0)))))))))))) =
(2,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,1)))))))))))) =
...

(2,(0,12)) =
(2,13) =
(1,(2,12)) =
(1,(1,(2,11))) =
(1,(1,(1,(2,10)))) =
(1,(1,(1,(1,(2,9))))) =
(1,(1,(1,(1,(1,(2,8)))))) =
(1,(1,(1,(1,(1,(1,(2,7))))))) =
(1,(1,(1,(1,(1,(1,(1,(2,6)))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(2,5))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(2,4)))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(2,3))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(2,2)))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(2,1))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(2,0))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,1))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(1,0))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(0,1))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,2))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,3))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(1,2))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(0,(1,1))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(0,(0,(1,0))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(0,(0,(0,1))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(0,(0,2))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,(0,3))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(0,4))))))))))))) =
(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,5))))))))))) =

$$
\begin{aligned}
&\ldots\\
&(1,(1,(1,(1,(1,(1,(1,(1,(1,(1,7)))))))))) =\\
&\ldots\\
&(1,(1,(1,(1,(1,(1,(1,(1,(1,9))))))))) =\\
&\ldots\\
&(1,(1,(1,(1,(1,(1,(1,(1,(1,11))))))))) =\\
&\ldots\\
&(1,(1,(1,(1,(1,(1,(1,(1,13))))))))) =\\
&\ldots\\
&(1,(1,(1,(1,(1,(1,(1,15))))))) =\\
&\ldots\\
&(1,(1,(1,(1,(1,(1,17)))))) =\\
&\ldots\\
&(1,(1,(1,(1,(1,19))))) =\\
&\ldots\\
&(1,(1,(1,(1,21)))) =\\
&\ldots\\
&(1,(1,(1,23))) =\\
&\ldots\\
&(1,(1,25)) =\\
&\ldots\\
&(1,27) =\\
&\ldots\\
&29
\end{aligned}
$$

d) $A(3,A(0,y)) = A(3,y+1) = A(2+1,y+1) = A(2,A(3,y))$. Es ist also $x = 2$.

e) Es gilt:

$$q < A(p,q) \quad \text{für alle } p > 0, q > 1 \quad (*)$$

$A(p,q) < A(p,q+1)$ ist zu zeigen:

$p = 0$:

$$A(0,q) = 2q < 2(q+1) = A(0,q+1)$$

$p > 0$: Man setzt in $(*)$ für p: $p-1$ und $q = A(p,q)$:

$$A(p,q) < A(p-1,A(p,q)) = A(p,q+1) \quad \text{für } p > 1, q > 1$$

$p > 1, q = 0$:

$$A(p,0) = 0 < A(0,1) = 2$$

A 12.4 (P3) Schreiben sie ein rekursives und ein iteratives Programm zur Berechnung der Ackermannfunktion. Vergleichen Sie die Ausführungszeiten.

Lösung

Der Programmcode wird separat bereitgestellt.

A 12.5 (M2) Der übliche Algorithmus zur Multiplikation zweier $n \times n$-Matrizen hat die Komplexität $\mathcal{O}(n^3)$. Nach dem Verfahren von Strassen lassen sich zwei $n \times n$-Matrizen mit der Komplexität $\mathcal{O}(n^{\mathrm{ld}\,7})$ multiplizieren. Für $n = 10$ benötige der übliche Algorithmus auf einem bestimmten Rechner 0,1 Sekunden und der Strassen-Algorithmus 0,12 Sekunden.

a) Um wie viele Sekunden arbeitet der Strassen-Algorithmus schneller als der übliche Algorithmus, wenn $n = 100$ ist?

b) Wie groß muss n sein, damit der Strassen-Algorithmus auf dem gegebenen Rechner doppelt so schnell abläuft wie der konventionelle Algorithmus?

Lösung

a) konventioneller Algorithmus:
$T_M(n) = c_M \cdot n^3$ (dies ist bereits eine Vereinfachung, da alle Terme kleiner dritten Grades vernachlässigt werden).
$T_M(10) = c_M \cdot 10^3 s = 0{,}1s$, also $c_M = 0{,}1/1000 = 10^{-4}$.
$T_M(100) = 10^{-4} \cdot 100^3 s = 100s$.
Strassen-Algorithmus:
$T_S = c_S \cdot n^{\mathrm{ld}\,7} = c_S \cdot n^{2{,}807}$
$T_S(10) = c_S \cdot 10^{2{,}807} s = 0{,}12s$, also $c_S \approx 0{,}12/641{,}21 \approx 1{,}87 \cdot 10^{-4}$.
$T_S(100) = 1{,}87 \cdot 10^{-4} \cdot 100^{2{,}807} s \approx 1{,}87 \cdot 10^{-4} \cdot 4{,}1115 \cdot 10^5 s \approx 76{,}9s$.
Differenz: $D = T_M(100) - T_S(100) = (100 - 76{,}9)s = 23{,}1s$.
Für $n = 100$ arbeitet der Strassen-Algorithmus also um 23,1 sec schneller als der konventionelle Algorithmus.

b) $c_M \cdot n^3 = 2 \cdot c_s n^{2{,}807}$
$n^{3-2{,}807} = n^{0{,}193} = 2 \cdot c_S / c_M = 2 \cdot 1{,}87 \cdot 10^{-4} / 10^{-4} = 3{,}74$
$n = 3{,}74^{1/0{,}193} \approx 3{,}74^{5{,}1813} \approx 929$.
Erst für $n = 929$ arbeitet der Strassen-Algorithmus doppelt so schnell wie der konventionelle Algorithmus.

12 Lösungen zu Kapitel 13 – Suchen und Sortieren

A 13.1 (T1) Beantworten Sie die folgenden Fragen:

a) Ist Suchen in Arrays oder in linearen Listen effizienter durchführbar?

b) Was versteht man unter Radix-Suche?

c) Wodurch unterscheiden sich die sequentielle und die binäre Suche?

Lösung

a) In Arrays ist eine Suche effizienter. Bei einer linearen Liste muss man sich immer vom Start (oder Ende) der Liste von einem Element zum nächsten „hangeln“, während bei Arrays ein direkter Zugriff möglich ist.

b) siehe Buch, Kap. 12.1.4, S. 473

c) Für eine binäre Suche müssen die Daten sortiert sein, für eine sequentielle nicht. Die binäre Suche ist dann erheblich effizienter.

A 13.2 (P3) Vergleich von binärer Suche und Interpolationssuche:

- Erstellen Sie ein Array `a[]` mit 30.000 Integer-Zufallszahlen. Verwenden Sie dabei die Uhrzeit als Startwert für den Zufallszahlengenerator.
- Ordnen Sie das Array in aufsteigender Folge mit einer beliebigen Sortierfunktion.
- Schreiben Sie eine Funktion `search_bin` zum binären Suchen. Suchen Sie damit in einer eine Million mal durchlaufenen Schleife nach zufällig ausgewählten Zahlen und geben Sie die mittlere Anzahl der Intervallteilungen sowie die mittlere Ausführungszeit aus.
- Schreiben Sie eine Funktion `search_int` zur Interpolationssuche. Suchen Sie damit in einer ebenfalls eine Million mal durchlaufenen Schleife nach denselben Zufallszahlen wie mit der binären Suche und geben Sie auch dafür die mittlere Anzahl der Intervallteilungen sowie die benötigte Ausführungszeit aus. Warnung: es ist auf effiziente Implementierung und auf Rundungsfehler bei der Intervallteilung zu achten.
- Vergleichen und interpretieren Sie die Ergebnisse für die binäre Suche und die Interpolationssuche.

Lösung

Der Programmcode wird separat bereitgestellt.

A 13.3 (T1) Beantworten Sie die folgenden Fragen:

a) Was ist der wesentliche Vorteil von Hash-Verfahren?

b) Was versteht man unter Klumpenbildung?

c) Nennen Sie einen Vorteil und einen Nachteil der quadratischen im Vergleich zur linearen Kollisionsauflösung.

d) Welche Forderungen stellt man üblicherweise an Hash-Funktionen?

e) Wie ist der Belegungsfaktor definiert?

f) Kann der Belegungsfaktor größer als eins werden?

Lösung

a) Rascher Zugriff auf die Datensätze mit wenig Schlüsselvergleichen auch bei großen Datenmengen.

b) Wenn durch die Kollisionsauflösung Häufungen von Einträgen im Adressraum entstehen, spricht man von Klumpenbildung. Die lineare Kollisionsauflösung neigt zur Klumpenbildung. Der Nachteil ist, dass in Klumpen eine erhöhte Anzahl von Schlüsselvergleichen erforderlich ist.

c) Die quadratische Kollisionsauflösung verteilt die Einträge gleichmäßiger über den Adressraum, die Klumpenbildung ist also geringer. Die Größe des Adressraums muss eine Primzahl sein, aber auch dann wird von einer bestimmten Primäradresse ausgehend nur die Hälfte des Adressraums erfasst.

d)
- Die Hash-Funktion soll schnell aus dem Primärschlüssel berechnet werden können.
- Ein bestehende Ordnung der Primärschlüssel soll erhalten bleiben.
- Die Hash-Funktion soll die Adressen gleichmäßig über die m möglichen Adressen verteilen.
- Alle Schlüssel sollen mit gleicher Wahrscheinlichkeit auftreten.
- Die durch die Kollisionsbehandlung berechneten Adressen sollen gleichmäßig über den Adressraum verteilt werden.

e) Der Belegungsfaktor lautet $\mu = n/m$, wobei n die Anzahl der gespeicherten Datensätze ist und m der Umfang des Adressraums.

f) Der Belegungsfaktor kann größer als 1 werden, wenn die Anzahl der gespeicherten Datensätze die Größe des Adressraums übersteigt. Dies kann dann der Fall sein, wenn der Adressraum nur für die Primäradressen gilt, der Speicherplatz für die Kollisionsauflösung aber durch dynamische Speicherplatzzuweisung erfolgt, beispielsweise bei Kollisionsauflösung durch lineare Listen.

A 13.4 (M1) Berechnen Sie die mittlere Anzahl S von Vergleichen zum Auffinden eines beliebigen Datensatzes in einer Hash-Tabelle, die maximal 100.000 Datensätze enthalten kann und bereits mit 86.000 Datensätzen gefüllt ist. Dabei kann von idealen Bedingungen ausgegangen werden.

Lösung

Für ideale Verhältnisse gilt (siehe Gleichung (13.31) Buch S. 596):

$$S(\mu) = \frac{1}{\mu} \ln \frac{1}{1-\mu}$$

Mit $\mu = n/m = 86000/100000 = 0{,}86$ folgt: $S = 1{,}16279 \cdot \ln 7{,}14286 = 2{,}28618$. Es sind also im Mittel etwas mehr als zwei Vergleiche erforderlich.

A 13.5 (M3) Es soll eine Hash-Tabelle mit einem Adressraum von $m = 13$ angelegt werden. Die Hash-Funktion sei definiert durch die Vorschrift:
$h(\text{Name}) = (\text{pos}(1.\ \text{Buchstabe}) + \text{pos}(2.\ \text{Buchstabe})) \bmod 13$.
Dabei gibt pos() die Position des betreffenden Buchstaben im Alphabet an, also $\text{pos}(\text{A}) = 1$ etc.

a) Tragen Sie die folgenden Datensätze in der angegebenen Reihenfolge unter Verwendung der linearen und der quadratischen Kollisionsauflösung in die Hash-Tabelle ein: Hammer, Feile, Nagel, Zange, Zwinge, Raspel, Schraube, Niete, Pinsel.

b) Geben Sie den Belegungsfaktor an.

c) Berechnen Sie die mittlere Anzahl der Vergleiche für erfolgreiche und erfolglose Suche.

Lösung

a) Berechnung des Hashwerts:

Name	Hammer	Feile	Nagel	Zange	Zwinge	Raspel	Schraube	Niete	Pinsel
Pos	$8+1$	$6+5$	$14+1$	$26+1$	$26+23$	$18+1$	$19+3$	$14+9$	$16+9$
h(Name)	9	11	2	1	10	6	9	10	12

Hash-Tabelle:

Hash-Adresse	Name (linear)	Name (quadratisch)
0	Niete	Schraube
1	Zange	Zange
2	Nagel	Nagel
3	Pinsel	–
4	–	–
5	–	–
6	Raspel	Raspel
7	–	Niete
8	–	–
9	Hammer	Hammer
10	Zwinge	Zwinge
11	Feile	Feile
12	Schraube	Pinsel

b) Belegungsfaktor: $\mu = n/m$, wobei n die Anzahl der gespeicherten Datensätze ist und m der Umfang des Adressraums. Hier: $\mu = 9/13 \approx 0{,}692$.

Es ist $n = 9$, da die 9 Datensätze aus Aufgabe (b) eingetragen werden sollen.

c) linear

Anzahl Vergleiche bei erfolgreicher Suche:

$S_l = (1+1+1+1+1+1+4+4+5)/9 = 19/9 \approx 2{,}11$

Anzahl Vergleiche bei erfolgloser Suche:

$U_l = (5+4+3+2+1+1+2+1+1+9+8+7+6)/13 = 50/13 \approx 3{,}85$

quadratisch Anzahl Vergleiche bei erfolgreicher Suche:

$S_q = (1+1+1+1+1+1+3+7+1)/9 = 17/9 \approx 1{,}89$

Anzahl Vergleiche bei erfolgloser Suche:

$U_q = (3+3+2+1+1+1+6+2+1+4+13)/13 = 37/13 \approx 2{,}85$

Der letzte Fall (die 13) ist hier interessant: Will man in die Tabelle noch ein weiteres Element mit Index 12 und quadratischer Kollisionsauflösung eintragen, so stellt man fest, dass dies nicht möglich ist, weil man in einen Zyklus läuft. Die Tabelle ist in diesem Fall also praktisch voll, obwohl eigentlich Elemente frei sind.

Es genügt bei der quadratischen Kollisionsauflösung den Zähler i bis maximal $n-1 = 12$ laufen zu lassen. Hat man bis zu diesem Punkt keinen freien Platz gefunden, so ist ein Eintrag in die Tabelle nicht möglich.

A 13.6 (P4) Es soll eine Datenverwaltung unter Verwendung der gestreuten Speicherung aufgebaut werden. Die Primäradresse A soll dabei möglichst einfach unter Einhaltung der lexikografischen Ordnung aus dem

Primärschlüssel der Datensätze berechnet werden. Die Kollisionsbehandlung soll mithilfe einfach verketteter nach dem Primärschlüssel geordneter linearer Listen erfolgen. Es sollen die Operationen Einfügen, Suchen, Löschen und Auflisten von Datensätzen realisiert werden.

Lösung

Der Programmcode wird separat bereitgestellt.

A 13.7 (T1) Beantworten Sie die folgenden Fragen:

a) Welches direkte Sortierverfahren benötigt im Mittel am wenigsten Schlüsselvergleiche?

b) Welches direkte Sortierverfahren benötigt im Mittel am wenigsten Zuweisungen?

c) Welches direkte Sortierverfahren arbeitet für bereits sortierte Daten am schnellsten?

Lösung

a) Insertion Sort mit binärem Einfügen

b) Selection Sort

c) Abgesehen von Selection Sort ist es egal, welches verwendet wird, alle benötigen die kleinstmögliche Anzahl von n Vergleichen bei n Daten. Betrachtet man noch die Anzahl an Zuweisungen, so ist Bubblesort am schnellsten.

A 13.8 (P2) Implementieren Sie die Sortierverfahren direktes Einfügen, direktes binäres Einfügen, direktes Auswählen, Bubblesort und Shakersort. Erzeugen Sie Zufallsfolgen von double-Zahlen, sortieren Sie diese mit allen implementierten Sortierfunktionen und geben Sie die Sie jeweiligen die Laufzeiten aus.

Lösung

Der Programmcode wird separat bereitgestellt.

A 13.9 (T1) Beantworten Sie die folgenden Fragen:

a) Was bedeutet die Aussage, dass der Quicksort entarten kann?

b) Welches Sortierverfahren benötigt die geringste Anzahl von Schlüsselvergleichen?

c) Wodurch ist eine Partition gekennzeichnet?

d) Was ist ein Median?

Lösung

a) Bei Quicksort wird das zu sortierende Array in zwei Teile zerlegt. Optimal ist es, wenn beide Teile gleich groß sind. Im Extremfall kann aber eine Entartung auftreten, nämlich eine Partitionierung in zwei Zerlegungen, von denen die eine nur ein einziges Element enthält und die andere den gesamten Rest des Arrays.

b) Siehe hierzu Buch, Kap. 13.5.3, S. 612.

Grundsätzlich ist hier natürlich die Frage, ob die Anzahl Schlüsselvergleiche für den besten, schlechtesten oder einen mittleren Fall betrachtet werden sollen. Die Antwort wird dann unterschiedlich ausfallen. Ist nichts darüber gesagt, wird man vom mittleren Fall ausgehen.

c) Eine Partition ist eine Zerlegung einer Menge in Teilmengen, die disjunkt und nicht leer sind. Die Vereinigung aller Teilmengen muss wieder die ursprüngliche Menge ergeben (jedes Element ist also in genau einer Teilmenge enthalten).

d) Den Median einer Menge erhält man, indem man zunächst die Elemente der Größe nach sortiert. Der Median ist dann das Element, das in der Mitte steht.
Bsp.: $S = \{45, 12, 8, 22, 30\}$. Sortierung ergibt: $8, 12, 22, 30, 45$. Der Median ist also 22.

A 13.10 (P2) Schreiben Sie unter Verwendung der C-Funktion `qsort()` ein Programm zum Sortieren eines Feldes von Zeigern, die auf Strings deuten. Die Strings sollen dabei in absteigender Reihenfolge lexikografisch sortiert werden, also c vor b vor a etc.

Lösung

Der Programmcode wird separat bereitgestellt.

A 13.11 (M2) Ein Test habe ergeben, dass in Abhängigkeit von der Anzahl n der Daten für das Sortieren von Integer-Arrays mit dem Insertionsort auf einer bestimmten Maschine $t_i = 2{,}8n^2 \cdot 10^{-5}$ Sekunden benötigt werden. Mit dem Quicksort benötigt man $t_q = 1{,}4n \cdot \mathrm{ld}\, n \cdot 10^{-4}$ Sekunden. Von welchem n ab ist der Quicksort schneller als der Insertionsort?

Lösung

$2{,}8n^2 \cdot 10^{-5} = 1{,}4n \cdot \mathrm{ld}\, n \cdot 10^{-4}$ ist nach n aufzulösen. Es folgt:
$0{,}2n = \mathrm{ld}\, n \rightarrow 2^{0{,}2n} = n$.
Wertetabelle:

n	$2^{0{,}2n}$
10	$2^2 = 4$
20	$2^4 = 16$
22	$2^{4{,}4} \approx 21{,}1$
23	$2^{4{,}6} \approx 24{,}25$

ab $n = 23$ ist der Quicksort schneller.

A 13.12 (T1) Beantworten Sie die folgenden Fragen:

a) Was ist ein nicht-flüchtiger Speicher?

b) Was ist sequentieller und halbsequentieller Speicherzugriff?

c) Was versteht man unter der Zykluszeit im Zusammenhang mit Speicherzugriffen?

d) Was bewirkt eine Defragmentierung?

e) Was ist der Interleave-Faktor?

Lösung

a) Man unterscheidet je nachdem, ob der Speicher seine Daten ohne Aufrechterhaltung der Betriebsspannung verliert oder nicht, als flüchtig (volatile) oder nichtflüchtig (non-volatile).

b) siehe Buch, Kap. 13.6.2, S. 510.

c) siehe Buch, Kap. 13.6.2, S. 510.

d) Durch häufiges Löschen und erneutes Beschreiben von Daten auf einer Festplatte tritt eine mit der Zeit immer stärkere Fragmentierung ein. Dies kann schnell einen erheblichen Geschwindigkeitsverlust zur Folge haben. Durch Defragmentier-Programme können die Dateien so umgeordnet werden, dass wieder eine fortlaufende Speicherung entsteht.

e) siehe Buch, Kap. 13.6.2, S. 512.

A 13.13 (T1) Beantworten Sie die folgenden Fragen:

a) Grenzen Sie die Begriffe direktes und natürliches Mischen ab.

b) Um welchen Faktor könnte das Sortieren durch Mischen schneller werden, wenn man an Stelle von zwei Sequenzen jeweils vier Sequenzen mischt?

c) Nennen Sie den wesentlichen Vorteil des Ein-Phasen-Mischens im Vergleich zum Zwei-Phasen-Mischen.

Lösung

a) siehe Buch, Kap. 13.6.4, S. 624.

b) Mit $n =$ Anzahl Sequenzen und $m =$ Anzahl Daten erhält man einen Aufwand von $m \log_n m$. Bei zwei Sequenzen also $m \log_2 m$, bei vier $m \log_4 m$.
Es gilt

$$m \log_2 m = m \frac{\ln m}{\ln 2} \quad \text{und} \quad m \log_4 m = m \frac{\ln m}{\ln 4} \quad .$$

Da

$$m \frac{\ln m}{\ln 4} = m \frac{\ln m}{\ln 2^2} = m \frac{\ln m}{2 \ln 2} = \frac{1}{2} m \log_2 m \quad ,$$

ist das Sortieren mit vier Sequenzen also doppelt so schnell wie das mit zwei Sequenzen.

c) Vergleicht man 2-Phasen-3-Band-Mischen mit dem ausgeglichenen 1-Phasen-4-Band-Mischen, so ist ein wesentlicher Vorteil des letzteren, dass hier auch die Verteilungsphase zum Sortieren beiträgt.

A 13.14 (L2) Sortieren Sie die Daten $a = \{27, 31, 11, 42, 89, 16, 17, 14, 12, 64, 50, 61, 72, 26, 28, 32, 66, 19, 22, 83, 87, 99\}$ nach dem in Abb. 13.17, S. 624 vorgeführten Muster per Hand unter Verwendung des natürlichen Mischens mit zwei Hilfsbändern b und c.

Lösung

a:	27	31	11	42	89	16	17	14	12	64	50	61	72	26	28	32	66	19	22	83	87	99
b:	27	31	16	17	12	64	26	28	32	66												
c:	11	42	89	14	50	61	72	19	22	83	87	99										
a:	11	27	31	42	89	14	16	17	12	50	61	64	72	19	22	26	28	32	66	83	87	99
b:	11	27	31	42	89	12	50	61	64	72												
c:	14	16	17	19	22	26	28	32	66	83	87	99										
a:	11	14	16	17	27	31	42	89	12	19	22	26	28	32	50	61	64	66	72	83	87	99
b:	11	14	16	17	27	31	42	89														
c:	12	19	22	26	28	32	50	61	64	66	72	83	87	99								
a:	11	12	14	16	17	19	22	26	27	28	31	32	42	50	61	64	66	72	83	87	89	99

A 13.15 (P4) Schreiben Sie eine Funktion zum Sortieren durch Mischen eines Arrays am Platz. Es darf also kein wesentlicher zusätzlicher, von der Anzahl n der Daten abhängiger Speicherplatz verwendet werden.

Lösung

Der Mergesort arbeitet üblicherweise nicht in-place, braucht also zusätzlichen Speicher. Eine in-place Implementierung ist nicht ganz einfach; ein Beispiel findet man hier:
`https://github.com/liuxinyu95/AlgoXY/blob/algoxy/sorting/merge-sort/src/mergesort.c`
Eine Beschreibung des Verfahrens hier:
`https://sites.google.com/site/algoxy/home/elementary-algorithms.pdf`

13 Lösungen zu Kapitel 14 – Datenstrukturen, Bäume und Graphen

Übungsaufgaben zu Kapitel 14.2

A 14.6 (T1) Beantworten Sie die folgenden Fragen:

a) Was ist ein Nullbaum?

b) Wodurch ist ein innerer Knoten definiert?

c) Was versteht man unter Preorder, Inorder und Postorder?

d) Beschreiben Sie das Ordnungskriterium von binären Suchbäumen.

Lösung

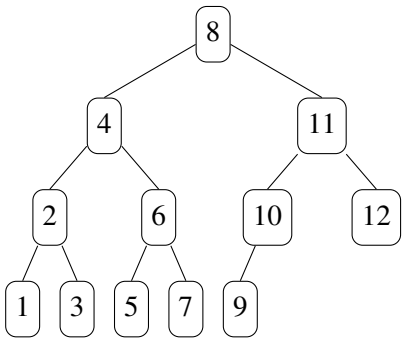
a) Ein Nullbaum ist ein Baum, der keine Knoten hat, also leer ist.

b) In einem Binärbaum ist ein innerer (oder auch interner) Knoten einer, der genau zwei Nachfolger hat.

c) siehe Buch, Kap. 14.2.1, S. 643.

d) Ein Binärbaum heißt *binärer Suchbaum*, wenn für jeden Knoten k gilt, dass alle Schlüssel im linken Teilbaum von k kleiner als der Schlüssel von k sind, und dass alle Schlüssel im rechten Teilbaum von k größer (oder gleich) als der Schlüssel von k sind.

A 14.7 (L2) Gegeben sei der nebenstehende Binärbaum.



a) Geben Sie die Höhe des Baumes an.

b) Handelt es sich um einen erweiterten Binärbaum?

c) Handelt es sich um einen vollständigen Binärbaum?

d) Geben Sie die Durchsuchungslisten an:

- Hauptreihenfolge (Preorder)
- Nebenreihenfolge (Postorder)
- symmetrischer Reihenfolge (Inorder)
- Ebenenreihenfolge (Levelorder)

Lösung

a) Als *Höhe* eines Binärbaums wird die Tiefe des längsten Astes des Baumes bezeichnet. Die *Tiefe* eines Knotens gibt an, wie viele Kanten er von der Wurzel aus entfernt liegt; die Wurzel hat Tiefe 0.
Die Höhe des Baums ist also 3.

b) Für einen erweiterten Binärbaum gilt, dass jeder Knoten entweder keinen Nachfolger hat oder zwei hat. Der abgebildete Teilbaum ist also kein erweiterter Binärbaum, da der Knoten 10 nur einen Nachfolger hat.

c) Bei einem vollständigen Binärbaum sind alle Ebenen (Niveaus) vollständig besetzt. Hier sind die Ebenen 0, 1 und 2 vollständig besetzt, die letzte Ebene 3 ist nicht vollständig besetzt. Es handelt sich also nicht um einen vollständigen Baum.

d) Hauptreihenfolge, Preorder: Wurzel, linker Teilbaum, rechter Teilbaum
8, 4, 2, 1, 3, 6, 5, 7, 11, 10, 9, 12
Nebenreihenfolge, Postorder: linker TB, rechter TB, Wurzel
1, 3, 2, 5, 7, 6, 4, 9, 10, 12, 11, 8
Symmetrische Reihenfolge, Inorder: linker TB, Wurzel, rechter TB
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Ebenenreihenfolge, Levelorder: Ebenenweise von links nach rechts
8, 4, 11, 2, 6, 10, 12, 1, 3, 5, 7, 9

A 14.8 (P2) In einem verkettet gespeicherten Binärbaum seien numerische Werte gespeichert. Schreiben Sie ein Programm zur Durchsuchung dieses Baumes in Hauptreihenfolge, das folgende Informationen ausgibt: Anzahl der besuchten Knoten, kleinster Inhalt, größter Inhalt, Mittelwert aller Inhalte der Knoten. Programmieren Sie eine rekursive und eine iterative Variante mit folgender Knotenstruktur:

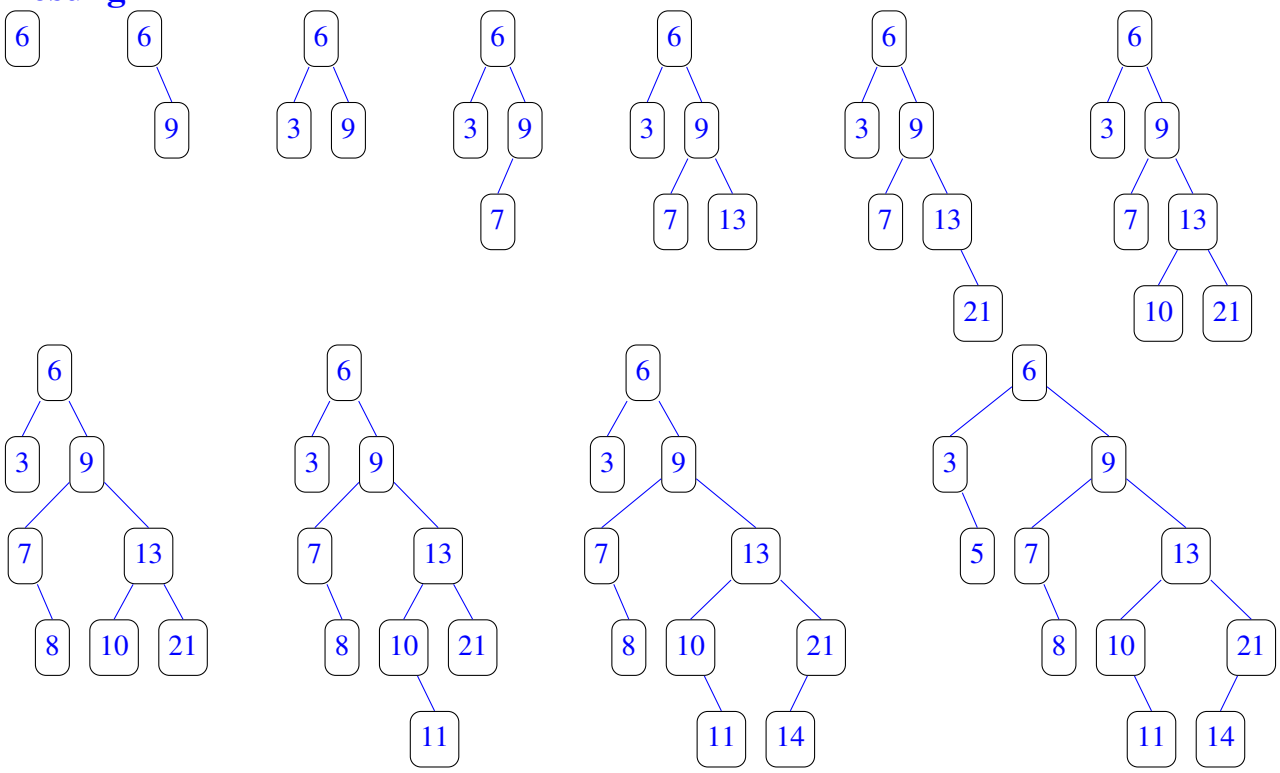
```
struct node { double value; struct node *l; struct node *r; };
```

Lösung

Der Programmcode wird separat bereitgestellt.

A 14.9 (L2) Ordnen Sie die in der Reihenfolge {6, 9, 3, 7, 13, 21, 10, 8, 11, 14, 5} gegebenen Zahlen als binären Suchbaum an.

Lösung



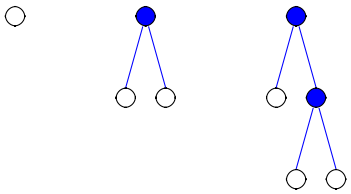
A 14.10 (P4) Schreiben Sie ein Programm zum Verwalten eines binären Suchbaums. Es sollen die Funktionen Initialisieren des Baums, Eingabe, Suchen und Löschen eines Knotens und Ausgabe der Knoteninhalte in Haupt-, Neben- und symmetrischer Reihenfolge. Verwenden Sie dabei die Knotenstruktur `struct node { int info; node *l; struct node *r; };`

Lösung

Der Programmcode wird separat bereitgestellt.

A 14.11 (L2) Ein erweiterter Binärbaum ist dadurch gekennzeichnet, dass jeder Knoten entweder keinen oder zwei Nachfolger hat. Zeigen Sie: In einem erweiterten Binärbaum gilt $n_e = n_i + 1$, wobei n_e die Anzahl der Blätter ist und n_i die Anzahl der der inneren Knoten.

Lösung



Die Skizze zeigt die drei kleinsten erweiterten Binärbäume, wobei die inneren Knoten als schwarze Punkte und die externen Knoten (Blätter) als leere Kreise gezeichnet sind. Für diese gilt offenbar die Behauptung $n_e = n_i + 1$. Ein erweiterter Binärbaum kann nur wachsen, indem ein Blatt durch einen aus Wurzel und zwei Blättern bestehenden Baum ersetzt wird. Es kommen also immer genau ein innerer Knoten und genau ein Blatt hinzu, so dass die Behauptung weiterhin erfüllt bleibt.

A 14.12 (P2) Schreiben Sie eine C-Funktion, die für einen verkettet gespeicherten Binärbaum die Anzahl der Knoten mit zwei Nachfolgern angibt. Die Knotenstruktur sei:
`struct node { int info; node *l; struct node *r; };`

Lösung

```
int nodes(struct node *w)
{
  if(w == NULL) return 0;
  if(w->l != NULL && w->r != NULL) return nodes(w->l) + nodes(w->r) + 1;
  else if(w->l != NULL && w->r == NULL) return nodes(w->l);
  else if(w->l == NULL && w->r != NULL) return nodes(w->r);
}
```

A 14.13 (P3) Schreiben Sie ein möglichst effizientes Programm, mit dem festgestellt werden kann, ob zwei lineare Listen mit jeweils n Elementen vom Typ Integer dieselben Elemente enthalten. Dabei ist nicht vorausgesetzt, dass die Elemente in den beiden Listen in derselben Reihenfolge angeordnet sind. Verwenden Sie dazu einen binären Suchbaum. Bestimmen Sie die im Mittel zu erwartende Komplexität hinsichtlich der Anzahl der Vergleiche.

Lösung

Der Programmcode wird separat bereitgestellt.

A 14.14 (P2) Schreiben Sie unter Verwendung eines binären Suchbaums ein Programm, das alle doppelt vorkommenden Zahlen aus einem Array von Zufallszahlen zählt und löscht. Bestimmen Sie die Komplexität des Algorithmus hinsichtlich der Anzahl der Vergleiche.

Lösung

Der Programmcode wird separat bereitgestellt.

A 14.15 (T2) Beantworten Sie die folgenden Fragen:

a) Was versteht man unter einem Heap?

b) Warum ist für einen Heap die Speicherung in einem Array besonders geeignet?

c) Warum ist ein binärer Suchbaum besser zur Verwaltung von Daten geeignet als ein Heap?

d) Wofür werden Heaps hauptsächlich eingesetzt?

Lösung

a) Man unterscheidet Max-Heaps und Min-Heaps. Unter einem Max-Heap versteht man einen vollständig ausgeglichenen Binärbaum, bei dem für jeden Knoten k gilt, dass die Schlüssel aller seiner Nachfolger kleiner (oder gleich) sind als der Schlüssel von k. Für einen Min-Heap wird die Relation „kleiner“ durch die Relation „größer“ ersetzt. Spricht man nur von einem Heap, so ist ein Max-Heap gemeint.

b) Da ein Heap ein vollständig ausgeglichener Binärbaum ist, entstehen bei der Speicherung in einem Array keine Lücke, so dass die Speicherplatzausnutzung optimal ist. Dadurch ist auch die Adressberechnung zur Ermittlung der Position der Nachfolger und auch des Vorgängers sehr einfach.

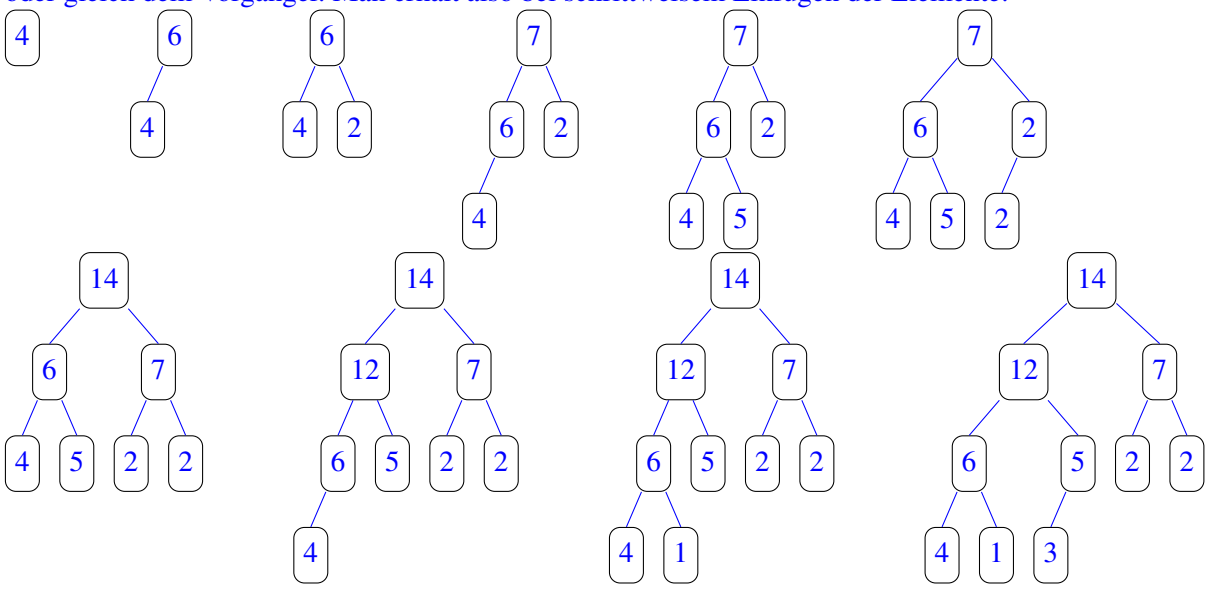
c) In einem binären Suchbaum ist die Komplexität für das Auffinden eines Datensatzes im Mittel von der Ordnung $\mathcal{O}(\mathrm{ld}\, n)$, in einem Heap jedoch nur von der Ordnung $\mathcal{O}(n)$ und damit viel langsamer.

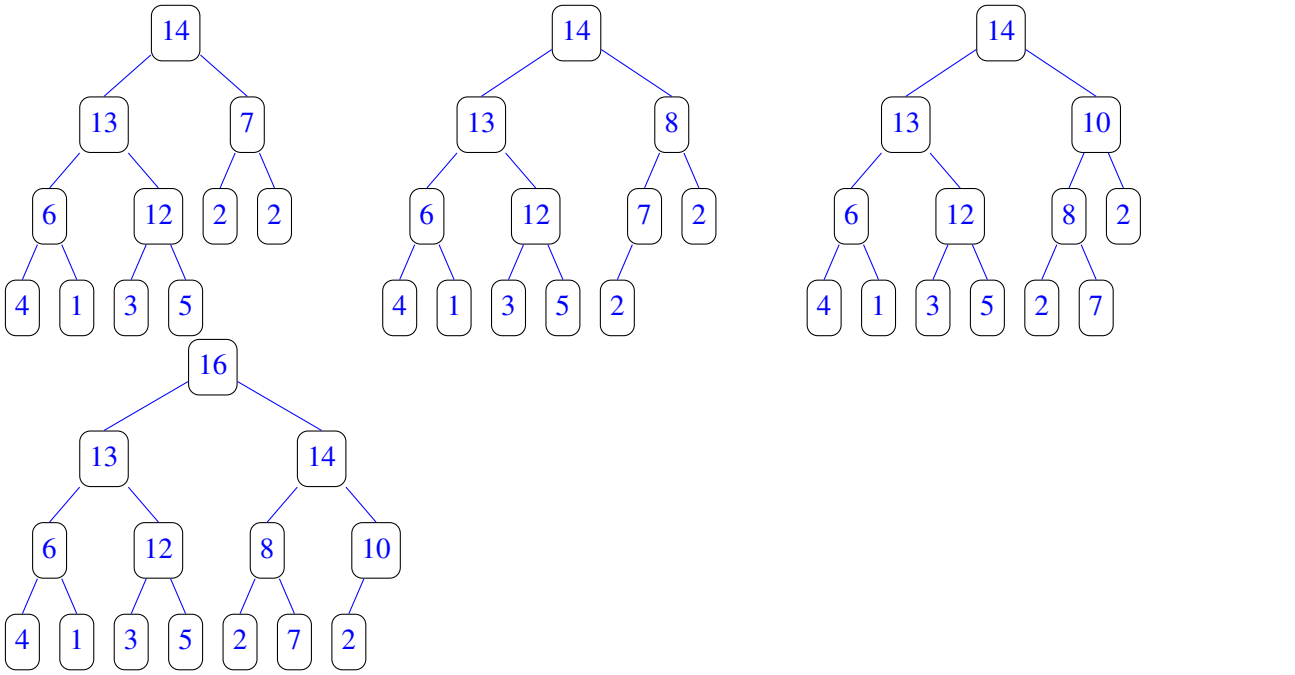
d) Die Hauptanwendungen von Heaps sind der Heap-Sort und Prioritätswarteschlangen.

A 14.16 (L2) Ordnen Sie die Schlüssel {4, 6, 2, 7, 5, 2, 14, 12, 1, 3, 13, 8, 10, 16} unter Beachtung der gegebenen Reihenfolge als Max-Heap und als Min-Heap.

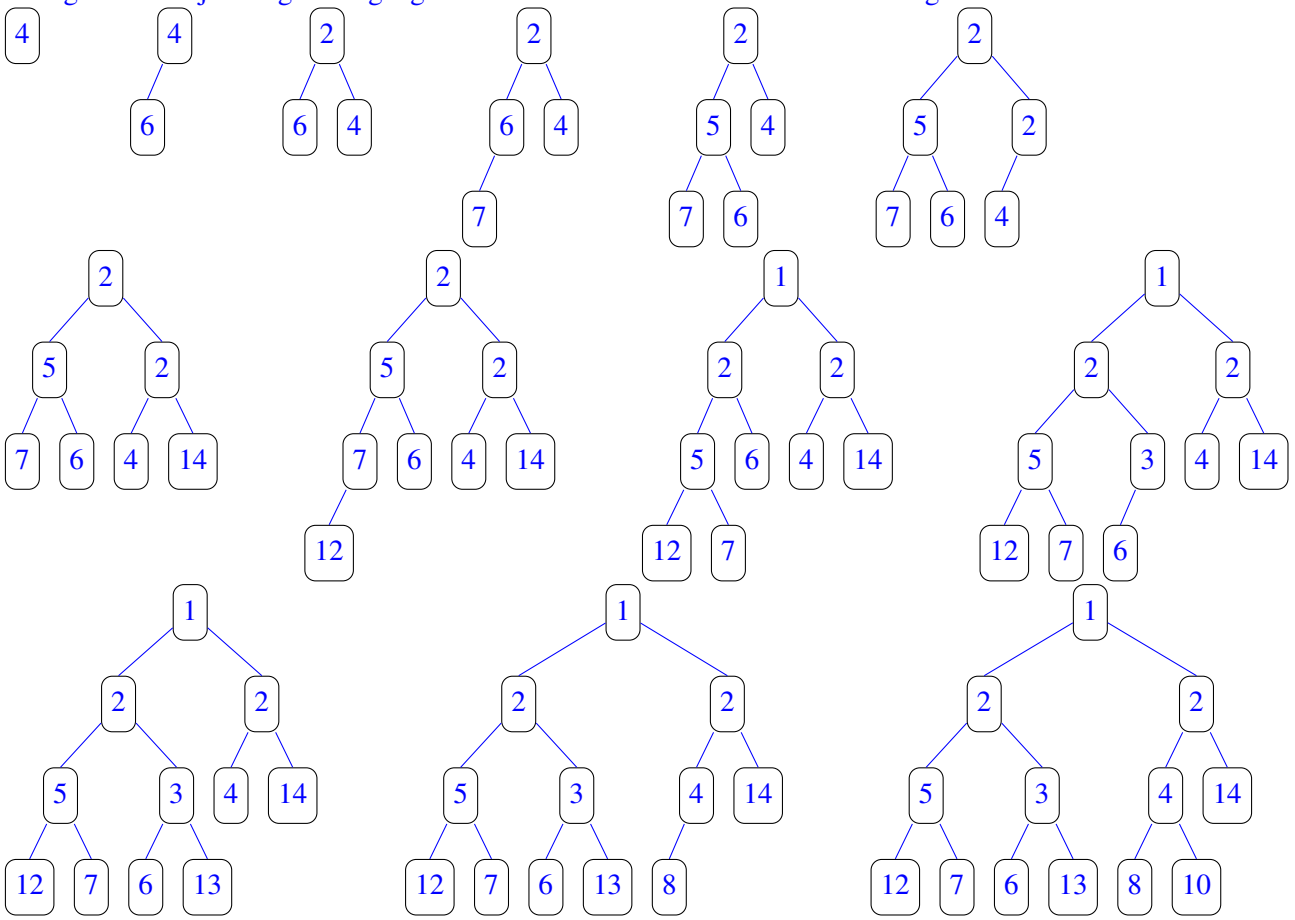
Lösung

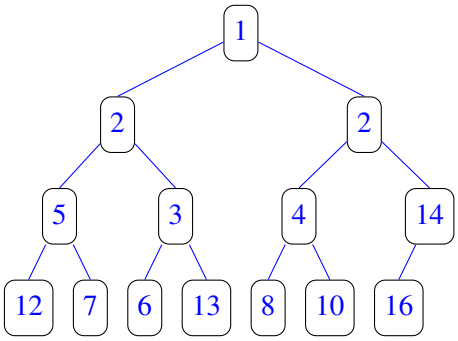
Bei der Ordnung als Max-Heap steht das größte Element in der Wurzel, alle weiteren Elemente sind kleiner oder gleich dem Vorgänger. Man erhält also bei schrittweisem Einfügen der Elemente:





Bei der Ordnung als Min-Heap steht das kleinste Element in der Wurzel, alle weiteren Elemente sind größer oder gleich dem jeweiligen Vorgänger. Man erhält also bei schrittweisem Einfügen der Elemente:

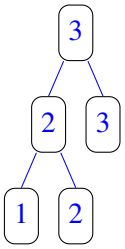




A 14.17 (L2) Geben Sie ein Beispiel für einen Max-Heap mit fünf Knoten an, der gleichzeitig ein binärer Suchbaum ist. Dabei dürfen auch identische Schlüssel zugelassen werden.

Lösung

Bei einem binären Suchbaum gilt für den Schlüssel eines jeden Knotens, dass der Schlüssel des linken Nachfolgers kleiner (oder gleich) und der Schlüssel des rechten Nachfolgers größer (oder gleich) ist. Bei einem Max-Heap gilt eine schwächere Ordnungsbeziehung. Es wird nur gefordert, dass der Schlüssel eines jeden Knotens mit Ausnahme der Wurzel kleiner (oder gleich) dem Schlüssel des Vorgängers ist. Lässt man identische Schlüssel zu, so kann man Heaps konstruieren, die gleichzeitig binäre Suchbäume sind, wie der unten stehende Baum mit fünf Knoten zeigt.



Übungsaufgaben zu Kapitel 14.3

A 14.18 (T2) Beantworten Sie die folgenden Fragen:

a) Was ist bei Vielwegbäumen ein Bruder?

b) Wodurch unterscheidet sich ein BB-Baum von einem $(2,4)$-Baum?

c) Was ist ein (a,b)-Baum?

d) Was ist ein B^+-Baum?

e) Was ist eine Hecke?

Lösung

a) Die auf derselben Ebene befindlichen Nachfolger eines Knotens werden *Brüder* genannt.

b) Sowohl ein BB-Baum als auch ein $(2,4)$-Baum ist ein B-Baum. Die Bezeichnung BB-Baum steht für $(1,2)$-Baum, jede Seite umfasst daher mindestens ein Element und höchstens zwei. Ein BB-Baum mit n Elementen hat ca. die doppelte Tiefe wie ein $(2,4)$-Baum mit derselben Anzahl von Elementen. Ein BB-Baum ist weniger für die Arbeit auf externen Speichern geeignet, sondern eher dann, wenn er komplett im Hauptspeicher gehalten werden kann.

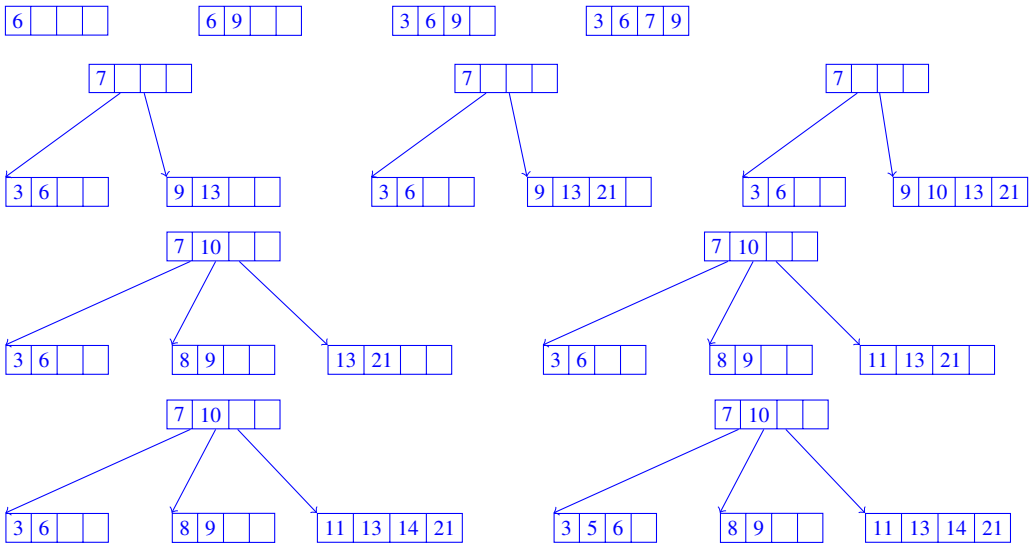
c) siehe Buch, Kap. 13.2.2, S. 551

d) siehe Buch, Kap. 13.2.3, S. 554

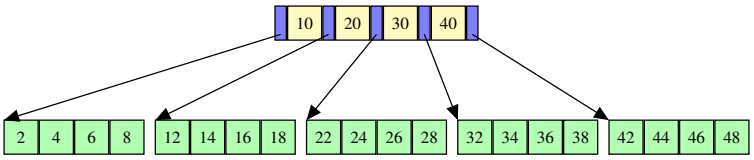
e) siehe Buch, Kap. 13.2.3, S. 554

A 14.19 (L2) Ordnen Sie die in der Reihenfolge {6, 9, 3, 7, 13, 21, 10, 8, 11, 14, 5} gegebenen Zahlen als $(2,4)$-Baum.

Lösung

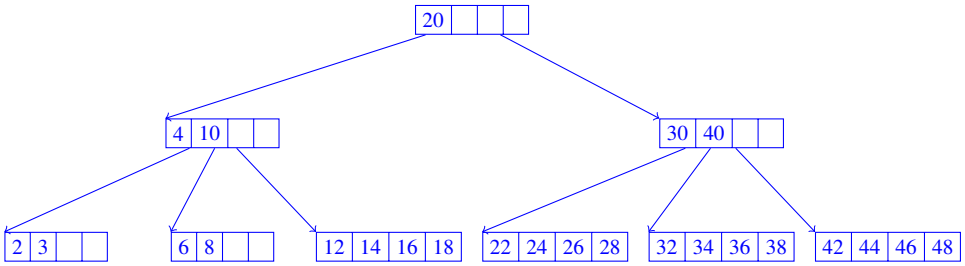


A 14.20 (L3) In den unten abgebildeten $(2,4)$-Baum soll das Element mit Schlüssel 3 eingefügt werden. Zeichnen Sie den resultierenden Baum. Löschen Sie anschließend das Element mit dem Schlüssel 6 und zeichnen Sie den nun resultierenden Baum.

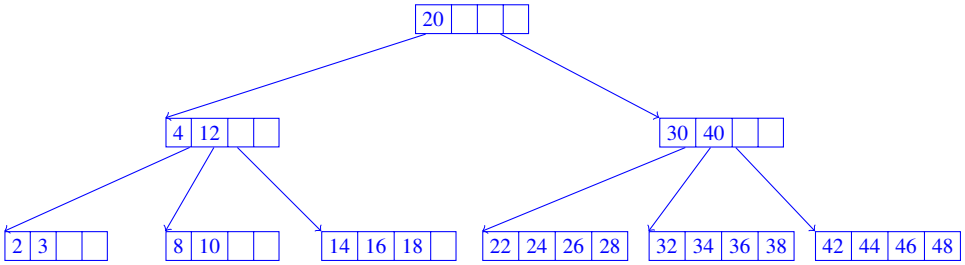


Lösung

Nach dem Einfügen von 3:



Beim Löschen von 6 tritt ein Unterlauf ein, da im Blatt dann nur noch ein einziges Element vorhanden wäre. Der resultierende Baum sieht wie folgt aus:



A 14.21 (L3) Wie viele Elemente können in einem $(2,4)$-Baum der Höhe 2 maximal und minimal enthalten sein? Die Wurzel hat die Tiefe 0.

Lösung

Die Maximale Anzahl ergibt sich, wenn alle Seiten der drei Niveaus mit jeweils 4 Elementen voll besetzt sind. Da dann jede Seite fünf Nachfolgeseiten hat, folgt:
Niveau 0: $n_0 = 4$, Niveau 1: $n_1 = 5 \cdot 4 = 20$, Niveau 2: $n_2 = 5 \cdot 5 \cdot 4 = 100$, Summe: $n = 124$.
Die minimale Anzahl von Elementen in einem $(2,4)$-Baum der Höhe 2 ergibt sich, wenn die Wurzel (Niveau 0) mit nur einem Element besetzt ist. Daraus folgt dann, dass Niveau 1 aus nur zwei Seiten besteht. Diese können dann minimal mit je zwei Elementen besetzt sein, so dass jede Seite aus Niveau 1 auf je drei Seiten in Niveau 2 verweist. Die Seiten in Niveau 2 können wiederum minimal mit zwei Elementen besetzt sein. Insgesamt ergibt sich:
Niveau 0: $n_0 = 1$, Niveau 1: $n_1 = 2 \cdot 2 = 4$, Niveau 2: $n_2 = 2 \cdot 2 \cdot 3 = 12$, Summe: $n = 17$.

A 14.22 (P4) Schreiben Sie ein Programm zum Verwalten eines $(2,4)$-Baums. Es sollen die Funktionen Initialisieren des Baums, Eingabe, Suchen und Löschen von Einträgen und sortierte Ausgabe der Inhalte realisiert werden.

Lösung

Der Programmcode wird separat bereitgestellt.

Übungsaufgaben zu Kapitel 14.4

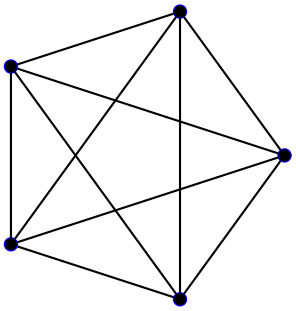
A 14.23 (T1) Beantworten Sie die folgenden Fragen:

a) Wann ist ein Graph schlicht?
b) Was ist ein Eulerscher Kreis?
c) Was ist ein Hamiltonscher Kreis?
d) Was ist der Grad eines Knotens?
e) Was ist ein Wald?
f) Was ist ein Kuratowski-Graph?

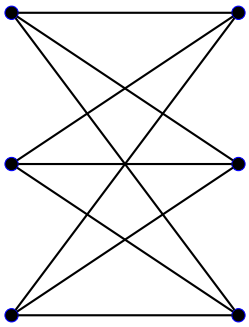
Lösung

a) Ein Graph heißt *schlicht*, wenn er keine Schlingen und Mehrfachkanten enthält.
b) Ein Kreis, auf dem alle Kanten des Graphen genau einmal durchlaufen werden.
c) Ein Weg, auf dem alle Knoten des Graphen genau einmal besucht werden.
d) Der *Grad* $\text{grad}(u)$ eines Knotens u ist die Anzahl der Kanten, bei denen u als einer der Endknoten auftritt.
e) Ein schlichter Graph heißt *Wald* (forest) genau dann wenn er keinen Kreis enthält. Im Grunde ist dies also ein Graph, dessen einzelne Zusammenhangskomponenten Bäume sind.
f) Als Kuratowski-Graph bezeichnet man typischerweise die Graphen K_5 und $K_{3,3}$. K_5 (mit fünf Knoten) ist der kleinste, vollständige, nicht-ebene Graph. $K_{3,3}$ ist ein vollständiger bipartiter Graph mit 6 Knoten, von denen 3 Stück mit den anderen 3 vollständig verbunden sind, untereinander aber gar nicht (deswegen bipartit).

K_5:



$K_{3,3}$:



A 14.24 (L3) Beantworten Sie die folgenden Fragen:

a) Welche Bedingung muss erfüllt sein, damit ein vollständiger Graph genau $n(n-1)/2$ Kanten hat?
b) Beweisen Sie durch vollständige Induktion, dass ein vollständiger Graph mindestens $n(n-1)/2$ Kanten hat.
c) Zeigen Sie: Jeder kreisfreie Graph mit n Knoten enthält höchstens $n-1$ Kanten.
d) Auf welche Eigenschaft muss man die Adjazenzmatrix eines Graphen testen, um zu entscheiden, ob er schlingenfrei ist?
e) Wie groß ist die minimale und wie groß ist die maximale Anzahl von Kanten eines schlichten, zusammenhängenden Graphen mit n Knoten?
f) Wie viele Knoten hat ein ungerichteter, schlichter, vollständiger Graph mit 465 Kanten?
g) Wie viele Kanten hat ein nicht-ebener, zusammenhängender Graph mit n Knoten mindestens?

Lösung

a) $n(n-1)/2$ ist die Mindestanzahl Kanten, die ein Graph mit n Knoten haben muss, damit er vollständig ist. Damit er exakt diese Anzahl hat, darf er keine parallelen Kanten und keine Schlingen enthalten, er muss also schlicht sein.

b) Induktionsbeginn $n = 1$: Der einfachste vollständige Graph mit einem Knoten besteht nur aus diesem Knoten selbst, hat also mindestens $n(n-1)/2 = 0$ Kanten $\rightarrow$ korrekt.
Induktionsschluss: Wenn ein vollständiger Graph mit n Knoten mindestens $n(n-1)/2$ Kanten hat, muss ein Graph mit $n+1$ Knoten nach Annahme mindestens $(n+1)n/2$ Kanten haben.
Nimmt man zu einem Graphen mit n Knoten noch einen weiteren hinzu, so muss dieser mindestens durch eine Kante mit jedem bereits vorhandenen Knoten verbunden werden, es kommen also mindestens n Kanten hinzu:

$$n + \frac{n(n-1)}{2} = \frac{2n + n(n-1)}{2} = \frac{n^2 + n}{2} = \frac{n(n+1)}{2}.$$

Die Annahme ist folglich richtig.

c) Der Beweis erfolgt wieder durch vollständige Induktion.
Induktionsbeginn $n = 1$: Der kreisfreie Graph mit einem Knoten hat keine Kanten, die Annahme ist also richtig.
Induktionsschluss: Nimmt man in einem zusammenhängenden kreisfreien Graphen mit $n+1$ Knoten eine Kante heraus, so zerfällt dieser in zwei Zusammenhangskomponenten mit n_1 bzw. n_2 Knoten. Es gilt $n_1 + n_2 = n + 1$. Jede der Zusammenhangskomponenten enthält höchstens n Knoten, d. h. nach Induktionsvoraussetzung $n_1 - 1$ bzw. $n_2 - 1$ Kanten. Der Originalgraph mit n Knoten hatte eine Kante mehr, also insgesamt $(n_1 - 1) + (n_2 - 1) + 1 = n_1 + n_2 - 1 = n$ Kanten.
Die Annahme ist also richtig.
War der Graph nicht zusammenhängend (es gab also mindestens einen Knoten vom Grad 0), so ist die Argumentation wie oben, nur dass eine Zusammenhangskomponente in zwei Teile zerfällt.

d) In der Hauptdiagonalen müssen überall Nullen stehen.

e) Minimal: $n - 1$ Kanten (sonst ist er nicht zusammenhängend).
Maximal: Man erhält einen vollständigen (schlichten) Graphen, dieser hat $n(n-1)/2$ Kanten.

f) Auflösen von

$$\frac{n(n-1)}{2} = 465$$

nach n ergibt $n = 31$. Die zweite Lösung der quadratischen Gleichung ist negativ und damit nicht sinnvoll.

g) Aussagen über die Planarität von Graphen macht der Satz von Kuratowski. Die beiden in Aufgabe A 1.18 gezeigten Graphen K_5 und $K_{3,3}$ sind die kleinsten nicht-planaren Graphen, die es gibt. Jeder nicht-planare Graph enthält nach Kuratowski eine Unterteilung bestehend aus dem K_5 (10 Kanten) oder dem $K_{3,3}$ (9 Kanten). Beim Unterteilen eines Graphen nach Kuratowski kommen nur noch Kanten dazu, daher muss ein nicht-planarer Graph mindestens 9 Kanten besitzen.

A 14.25 (L3) Gegeben sei der unten gezeigte Graph.

a) Geben Sie die Knotenliste und die Kantenliste an.

b) Geben Sie den Forward Star an.

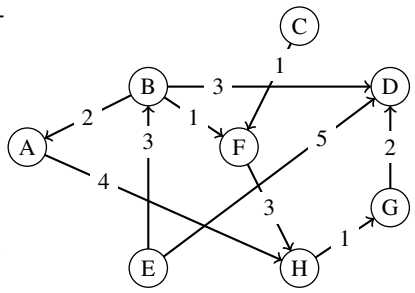
c) Geben Sie die Adjazenz- und Erreichbarkeitsmatrix an.

d) Listen Sie, beginnend mit Knoten E, die Knoten in der Reihenfolge gemäß Tiefensuche und Breitensuche auf.

e) Zeichnen Sie den Graphen so um, dass die Knoten von links nach rechts in einer topologischen Sortierung angeordnet sind.

f) Fassen Sie den Graphen als ungerichteten Graphen auf und geben Sie den minimalen Spannbaum an.

g) Welche der folgenden Eigenschaften treffen auf den Graphen zu: eben, zusammenhängend, stark zusammenhängend, kreisfrei, schlicht, gerichtet, vollständig, bewertet?



Lösung

a)

Knotenliste	Kantenliste
A	H
B	A, D, F
C	F
D	–
E	B, D
F	H
G	D
H	G

b) Zunächst werden die Knoten Indizes wie folgt zugeordnet:

A	B	C	D	E	F	G	H
1	2	3	4	5	6	7	8

Die Speicherung als Forward Star sieht dann so aus:

Knotenindex i:	1	2	3	4	5	6	7	8		
Knotenliste K:	1	2	5	6	6	8	9	10	11	
Nachfolgerliste N:	8	1	4	6	6	2	4	8	4	9

c) Adjazenzmatrix (Knoten in alphabetischer Reihenfolge):

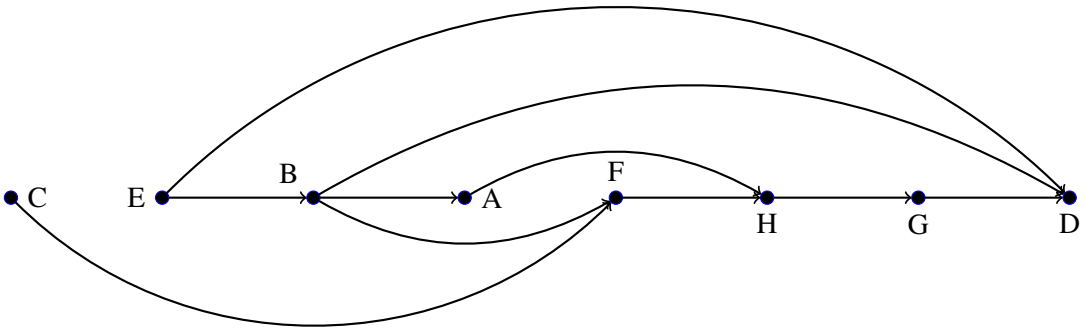
$$\boldsymbol{A} = \begin{pmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \end{pmatrix}$$

Die binäre Erreichbarkeitsmatrix wird über Potenzen von $\boldsymbol{A}$ bestimmt:

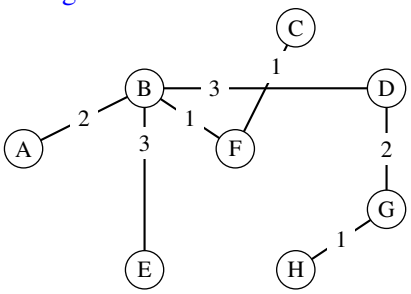
$$\boldsymbol{E}_{\text{bin}} = \begin{pmatrix} 0&0&0&1&0&0&1&1 \\ 1&0&0&1&0&1&1&1 \\ 0&0&0&1&0&1&1&1 \\ 0&0&0&0&0&0&0&0 \\ 1&1&0&1&0&1&1&1 \\ 0&0&0&1&0&0&1&1 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&1&0 \end{pmatrix}$$

d) Tiefensuche: EBAHGFD
Breitensuche: EBDAFHG

e) Topologische Sortierung:



f) Es ergibt sich:



g)

Eigenschaften:	Ja	Nein
eben	X	
zusammenhängend	X	
stark zusammenhängend		X
kreisfrei	X	
schlicht	X	
gerichtet	X	
vollständig		X
bewertet	X	

A 14.26 (P2) Schreiben Sie eine C-Funktion zur topologischen Sortierung eines Digraphen mit n Knoten. Gegeben seien die Kantenliste als Array klist und die global deklarierte Adjazenzmatrix $a[n][n]$. Die Funktion soll als Ergebnis die Indizes der Knoten bezüglich klist in der sortierten Reihenfolge in ein Integer-Array tlist eintragen.

Lösung

```
#define DIM 20
int a[DIM][DIM];          // Global deklarierte Adjazenzmatrix

int tsort(int tlist[], int klist[], int n)
{
  int i, j, k = 0, grad[DIM];
  if(n < 1 || n >= DIM) return(-1);
  for(i = 0; i < n; i++) grad[i] = egrad(i);    // Alle Eingangsgrade
  for(i = 0; i < n; i++)
    if(grad[i] == 0) tlist[k++] = i;            // Vorbesetzen
  for(i = 0; i < n; i++)          // Durchlaufen und Aufbauen von tlist
  {
    for(j = 0; j < n; j++)        // Durchlaufe Zeile von Adjazenzm.
    {
      if(a[tlist[i]][j] > 0)      // Knoten tlist[i] hat Nachfolger
      {
        grad[j]--;                // Dekrementiere Grad des Nachfolgers
        if(grad[j] == 0) tlist[k++] = j; // neuer Eintrag in tlist
      }
    }
  }
  return(k);
}
```

14 Lösungen zu Kapitel 15 – Software-Engineering

A 14.1 (T1) Recherchieren Sie im Internet den Begriff „Softwarekrise“. Was bedeutet er?

Lösung

Der Begriff „Softwarekrise“ bezieht sich auf die wachsenden Probleme, die in den 1960er und frühen 1970er Jahren in der Softwareentwicklung auftraten, als die Komplexität der Computerhardware und die damit verbundenen Softwareanforderungen rasant zunahmen. Es gab eine Reihe von Herausforderungen, die die Softwareindustrie damals beschäftigten:

Unvorhersehbare Entwicklungszeiten und -kosten Softwareprojekte überzogen häufig den geplanten Zeitrahmen und das Budget.

Mangelnde Qualität Die entwickelte Software enthielt oft viele Fehler und war nicht zuverlässig.

Unzureichende Leistung Trotz steigender Hardwarekapazitäten konnte die Software oft nicht mit der gestiegenen Leistung mithalten.

Wartungsschwierigkeiten Es war schwierig, Software zu aktualisieren oder zu erweitern, ohne neue Fehler zu einzubauen oder vorhandene Funktionen zu beeinträchtigen.

Kommunikation zwischen Entwicklern und Benutzern Es gab eine wachsende Kluft zwischen dem, was Benutzer von Software erwarten, und dem, was Entwickler lieferten.

Mangel an standardisierten Methoden Es gab keine allgemein akzeptierten besten Praktiken oder Standards für die Softwareentwicklung.

Diese Probleme führten zu erheblichen Anstrengungen, die Qualität und Vorhersagbarkeit der Softwareentwicklung zu verbessern. Hierzu fand in den Jahren 1968 und 1969 eine berühmt gewordene Konferenz in Garmisch statt. Auf dieser wurde der Begriff des Software-Engineerings von F. L. Bauer geprägt und ein Vorgehen wie in den Ingenieur-Wissenschaften eingefordert. Ein Vorschlag dazu wurde beispielsweise von Winston Royce 1970 unterbreitet. Sein Vorschlag bildete die Grundlage für das Wasserfallmodell.

A 14.2 (T2) Gehen Sie die elektronischen Systeme in Ihrem Auto durch (beispielsweise Motorsteuerung, ABS, ESP, Navi, Radio): Überlegen Sie was passiert, wenn das System ausfällt und was passiert, wenn das System falsch funktioniert. Sortieren Sie diese Systeme nach dem Schaden, den Fehlfunktionen verursachen könnten. Müssen Sie das Navi genauso gut testen wie das ABS?

Lösung

Autos sind mit vielen elektronischen Systemen ausgestattet. Diese Teilsysteme sind unterschiedlich wichtig für die Sicherheit des Fahrers oder der Fahrerin. Wenn ein Teilsystem ausfällt, kann das die Insassen des Fahrzeugs mehr oder weniger gefährden. Sicherheitskritische Teile müssen Sie nachvollziehbarer bauen und besser testen als unkritische. Wir gehen einige dieser Systeme durch und überlegen, was passieren könnte, wenn sie ausfallen:

Motorsteuerung Bei einem Ausfall startet das Auto nicht, kann während der Fahrt plötzlich stehen bleiben oder den Motor sogar beschädigen. Ebenso bei einer Fehlfunktion: Der Motor kann unregelmäßig laufen,

zu viel Kraftstoff verbrauchen, weniger Leistung haben oder im schlimmsten Fall einen Motorschaden erleiden.

ABS (Antiblockiersystem) Fällt das ABS aus, können bei einer starken Bremsung die Räder blockieren, wodurch das Fahrzeug ins Rutschen gerät. Eine Fehlfunktion führt beispielsweise zu einer Verlängerung des Bremswegs oder zum Ausbrechen des Fahrzeugs während der Bremsung.

ESP (Elektronisches Stabilitätsprogramm) Das Auto bietet weniger Unterstützung beim Korrigieren von Über- oder Untersteuern in Kurven oder bei rutschigen Bedingungen, wenn das ESP ausfällt. Die älteren Leser werden sich noch an den Elchtest erinnern. Das System könnte das Fahrzeug bei einer Fehlfunktion unnötig korrigieren.

Navigationssystem Bei einem Ausfall würde der Fahrer keine Routenführung oder Verkehrsinformationen erhalten. Damit wäre er oder sie aber nicht direkt gefährdet. Ebensowenig wenn das Radio und das Infotainment ausfällt.

Airbagsystem Dagegen würde er Ausfall oder eine falsche Auslösung des Airbags das Leben der Fahrzeuginsassen direkt bedrohen. Offenbar ist dieses also deutlich kritischer einzuschätzen als das Radio.

Eine mögliche Reihung von sicherheitskritisch nach unkritisch wäre etwa: Airbagsystem, ABS, Motorsteuerung, ESP, Navigationssystem. Für eine genauere Bewertung müssten wir Ausfälle im Detail durchspielen und deren Schaden bewerten.

A 14.3 (T2) Wenn nur besonders gut getestet wird, ist die Software in jedem Fall fehlerfrei? Ist eine fehlerfreie Software möglich? Unter welchen Randbedingungen sollten Sie Ihre Software besonders intensiv testen und alle Tests auch dokumentieren?

Lösung

Edsger W. Dijkstra schrieb zu dieser Frage: *Program testing can be used to show the presence of bugs, but never to show their absence!*.
Selbst wenn Software intensiv und umfassend getestet wird, kann sie nie mit absoluter Sicherheit als völlig fehlerfrei betrachtet werden. Es gibt mehrere Gründe dafür:

Selten vollständige Verhaltensbeschreibungen Häufig wird Software durch natürliche Sprache oder einfache und unvollständige Modelle beschrieben, beispielsweise sagt ein Klassendiagramm noch nichts über das Verhalten des Systems aus. Ein oder mehrere Sequenzdiagramme beschreiben selten das vollständige Verhalten, sondern geben in der Regel nur Beispiele für das Verhalten des Systems. Wenn wir gar keine vollständige Beschreibung des Verhaltens der Software haben, kann es Fälle geben im Test, wo wir nicht zwischen Fehler und gewolltem Verhalten unterscheiden können.

Kombinatorische Explosion In der Theorie wäre es notwendig, jeden möglichen Pfad durch die Software und jede mögliche Kombination von Eingabewerten zu testen, um die Software als völlig fehlerfrei zu betrachten. Bei komplexer Software kann dies zu einer nahezu unendlichen Anzahl von Testfällen führen.

Externe Faktoren Software kann von externen Faktoren beeinflusst werden, die während des Tests nicht berücksichtigt wurden, z. B. Hardwarefehler, neue Version von Basissoftware, anderes Verhalten des Netzwerks, ...

Veränderung der Umgebung Die Umgebung, in der Software betrieben wird (z. B. Betriebssysteme, Hardware, Netzwerke), ändert sich ständig. Ein Update oder eine Änderung in einem dieser externen Systeme kann zu Problemen in der zuvor fehlerfreien Software führen. Speziell das Verhalten im Scheduling vom Betriebssystem und darunter liegender Hardware mit mehreren Prozessen bzw. Threads kann durch Tests kaum nachgestellt werden.

A 14.4 (T1) Beschaffen Sie sich das V-Modell XT als PDF-Dokument. Betrachten Sie den Vorgehensbaustein „SW-Entwicklung“. Welche Produkte werden gefordert? Welche Rollen sind an der Erstellung und Qualitätssicherung dieser Produkte beteiligt? Was genau ist ein Prüfkonzept?

Lösung

Vorgehensbaustein SW-Entwicklung

Gefordert werden:

- Systemspezifikation: Software-Spezifikation, Externes SW-Modul Spezifikation.
- Systementwurf: Datenbankentwurf, Software-Architektur sowie das Implementierungs-, Integrations- und Prüfkonzept Software
- Systemelemente (Programmierung): Externes SW-Module, SW-Einheiten, SW-Komponenten, SW-Modul

Dabei ist ein SW-Modul eine kleine Einheit, die programmiert wird. Eine oder mehrere SW-Einheiten werden dann zu einer SW-Komponente zusammengesetzt, diese wiederum werden zu einer SW-Einheit integriert.
Zwei Rollen werden in dem Vorgehensbaustein SW-Entwicklung genannt: SW-Architekt und SW-Entwickler. Der Architekt verantwortet die Produkte aus der Systemspezifikation und dem Systementwurf. Der SW-Entwickler ist verantwortlich für die zu programmierenden bzw. integrierenden Systemelemente.

Prüfkonzept

Ein Prüfkonzept beschreibt, wie ein Produkt z. B. eine Software-Modul von der Qualitätssicherung zu prüfen ist. Es gibt dem SW-Entwickler und dem Prüfer Richtlinien für ihre Aufgaben. Das Konzept beschreibt Programmierkonventionen und macht Vorgaben bezüglich Dokumentation, Vorgehen, Werkzeugen und Umgebungen für Implementierung, Installation und Integration. Das Prüfkonzept macht Aussagen darüber, wie genau geprüft werden soll, es schreibt beispielsweise Code-Reviews über Merge- bzw. Pull-Requests vor, definiert eine zu erreichende Testabdeckung mit automatisierten Tests oder legt fest, welche Qualitätsprüfungen über die Build-Pipeline automatisiert werden müssen.

A 14.5 (T1) Beschaffen Sie sich den Scrum-Guide. Was genau steht im Product Backlog? Wer ist dafür verantwortlich, dass die Anforderungen gepflegt werden? Wer ist verantwortlich für den Sprint Backlog und was genau steht da drin?

Lösung

Im Scrum-Guide steht: „The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team.“ Backlog-Items können wir vereinfachend als Anforderungen an das Produkt auffassen, Backlog Items können aber auch Fehlerkorrekturen, technische Experimente (Spikes) oder andere Tätigkeiten zur Wissensbeschaffung sein. Es ist ein lebendiges Dokument, das sich ständig weiterentwickelt und anpasst, basierend auf dem Feedback von Stakeholdern und der Leistung des Entwicklungsteams. Während des gesamten Scrum-Prozesses wird das Product Backlog regelmäßig überarbeitet und aktualisiert, um die sich ändernden Anforderungen und Prioritäten widerzuspiegeln
Der Product Backlog wird vom Product Owner gepflegt, er entscheidet darüber, welche Anforderungen (Backlog Items) in welcher Reihenfolge umgesetzt werden. Das Team hilft ihm dabei, z. B. werden in sog. Backlog Refinement Meetings die Backlog Items weiter verfeinert, priorisiert und es wird der Aufwand zur Umsetzung geschätzt, entweder in Story Points (manchmal auch T-Shirt-Größen) oder in idealen Personen Stunden.

A 14.6 (T1) Definieren Sie die Begriffe Konfigurationsmanagement, Versionsmanagement und Qualitätsmanagement in eigenen Worten!

Lösung

Konfigurationsmanagement

Das Konfigurationselemente sind beispielsweise Code, Konfigurationsdaten, Hardwarekomponenten oder Softwarekomponenten. Zu einem Anwendungsserver in Java gehören dann die Konfigurationselemente: Docker-Images (der gesamte Stapel), zugrundeliegende Linux-Version, Docker-Runtime, die Java Laufzeitumgebung (JRE) sowie die Java-Quelltexte und Konfigurationsdaten.
Eine Konfiguration fasst mehrere Konfigurationselemente in bestimmten Versionen zusammen. Nicht alle Linux-Versionen passen zu jeder Docker-Runtime oder zu jeder JRE. Für den sicheren Betrieb unserer Systeme brauchen wir eine Konfiguration von Konfigurationselementen, die zueinander passen, kompatibel sind.
Konfigurationsmanagement (KM) ist der Prozess der systematischen Handhabung von Änderungen in einem System, um die Integrität und Nachvollziehbarkeit des Systems über dessen gesamten Lebenszyklus zu gewährleisten. Es umfasst die Identifizierung, Dokumentation und Steuerung von Änderungen an Konfigurationselementen.

Versionsmanagement

Versionsmanagement ist ein Teil des Konfigurationsmanagements, der sich auf die Verwaltung von Änderungen an Dateien konzentriert, speziell Textdateien mit Quelltexten, Konfigurationen, Dokumentation oder anderen Daten. Es ermöglicht Entwicklern, frühere Versionen von Dateien und Projekten wiederherzustellen, Änderungen nachzuverfolgen und parallele Änderungen zu koordinieren. Versionsmanagement-Systeme wie beispielsweise git, cvs oder subversion erleichtern die Zusammenarbeit in Teams, indem sie Konflikte von Änderungen verschiedener Entwickler erkennen oder vermeiden. Sie bieten Werkzeuge zur Zusammenführung von Änderungen und zum Erkennen bzw. Vermeiden von Konflikten.

Qualitätsmanagement

Qualitätsmanagement bezieht sich auf die Gesamtheit der Aktivitäten und Verfahren, die darauf abzielen, die Qualität eines Produkts oder einer Dienstleistung zu gewährleisten und zu verbessern. Es umfasst vier Hauptkomponenten: Qualitätsplanung, Qualitätslenkung, Qualitätssicherung und Qualitätsverbesserung. Im Kontext der Softwareentwicklung beinhaltet Qualitätsmanagement die Definition von Qualitätsstandards, die Durchführung von Reviews und Tests zur Überprüfung dieser Standards, die Überwachung des Entwicklungsprozesses zur Vorbeugung von Qualitätsproblemen und die kontinuierliche Suche nach Möglichkeiten zur Verbesserung der Prozesse und des Endprodukts.

A 14.7 (T2) Definieren Sie den Begriff Produktivität von Entwicklern in eigenen Worten! Diskutieren Sie in einer kleinen Gruppe das Teufelsquadrat nach Sneed. Hat Sneed mit seinen Grundaussagen recht, insbesondere damit, dass die Produktivität der Entwickler quasi als konstant angenommen werden kann?

Lösung

Produktivität von Entwicklern

Die Produktivität von Entwicklern bezieht auf die Menge an neuen, reparierten oder geänderten Features, die Softwareentwickler in einer Zeiteinheit liefern. Früher wurden Maße wie Programmzeilen pro Stunde verwendet. Diese Maße kann man leicht manipulieren, in dem umfangreicherer oder umständlicher Code geschrieben wird. Die von A. Albrecht eingeführte Function Point Analyse liefert das Maß der sog. Function Points. Function Points werden durch ein normiertes Zählverfahren erhoben, z. B. werden in einem Datenmodell die Attribute von Klassen gezählt oder die Eingabefelder in einer Maske. Diese Zahlen werden später in Function Points umgerechnet. Function Points pro Stunde sind daher ein Maß dafür, wie viel verwendbare Features in einer Stunde erstellt werden.
Auch mit den Story Points aus Scrum wird ein zur Produktivität verwandtes Maß erhoben, in Form der sog. Project Velocity: Gemessen werden Story Points pro Sprint. Wenn die Produktivität im Team steigt, erhöht sich damit auch die Zahl der erarbeiteten Story Points in einem Sprint und die Velocity steigt.

Ist die Produktivität konstant?

Das Teufelsquadrat (auch bekannt als "magisches Quadrat") nach Harry Sneed ist ein Modell, das die Beziehung zwischen vier wesentlichen Aspekten der Softwareentwicklung darstellt: Kosten, Zeit (Zeitrahmen), Qualität und Umfang (Funktionsumfang). Nach diesem Modell beeinflussen Änderungen in einem dieser Bereiche zwangsläufig die anderen drei. Zum Beispiel kann die Beschleunigung eines Projekts (Zeit) die Kosten erhöhen, die Qualität beeinträchtigen oder zu einer Reduzierung des Funktionsumfangs führen.
Sneed argumentiert, dass die Produktivität von Entwicklern innerhalb eines bestimmten Rahmens als konstant angenommen werden kann, was bedeutet, dass signifikante Verbesserungen in einem Bereich (z.B. eine schnellere Fertigstellung von Projekten) nur durch Kompromisse in einem oder mehreren der anderen Bereiche (z.B. höhere Kosten, geringere Qualität, oder reduzierter Umfang) erreicht werden können.
In der Praxis können Faktoren wie Erfahrungszuwachs, bessere Werkzeuge, leistungsfähigere Technologien oder positiveres Arbeitsumfeld die Produktivität von Entwicklern erheblich beeinflussen. Die meisten Vorgehensmodelle und auch agile Methoden versuchen kontinuierliche Verbesserung: bessere Werkzeugen und -methoden, Ausbildung oder die Anpassung von Arbeitsprozessen.

A 14.8 (T2) Modellieren Sie den Inhalt des Entity-Relationship-Diagramms aus Abb. 15.11 mithilfe eines UML Klassendiagramms! Was sind die Unterschiede und Gemeinsamkeiten beider Modellierungssprachen? Betrachten Sie dazu auch den aktuellen UML Standard 2.5.1 (vgl. http://www.omg.org/spec/UML/2.5.1/).

Lösung

A 14.9 (T2) Modellieren Sie den folgenden Text als Klassendiagramm: *Ein Auto hat genau vier Räder. Ein VW-Käfer ist ein Auto. Ein Auto kann beliebig viele Fahrer haben. Ein Fahrer hat genau einen Führerschein. Ohne Führerschein ist es kein Fahrer.*

Lösung

A 14.10 (T3) Modellieren Sie die Entscheidungstabelle aus Beispiel 15.4 als Flussdiagramm!

Lösung

15 Lösungen zu Kapitel 16 – Datenbanken

A 15.1 (T) Welche Arten von Daten lassen sich gut als Tabelle darstellen, für welche Daten eignet sich eine Baumstruktur besser? Vergleichen Sie Kundendaten (Nummer, Vorname, Nachname, Geburtsdatum, Straße, Postleitzahl, Ort) mit Daten über eine Vektorgrafik.

Lösung

Tabellen

Tabellarische Daten sind in Zeilen und Spalten organisiert, wobei jede Zeile einen Datensatz repräsentiert und jede Spalte ein Attribut dieses Datensatzes. Tabellen eignen sich hervorragend für:

- Daten, die ein festes Schema haben, also wo die Struktur der Datensätze von Datensatz zu Datensatz nicht oder nicht allzu stark variiert. So dass diese Datenstrukturen mit einer oder wenigen einfachen Tabellen dargestellt werden können.
- Daten, die eher einfache Beziehungen zwischen verschiedenen Datenstrukturen haben. So dass wenige Fremdschlüsselbeziehungen oder Mapping-Tabellen erforderlich sind. Rekursive Beziehungen sind besonders problematisch, also Datensätze die auf Datensätze desselben Typs verweisen.
- Datenstrukturen, die eher flach sind, also ohne untergeordnete Strukturen auskommen. Wenn beispielsweise ein Kunde mehrere Adressen haben kann, ist eine eigene Tabelle für Adressen erforderlich, um den Kunden aus der Datenbank zu lesen oder ihn zu schreiben muss auf zwei Tabellen zugegriffen werden. Je komplizierter die Struktur, desto mehr Tabellen mit Lese- und Schreibzugriff.
- Daten, in denen Konzepte wie Vererbung selten sind.
- Anfragen, die auch mehrere verknüpfte Datenstrukturen in Beziehung setzen wollen. Hierfür eignen sich Joins mehrerer Tabellen gut.

Beispiel: Kundendaten

Kundendaten, wie Nummer, Vorname, Nachname, Geburtsdatum, Straße, Postleitzahl und Ort, lassen sich gut in einer Tabelle darstellen, da sie eine flache Struktur haben. Jeder Kunde kann als eine Zeile in der Tabelle repräsentiert werden, mit Spalten für jede Eigenschaft. Diese Struktur erleichtert das Auffinden, Aktualisieren oder Analysieren von Kundendaten.

Baumstrukturen

Baumstrukturen organisieren Daten hierarchisch, wobei jeder Knoten des Baumes auf einen oder mehrere untergeordnete Knoten (Kinder) verweisen kann. Diese Struktur eignet sich gut für Daten mit einer natürlichen hierarchischen Beziehung oder wenn Daten in einer verschachtelten oder mehrstufigen Hierarchie organisiert werden müssen. Vorteile von Baumstrukturen sind:

- Auch Datensätze, die sich leicht in ihrer Struktur unterscheiden, können jeweils als Baum dargestellt werden. Ein festes Schema ist nicht zwingend erforderlich. Ein festes Schema erleichtert aber das Durchsuchen

dieser Daten. Eine Vektorgrafik im SVG-Format ist bereits in einer Baumstruktur gespeichert (SVG). Die Strukturen von Grafiken sind offensichtlich sehr unterschiedlich.

- Kompliziertere Datenstrukturen mit Unterstrukturen sind ebenfalls gut darstellbar. Solange jeder Datensatz seine eigenen Unterstrukturen hat und nicht mehrere Datenstrukturen sich eine Unterstruktur teilen. Sonst sind Verweise zwischen den jeweils als Baum dargestellten Datenstrukturen erforderlich.
- Eine Baumstruktur kann leicht als ganzes gelesen und geschrieben werden. Joins oder der Zugriff auf mehrere Tabellen sind hier nicht erforderlich.
- Zugriffe innerhalb desselben Baums sind einfach realisierbar, Abfragen über mehrere Datensätze hinweg sind problematisch. Wenn beispielsweise jeder Kunde als Baum dargestellt wird und eine Abfrage nach allen Kunden aus Düsseldorf sucht. Solche Abfragen sind in Tabellarisch modellierten Strukturen einfacher realisierbar.

Beispiel: Daten einer Vektorgrafik

Daten einer Vektorgrafik im SVG-Format eignen sich besser für eine Baumstruktur. Eine Vektorgrafik besteht aus verschiedenen Elementen wie Pfaden, Formen und Gruppen von Elementen, die wiederum aus weiteren Elementen bestehen können. Diese verschachtelte und hierarchische Natur der Daten passt gut zu einer Baumstruktur, in der zum Beispiel ein „Gruppen“-Knoten Kinder für jede enthaltene Form oder jeden Pfad hat.

A 15.2 (T) Installieren Sie sich bitte ein relationales Datenbankmanagement-System auf Ihrem Laptop oder PC. Es gibt viele frei verfügbare relationale DBMS, beispielsweise MySQL, MariaDB oder PostgreSQL. Zusätzlich zum DBMS benötigen Sie eine Shell, mit der Sie SQL gegen die Datenbank absetzen können, beispielsweise phpMyAdmin.

Lösung

Zur Installation bieten sich Docker-Images an, die sie von Docker-Hub beziehen können. Ein Docker-Image kann leicht und restlos von Ihrem Laptop wieder entfernt werden. Wenn Sie das DBMS direkt auf dem Betriebssystem installieren, wird das schwieriger.

A 15.3 (P1) Erstellen Sie mithilfe von SQL ein Datenbankschema mit den beiden Tabellen `Mitarbeiter` und `Abteilung`. Machen Sie dazu geeignete Annahmen über die Datentypen der Attribute. Legen Sie auch die jeweiligen Primärschlüssel und den Fremdschlüssel über die Abteilungsnummer fest!

Lösung

```
CREATE TABLE Mitarbeiter (
    PersonalNr INTEGER NOT NULL,
    Name VARCHAR(100),
    Gehalt DECIMAL(10, 2),
    AbteilungsNr INTEGER,
    PRIMARY KEY (PersonalNr)
);

CREATE TABLE Abteilung (
    AbteilungsNr INTEGER NOT NULL,
    Name VARCHAR(100),
    Standort VARCHAR(100),
    PRIMARY KEY (AbteilungsNr)
);
```

Hinzufügen des Fremdschlüssel-Constraints.

```
ALTER TABLE Mitarbeiter
    ADD CONSTRAINT fk_abteilung_mitarbeiter FOREIGN KEY (AbteilungsNr) REFERENCES Abteilung (Abteilu
```

A 15.4 (P1) Legen Sie mithilfe von **INSERT**-Anweisungen die Daten aus den Beispielen dieses Kapitels in die oben erzeugten Tabellen `Mitarbeiter` und `Abteilung` ein. Was passiert, wenn Sie versuchen, denselben Datensatz, vielleicht Herrn Kunze, mehrfach anzulegen?

Lösung

Zuerst die Abteilung anlegen, wegen der Foreign-Key Constraints.

```
INSERT INTO Abteilung (AbteilungsNr, Name, Standort) VALUES (1, 'Einkauf', 'Muenchen');
INSERT INTO Abteilung (AbteilungsNr, Name, Standort) VALUES (2, 'DV-Org.', 'Augsburg');
INSERT INTO Abteilung (AbteilungsNr, Name, Standort) VALUES (3, 'Produktion', 'Straubing');
INSERT INTO Abteilung (AbteilungsNr, Name, Standort) VALUES (4, 'Entwicklung', 'Augsburg');
INSERT INTO Abteilung (AbteilungsNr, Name, Standort) VALUES (5, 'Verwaltung', 'Muenchen');
```

Danach können Mitarbeiter angelegt werden:

```
INSERT INTO Mitarbeiter (PersonalNr, Name, Gehalt, AbteilungsNr)
VALUES (101, 'Ingo Meissner', 75000, 1);
INSERT INTO Mitarbeiter (PersonalNr, Name, Gehalt, AbteilungsNr)
VALUES (102, 'Ernst Lehmann', 98000, 1);
INSERT INTO Mitarbeiter (PersonalNr, Name, Gehalt, AbteilungsNr)
VALUES (103, 'Heinz Kunze', 75000, 2);
```

Beim Versuch, einen Datensatz mit einem Bereits existierenden Wert für den Primärschlüssel anzulegen kommt folgender Fehler:

```
ERROR: duplicate key value violates unique constraint "mitarbeiter_pkey"
DETAIL: Key (personalnr)=(101) already exists.
```

A 15.5 (P1) Versuchen Sie nun, die in Abschnitt 16.6 dargestellten Ausdrücke der Relationalen Algebra (Selektion σ, Projektion π, kartesisches Produkt $\times$ und den natürlichen Join $\bowtie$) mit der **SELECT**-Anweisung auszuführen.

Lösung

Selektion σ

$\sigma_{Gehalt=75.000}(Mitarbeiter)$

```
select PersonalNr, Name, Gehalt, AbteilungsNr from Mitarbeiter where Gehalt=75000
```

oder

```
select * from Mitarbeiter where Gehalt=75000
```

Projektion π

$\pi_{Name,Gehalt}(Mitarbeiter)$

```
select Name, Gehalt from Mitarbeiter
```

Vereinigung (Union, ∪)

```
SELECT Name, Standort
FROM AbteilungA
UNION
SELECT Name, Standort
FROM AbteilungB;
```

Mengendurchschnitt ∩

Als Inner-Join:

```
SELECT DISTINCT AbteilungA.Name, AbteilungA.Standort
FROM AbteilungA
INNER JOIN AbteilungB ON AbteilungA.Name = AbteilungB.Name
AND AbteilungA.Standort = AbteilungB.Standort;
```

Alternativ, wenn das RDBMS Intersect unterstützt:

```
SELECT Name, Standort
FROM AbteilungA
INTERSECT
SELECT Name, Standort
FROM AbteilungB;
```

Mengendifferenz −

```
SELECT Name, Standort
FROM Abteilung
EXCEPT
SELECT Name, Standort
FROM Abteilung2;
```

```
SELECT Abteilung.Name, Abteilung.Standort
FROM Abteilung
LEFT JOIN Abteilung2 ON Abteilung.Name = Abteilung2.Name
AND Abteilung.Standort = Abteilung2.Standort
WHERE Abteilung2.Name   IS NULL;
```

A 15.6 (P3) Recherchieren Sie Details zum Thema „Isolation von Transaktionen“ (Transaction Isolation Level: Serializable) und versuchen Sie mit zwei parallel laufenden Transaktionen einen Deadlock zu erzeugen.

Lösung

Isolation von Transaktionen, ist ein fundamentales Konzept in Datenbankmanagementsystemen für den gleichzeitigen Zugriff mit mehreren Clients auf dieselben Daten. Es beschreibt, wie und wann die Änderungen, die durch eine Transaktion gemacht werden, für andere Transaktionen sichtbar werden. Diese Isolationsstufen sind entscheidend, um Probleme wie Lost Updates, Dirty Reads, Non-Repeatable Reads und Phantom Reads zu verhindern oder zu erlauben. Sie beeinflussen direkt die Konsistenz und die Integrität der Datenbankdaten. Viele relationale Datenbanksysteme (RDBMS) implementieren die Isolationsstufen gemäß den Spezifikationen des ANSI/ISO SQL Standards, welche vier Hauptstufen der Isolation definieren:

READ UNCOMMITTED

Die niedrigste Isolationsstufe, bei der eine Transaktion die unbestätigten Änderungen einer anderen Transaktion sehen kann. Dies kann zu Dirty Reads führen, d.h., eine Transaktion kann Daten lesen, die eine andere Transaktion geändert hat, aber noch nicht festgeschrieben (committed) hat.

READ COMMITTED

Eine Transaktion kann nur Daten lesen, die von anderen Transaktionen festgeschrieben wurden. Dies verhindert Dirty Reads, aber Non-Repeatable Reads sind möglich, da die Daten zwischen zwei Leseoperationen innerhalb derselben Transaktion von anderen parallel laufenden Transaktionen (Clients) geändert und festgeschrieben werden können.

REPEATABLE READ

Gewährleistet, dass wenn eine Transaktion eine Reihe von Daten liest, jede nachfolgende Leseoperation in derselben Transaktion dieselben Daten zurückgibt, selbst wenn andere Transaktionen Änderungen an diesen Daten vornehmen. Dies verhindert Non-Repeatable Reads, aber Phantom Reads können auftreten.

SERIALIZABLE

Die höchste Isolationsstufe, die vollständige Isolation von Transaktionen gewährleistet. Dies verhindert Dirty Reads, Non-Repeatable Reads und Phantom Reads, indem sichergestellt wird, dass Transaktionen so ausgeführt werden, als würden sie nacheinander ablaufen.

Dead-Lock erzeugen

Starten Sie zwei Datenbank-Shells. Achten Sie darauf, dass diese Transaktionen nicht nach jedem Befehl bestätigen (committen) sondern, dass Sie dies explizit über den Befehl `commit` erledigen können. Häufig wird für jeden Befehl eine eigene Transaktion durchgeführt. Stellen Sie die Transaktionsisolation in beiden auf SERIALIZABLE.
BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
In der einen Shell bearbeiten Sie `Mitarbeiter`, z. B. fügen Sie in die Tabelle einen ein, in der anderen bearbeiten Sie `Abteilung`en, z. B. löschen Sie eine Abteilung. Jetzt versuchen Sie in der ersten Shell zusätzlich Abteilungen und in der zweiten Shell zusätzlich Mitarbeiter zu bearbeiten. Das DBMS müsste jetzt einen Deadlock sehen.

A 15.7 (P2) Recherchieren Sie Details zu JDBC in der Sprache Java und erstellen Sie ein Java-Programm, das über JDBC auf die Datenbank aus den vorangegangenen Übungen zugreift. Lesen Sie alle Mitarbeiter und Abteilungen aus und geben Sie diese auf der Konsole aus!
A 15.8 (T) Recherchieren Sie Details zu den NoSQL DBMS! Erarbeiten Sie für jede der hier genannten Technologien ein Beispiel DBMS und listen Sie zu dem DBMS Einsatzbeispiele auf. Wozu werden NoSQL-DBMS in der Praxis verwendet?

Lösung

Dokumentorientierte Datenbanken Speichern Daten als Dokumente, meist im JSON- oder XML-Format. Diese eignen sich besonders gut für Anwendungen, die komplexe Datenstrukturen haben.

Ein bekanntes Beispiel für eine dokumentorientierte Datenbank ist MongoDB. MongoDB ist eine Open-Source-Datenbank, die Datenstrukturen im BSON-Format (einer binären Erweiterung von JSON)

speichert. Es unterstützt dynamische Schemata, was bedeutet, dass Sie Dokumente speichern können, die unterschiedliche Felder und Strukturen haben, ohne vorher ein Datenbankschema definieren zu müssen. MongoDB wird häufig für Webanwendungen, Content-Management-Systeme und als Backend für mobile Apps eingesetzt, wegen seiner Flexibilität bei den Datenstrukturen.

Schlüssel-Wert-Datenbanken Speichern Daten als Schlüssel-Wert-Paare. Sie sind sehr schnell bei Lese- und Schreiboperationen und eignen sich für Szenarien mit hoher Lese- und Schreiblast.

Ein gängiges Beispiel für eine Schlüssel-Wert-Datenbank ist Redis. Redis ist eine Open-Source-In-Memory-Datenbank, sie kann auch Daten auf einen Sekundärspeicher ablegen. Redis speichert Daten als Schlüssel-Wert-Paare und unterstützt verschiedene Datentypen wie Strings, Listen, Sets, Hashes und mehr. Redis wird häufig für Caching, Session Management, Pub/Sub-Systeme und als schnelle Datenspeicherlösung in (verteilten) Webanwendungen eingesetzt. Beispielsweise als gemeinsamer Cache für mehrere Server.

Spaltenorientierte Datenbanken Organisieren Daten in Spalten statt in Zeilen. Diese Art von Datenbank ist optimiert für das Lesen und Schreiben großer Datensätze und wird oft in der Datenanalyse und für Anwendungen mit extrem hohem Datenaufkommen verwendet.

Ein Beispiel für eine spaltenorientierte Datenbank ist Apache Cassandra. Cassandra ist für ihre Skalierbarkeit und hohe Verfügbarkeit bekannt, ohne dabei einen Single Point of Failure zu haben. Sie eignet sich besonders gut für Anwendungen, die mit großen Mengen von verteilten Daten arbeiten.

Graphdatenbanken Speichern Daten in Form von Graphen. Sie sind ideal für Anwendungen, bei denen Beziehungen zwischen den Daten zentral sind, wie beispielsweise Freundschafts-Graphen in sozialen Netzwerken.

Ein bekanntes Beispiel für eine Graphdatenbank ist Neo4j. Neo4j ist spezialisiert auf die Speicherung und Abfrage von Daten, die in Form von Graphen strukturiert sind, und ermöglicht es, komplexe Beziehungen zwischen Datenpunkten effizient zu modellieren und abzufragen. Es wird häufig in Anwendungen verwendet, die komplexe Beziehungsnetzwerke analysieren, wie soziale Netzwerke, Empfehlungssysteme und Netzwerkanalyse.

Es gibt weitere NoSQL Datenbanken, die derzeit an Bedeutung gewinnen. Das sind Suchmaschinen wie Elasticsearch oder Datenbanken, die speziell für Zeitreihendaten optimiert sind, wie InfluxDB. Sowie Datenbanken welche die Daten über Vektoren ablegen wie Chroma oder Pinecone.

A 15.9 (T) Vergleichen Sie XML mit dem in Kap. 17.4.5 dargestellten JSON-Format! Was sind die Gemeinsamkeiten, was sind die Unterschiede?

Lösung

XML (Extensible Markup Language) und JSON (JavaScript Object Notation) sind zwei weit verbreitete Formate für den Datenaustausch im Web und für die Speicherung von Daten. Beide Formate dienen dem Zweck, Daten in einer strukturierten Form zu übertragen, unterscheiden sich jedoch in Syntax, Verwendung und Eigenschaften. Hier sind die Gemeinsamkeiten und Unterschiede zwischen XML und JSON:

Gemeinsamkeiten

Datenübertragung und -speicherung Sowohl XML als auch JSON werden für die Speicherung und Übertragung von Daten verwendet, insbesondere in Webanwendungen.

Textbasiert Beide Formate sind textbasiert und können mit Standard-Texteditoren gelesen und bearbeitet werden.

Hierarchische Datenstruktur Sie können hierarchische Datenstrukturen repräsentieren, wobei Daten in verschachtelter Form organisiert werden können.

Sprachunabhängig XML und JSON sind sprachunabhängig und können mit den meisten modernen Programmiersprachen verarbeitet werden.

Menschlich lesbar Beide Formate sind für Menschen lesbar, obwohl die Lesbarkeit je nach Komplexität der Daten variieren kann.

Unterschiede

Syntax und Verwendung **XML:** Ist umfangreicher und flexibler in der Struktur. XML unterstützt Attribute, Kommentare, Namensräume und ist erweiterbar, was bedeutet, dass Benutzer ihre eigenen Tags definieren können. Es wird oft in komplexen Anwendungen oder dort eingesetzt, wo Dokumentstrukturen häufig geändert werden. **JSON:** Hat eine einfachere Syntax, basierend auf Schlüssel-Wert-Paaren, ähnlich den Objekten in JavaScript. Es unterstützt keine Attribute oder Kommentare direkt in den Daten. JSON wird aufgrund seiner Einfachheit und Kompaktheit oft für Web APIs und Konfigurationsdateien verwendet.

Prüfung Syntax und Struktur: Schema **XML:** XML bietet Schemata zur Validierung und Strukturierung von XML-Dateien. XML-Schema (XSD) und DTD (Document Type Definition) sind zwei gängige Mechanismen zur Definition und Validierung von XML-Dokumenten. XML-Schema bietet eine sehr umfassende und strenge Validierung mit Unterstützung für Datentypen, Referenzen und komplexe Strukturen (Elemente und Attribute). **JSON:** Ist schemafrei per Design, was es flexibel macht, aber weniger robust in Bezug auf die Validierung. Es gibt jedoch mit JSON Schema, eine separate Spezifikation, um die Struktur und den Datentyp von JSON-Dokumenten zu beschreiben. JSON Schema ist weniger streng als XML Schema.

Lesbarkeit **XML:** Kann als weniger menschenlesbar angesehen werden, insbesondere bei komplexen Dokumenten, da es mehr Zeichen für die Markierung (Tags und Attribute) benötigt. **JSON:** Wird im Allgemeinen als leichter lesbar angesehen, besonders für diejenigen, die mit JavaScript vertraut sind, da es weniger Verbose ist.

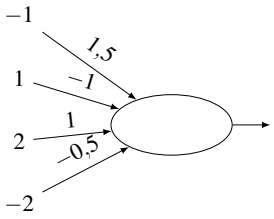
Datenanalyse und -verarbeitung **XML:** Das Parsen von XML kann aufgrund seiner Komplexität und Flexibilität langsamer und ressourcenintensiver sein. **JSON:** Wird aufgrund seiner einfachen Struktur im Allgemeinen schneller und mit weniger Ressourcenaufwand geparst.

Unterstützung durch Programmiersprachen **XML:** Erfordert in den meisten Fällen Bibliotheken oder Module, um geparst und manipuliert zu werden. **JSON:** Wird nativ von JavaScript unterstützt und von vielen anderen Programmiersprachen, oft ohne die Notwendigkeit zusätzlicher Bibliotheken.

Zeiger / Referenzen **XML:** XML unterstützt Referenzen auf andere Elemente mithilfe von IDs und den Attributen ID und IDREF bzw. IDREFS. Diese Mechanismen sind in XML definiert, um auf Elemente innerhalb des gleichen Dokuments oder auf externe Elemente zu verweisen. **JSON:** JSON selbst unterstützt keine Zeiger / Referenzen wie Programmiersprachen. Dennoch können Referenzen durch Konventionen wie das Verwenden von IDs implementiert werden.

16 Lösungen zu Kapitel 18 – Maschinelles Lernen: Deep Learning mit neuronalen Netzen

A 18.1 (M1) Berechnen Sie die Ausgabe des folgenden Neurons für die Aktivierungsfunktionen Sprungfunktion, Sigmoid, ReLU:



Lösung

In jedem Fall: Zuerst die gewichtete Summe der Eingänge bilden:

$$s = 1{,}5 \cdot (-1) + (-1) \cdot 1 + 1 \cdot 2 + (-0{,}5) \cdot (-2) = 0{,}5 \quad .$$

Sprungfunktion liefert den Wert 1.
Sigmoid liefert

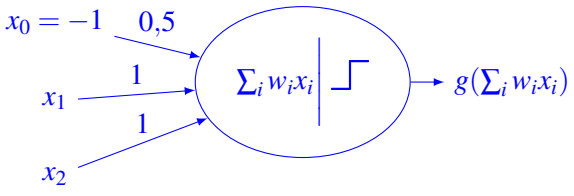
$$g_s(0{,}5) = \frac{1}{1 + e^{-0{,}5}} = 0{,}62 \quad .$$

ReLU liefert den Wert 0,5.

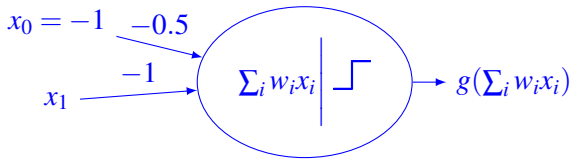
A 18.2 (M2) Beispiel 18.1, S. 820 zeigt ein Neuron zur Berechnung der logischen AND-Funktion. Geben Sie an, wie ein Neuron für OR und eines für NOT aussehen würde.

Lösung

OR:



NOT:



A 18.3 (L2) Gegeben sei ein kleines Convolutional Neural Network (CNN) mit folgender Struktur:

- Eingang: RGB-Bilder der Größe 28×28
- Faltungsschicht mit 32 Filtern der Größe 3×3, Stride 3, Zero-Padding mit 1
- Faltungsschicht mit 64 Filtern der Größe 3×3, Stride 1, Zero-Padding mit 1
- Max-Pooling der Größe 2×2, Stride 2
- Flattening
- Voll verbundene Schicht mit 128 Neuronen
- Ausgang: Voll verbundene Schicht mit 10 Neuronen

Wie groß sind die einzelnen Schichten? Wie viele Parameter müssen in den einzelnen Schichten trainiert werden? Wie viele insgesamt?
Offensichtlich kann das Netz 10 Klassen unterscheiden, z. B. die Ziffern 0 bis 9. Welche Aktivierungsfunktion würden Sie für die Neuronen am Ausgang verwenden?

Lösung

Aufgelistet sind die einzelnen Gewichtsschichten:

- $28 \times 28 \rightarrow 10 \times 10 \times 32$: $(3 \cdot 3 \cdot 3) \cdot 32 + 32 = 896$ Gewichte.
- $10 \times 10 \times 32 \rightarrow 10 \times 10 \times 64$: $(3 \cdot 3 \cdot 32) \cdot 64 + 64 = 18496$ Gewichte.
- $10 \times 10 \times 64 \rightarrow 5 \times 5 \times 64$: keine Gewichte.
- $5 \times 5 \times 64 \rightarrow 1 \times 1600$: keine Gewichte.
- $1600 \rightarrow 128$: $1600 \cdot 128 + 128 = 204928$ Gewichte.
- $128 \rightarrow 10$: $128 \cdot 10 + 10 = 1290$ Gewichte.

Gesamt: 225610 Gewichte.
Sind die Klassen disjunkt (wie bei den Ziffern) sollte in der Ausgabeschicht Softmax verwendet werden; bei nicht-disjunkten Klassen Sigmoide.