Technische Universität Darmstadt
Fachgebiet Theoretische Informatik

Prof. Johannes Buchmann
Fateme Shirazi
Johannes Braun
Dr. Albrecht Petzoldt

Grundlagen der
Informatik 2
WS 2014/15

TECHNISCHE UNIVERSITÄT DARMSTADT

Klausurlösung

Name: ..... Matrikelnummer: .....

Studiengang: ..... Diplom ☐ Bachelor ☐ Master ☐

Unterschrift: .....

Viel Erfolg!

Aufgabe	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	Σ
Mögliche Punkte	8	9	8	10	9	11	9	6	6	7	7	10	100
Erreichte Punkte													

Allgemeines

- Halten Sie Ihren Studienausweis und einen Lichtbildausweis zur Kontrolle bereit.
- Füllen Sie das Deckblatt vollständig aus.
- Prüfen Sie, ob die Klausur 12 Aufgaben enthält.
- Verwenden Sie für jede Aufgabe ein neues Blatt.
- Leerblätter werden von der Aufsicht gestellt. Verwenden Sie kein eigenes Papier.
- Geben Sie die verwendeten Formeln, Sachverhalte und Zwischenergebnisse an.

Bewertung

- Unleserlichkeit führt zu Punktabzug.
- Diese Klausur wird nur gewertet, falls die Semesterleistung ordnungsgemäß erbracht wurde.

Dauer der Klausur und zugelassene Hilfsmittel

- Ihnen stehen 120 Minuten zum Bearbeiten der Aufgaben zur Verfügung.
- Einzige zugelassene Hilfsmittel sind ein nicht programmierbarer Taschenrechner ohne Formelspeicher und ein beidseitig handschriftlich beschriebenes DIN-A4 Blatt (keine Kopien etc.).
- Bitte geben Sie andere elektronische Geräte (Handys, PDAs, Laptops, programmierbare Taschenrechner) der Klausuraufsicht zur Verwahrung.
- Studierende, deren Muttersprache nicht Deutsch ist, können zusätzlich ein zweisprachiges Wörterbuch verwenden.
- Die Klausuraufsicht überprüft die Hilfsmittel.

K1 Verschiedenes (8 Punkte)

a) Ergänzen Sie den folgenden Text.

Ein azyklischer Graph mit n Knoten hat höchstens _____ Kanten.

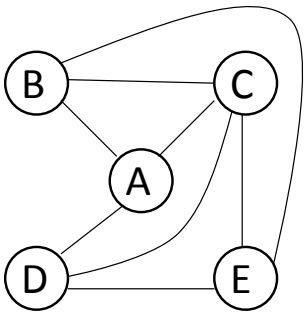
Ein bipartiter Graph mit n Knoten hat höchstens _____ Kanten.

Ein vollständiger trinärer Baum der Höhe h hat genau _____ Knoten.

b) Gegeben seien zwei Polynome A und B vom Grad n. Geben Sie die Laufzeit folgender Algorithmen in Abhängigkeit von n an (Groß O-Notation).

Polynomaddition $A+B$	
Evaluation von A (mittels Hornerschema)	
Polynommultiplikation $A \cdot B$ (mittels FFT)	

c) Ist auf untenstehendem Graphen eine Eulertour möglich? Anfangs- und Endpunkt müssen dabei nicht notwendigerweise übereinstimmen. Begründung?



Lösung.

a) $n-1$
$\frac{n^2}{4}$ bzw. $\lfloor \frac{n^2}{4} \rfloor$
$\frac{3^{h+1}-1}{2}$
(jeweils 1 Punkt)

b)

Polynomaddition $A+B$	$O(n)$
Evaluation von A (mittels Hornerschema)	$O(n)$
Polynommultiplikation $A \cdot B$ (mittels FFT)	$O(n \cdot \log n)$

(jeweils 1 Punkt)

c) Nein. Eine Eulertour ist genau dann möglich, wenn es entweder 0 oder 2 Knoten mit ungerader Kantenzahl gibt. Mit A, B, D und E gibt es aber 4. (2 Punkte)

K2 Algorithmen (9 Punkte)

Betrachten Sie den folgenden Algorithmus Selection-Sort. Der Algorithmus erhält als Input ein Array A und überschreibt dieses mit dem sortierten Array.

```
Selection-Sort(A)
1  n = length(A)
2  for j = 1 to n − 1 do
3     smallest = j
4     for i = j + 1 to n do
5        if A[i] < A[smallest] then
6           smallest = i
7     exchange A[j] and A[smallest]
```

Beweisen Sie die Korrektheit des Algorithmus Selection-Sort. Geben Sie dazu zunächst eine geeignete Schleifeninvariante für die äußere for-Schleife (Zeile 2 -7) an.

Bei Ihrer Argumentation können Sie der Einfachheit halber annehmen, dass alle Elemente des Arrays A verschieden sind.

Lösung. Schleifeninvariante: Zu Beginn jeder Iteration der Schleife enthält das Subarray A[1..j-1] die $j-1$ kleinsten Elemente des Arrays A in sortierter Reihenfolge. (2 Punkte)

Initialisierung (j=1): Für $j = 1$ ist das Subarray A[1..j-1] leer. Es ist nichts zu zeigen. (1 Punkt)

Erhaltung ($j \to j+1$): In der inneren for-Schleife wird die Variable smallest auf das kleinste Element des Subarrays A[j..n] gesetzt. Anschließend wird A[j] mit A[smallest] vertauscht. Also ist A[j] nach dem Schleifendurchgang das kleinste Element des Subarrays A[j..n]. (2 Punkte)
Nach Induktionsvoraussetzung enthielt das Subarray A[1..j-1] die kleinsten j-1 Elemente des Arrays A. Insbesondere sind also alle Elemente des Subarrays A[1..j-1] kleiner als A[j]. Das Subarray A[1..j] ist also sortiert und enthält wegen der Minimalität von A[j] im Subarray A[j..n] die j kleinsten Elemente des Arrays A. (2 Punkte)

Beendigung (j=n): Das Subarray A[1..n-1] enthält die $n-1$ kleinsten Elemente des Arrays A in sortierter Reihenfolge. Damit ist A[n] automatisch das größte Element des Arrays. Somit ist auch das ganze Array A sortiert. (2 Punkte)

K3 Asymptotik (8 Punkte)

a) Vervollständigen Sie die folgende Definition.

$f(n) = \Omega(g(n)) \Leftrightarrow$ _____ $c \in \mathbb{R}_+$ _____ $n_0 \in \mathbb{N}$ so dass _____ $\forall n \geq n_0$.

b) Vervollständigen Sie folgende Tabelle. Kreuzen Sie dabei an, ob $f(n) = \omega(g(n))$, $f(n) = \Theta(g(n))$ oder $f(n) = o(g(n))$ gilt. In jeder Halbzeile ist nur eine Lösung richtig.

$f(n)$	$g(n)$	ω	Θ	o	$f(n)$	$g(n)$	ω	Θ	o
$3 \cdot \log(n^2)$	$0.5 \cdot (\log n)^2$				$2 \cdot n^2$	$e^{3 \log n}$			
e^{n^2}	$e^{5 \cdot n}$				$5 \cdot \log(3 \cdot n)$	$4 \cdot (\log n^2)$			
$2 \cdot e^{2 \cdot \log n}$	$3^{\sqrt{n}}$				n^4	$e^{2 \cdot \log \log n}$			

Lösung.

a) $f(n) = \Omega(g(n)) \Leftrightarrow \underline{\exists}\ c \in \mathbb{R}_+$, $\underline{\exists}\ n_0 \in \mathbb{N}$ so dass $\underline{f(n) \geq c \cdot g(n)}$ $\forall n \geq n_0$. (2 Punkte)

b)

$f(n)$	$g(n)$	ω	Θ	o	$f(n)$	$g(n)$	ω	Θ	o
$3 \cdot \log(n^2)$	$0.5 \cdot (\log n)^2$			X	$2 \cdot n^2$	$e^{3 \log n}$			X
e^{n^2}	$e^{5 \cdot n}$	X			$5 \cdot \log(3 \cdot n)$	$4 \cdot (\log n^2)$		X	
$2 \cdot e^{2 \cdot \log n}$	$3^{\sqrt{n}}$			X	n^4	$e^{2 \cdot \log \log n}$	X		

(jeweils 1 Punkt; insgesamt 6 Punkte)

K4 Sortieren (10 Punkte)

Illustrieren Sie die Operation von Partition (Subroutine von Quicksort) auf dem Array $A[p \ldots r] = \langle 17, 11, 3, 9, 22, 15 \rangle$. Der Algorithmus durchläuft in einer for-Schleife nacheinander die Elemente des Arrays. Geben Sie den Zustand des Arrays A nach jedem Durchlauf der for-Schleife an. Als Pivotelement wird dabei $A[r]$ gewählt. Der Zähler i wird immer dann erhöht, wenn das gerade betrachtete Element kleiner als das Pivotelement ist. Geben Sie den Wert von i nach jeder Iteration der for-Schleife an (inklusive des Wertes von i vor dem ersten Schleifendurchgang). Geben Sie darüber hinaus den Rückgabewert der Funktion an.

$$
\begin{array}{cc}
\langle 17, 11, 3, 9, 22, 15 \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i = \\
\langle \quad , \quad , \quad , \quad , \quad , \quad \rangle & i =
\end{array}
$$

Rückgabewert:

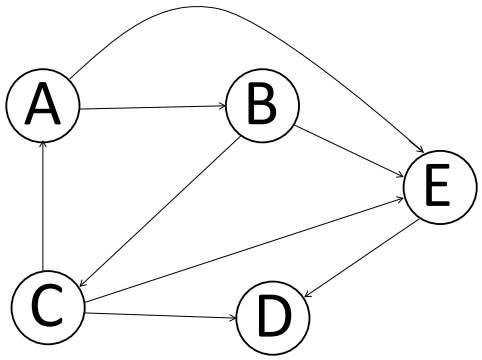
Lösung.

$$
\begin{array}{ll}
\langle 17, 11, 3, 9, 22, 15 \rangle & i = p - 1 \\
\langle 17, 11, 3, 9, 22, 15 \rangle & i = p - 1 \\
\langle 11, 17, 3, 9, 22, 15 \rangle & i = p \\
\langle 11, 3, 17, 9, 22, 15 \rangle & i = p + 1 \\
\langle 11, 3, 9, 17, 22, 15 \rangle & i = p + 2 \\
\langle 11, 3, 9, 17, 22, 15 \rangle & i = p + 2 \\
\langle 11, 3, 9, 15, 22, 17 \rangle & i = p + 2
\end{array}
$$

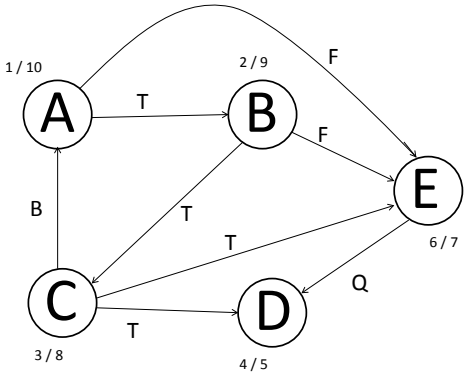
Rückgabewert: $p + 3$
(jeweils 1 Punkt für das korrekte Array + 1/2 Punkt für den richtigen Wert von i; insgesamt 10 Punkte)

K5 Tiefensuche (9 Punkte)

Führen Sie auf untenstehendem Graphen eine Tiefensuche durch. Benutzen Sie dabei Knoten A als Startknoten. Geben Sie für jeden Knoten discovery und finishing time an. Führen Sie darüber hinaus eine Kategorisierung der Kanten durch, indem Sie zu jeder Kante den jeweiligen Kantentyp (Baumkante (T), Vorwärtskante (F), Rückkante (B), Querkante (Q)) angeben.



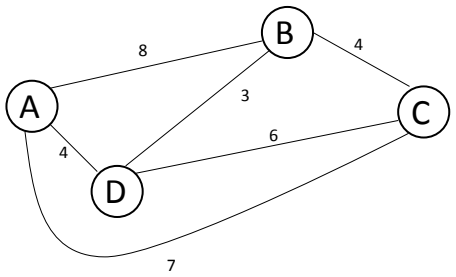
Lösung.



(jeweils 1/2 Punkt pro richtiger Zahl/Buchstabe; insgesamt 9 Punkte)

K6 Minimale aufspannende Bäume (11 Punkte)

Finden Sie für den folgenden Graphen einen minimalen aufspannenden Baum. Benutzen Sie den Algorithmus von Prim. Benutzen Sie den Knoten A als Startknoten. Der Algorithmus durchläuft in einer Schleife alle Knoten des Graphen. Tragen Sie nach jedem Schleifendurchlauf die Werte von $key[v]$ (Abstand von v zur bisherigen Zusammenhangskomponente), $\pi[v]$ (Vorgänger) (für alle Knoten v), Q (noch nicht abschließend betrachtete Knoten) und u (gerade abgeschlossener Knoten) in die folgende Tabelle ein. Jede Tabellenzeile entspricht dabei einem Iterationsschritt.
Geben Sie den minimalen aufspannenden Baum an, indem Sie die entsprechenden Kanten im untenstehenden Graphen markieren.

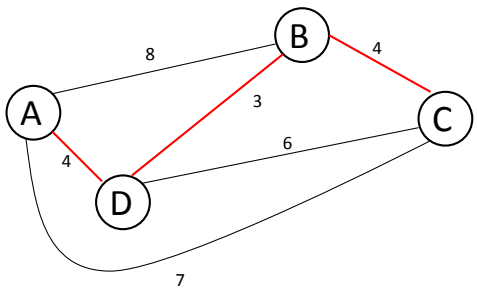


$key[A]$	$key[B]$	$key[C]$	$key[D]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	Q	u
0	∞	∞	∞	nil	nil	nil	nil	$\{A,B,C,D,E\}$	–

Lösung.

$key[A]$	$key[B]$	$key[C]$	$key[D]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	Q	u
0	∞	∞	∞	nil	nil	nil	nil	$\{A,B,C,D\}$	–
	8	7	4		A	A	A	$\{B,C,D\}$	A
	3	6			D	D		$\{B,C\}$	D
		4				B		$\{C\}$	B
								$\emptyset$	C

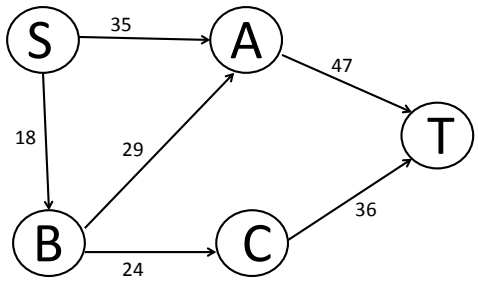
(1/2 Punkt pro Eintrag ab Zeile 2; insgesamt 10 Punkte)



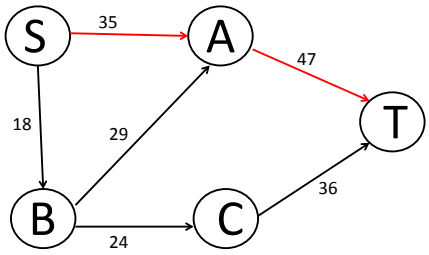
(1 Punkt fürs Bild)

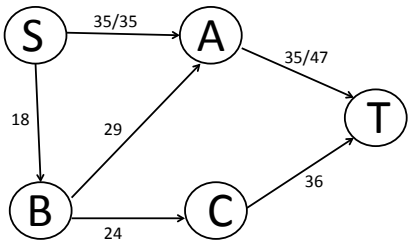
K7 Maximale Flüsse (9 Punkte)

Bestimmen Sie mit dem Algorithmus von Ford-Fulkerson den maximalen Fluss in untenstehendem Flussnetzwerk mit Quelle S und Senke T. Der Algorithmus fügt in jedem Schritt einen ergänzenden Pfad zum aktuellen Flussnetzwerk hinzu. Dies wird so lange wiederholt, bis es keinen ergänzenden Pfad mehr gibt.
Geben Sie nach jedem Schritt den gerade hinzugefügten ergänzenden Pfad, seine Kapazität und den aktuellen Gesamtfluss an. Skizzieren Sie außerdem das entstehende Restnetzwerk. Fügen Sie dabei stets den ergänzenden Pfad mit maximaler Kapazität hinzu.
Beweisen Sie die Maximalität des gefundenen Flusses, indem Sie einen minimalen Cut in das in der Aufgabenstellung angegebene Netzwerk einzeichnen.



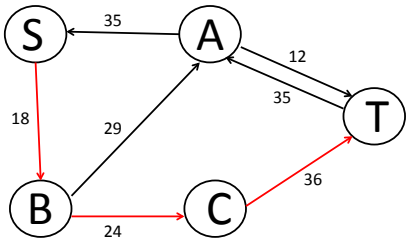
Lösung. Pfad S –> A –> T mit Kapazität 35 ⇒ Gesamtfluss $|f| = 35$ (1 Punkt)

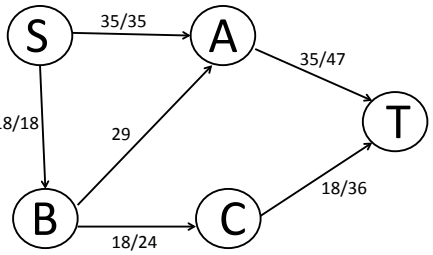




(3 Punkte)

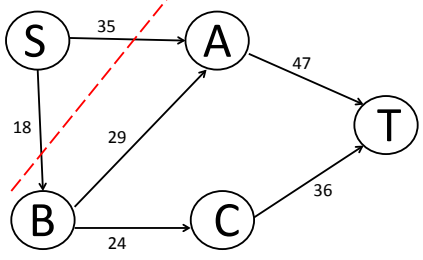
Pfad S –> B –> C –> T mit Kapazität 18 ⇒ Gesamtfluss $|f| = 53$ (1 Punkt)





(3 Punkte)

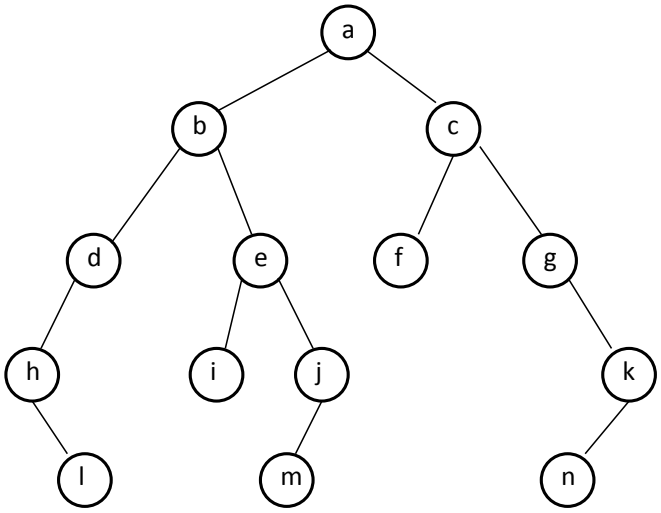
alternativ: Pfad S –> B –> A –> T mit Kapazität 12 ⇒ Gesamtfluss $|f| = 47$
+ Pfad S –> B –> C –> T mit Kapazität 6 ⇒ Gesamtfluss $|f| = 53$
Minimaler Cut:



(1 Punkt)

K8 Binäre Suchbäume (6 Punkte)

Betrachten Sie den folgenden binären Baum. Die im Baum enthaltenen Schlüssel $a, \ldots, n$ erfüllen die binäre Suchbaumeigenschaft.



a) Geben Sie den kleinsten der im Baum enthaltenen Schlüssel an.

b) Geben Sie den Nachfolger von b bzw. l an.

c) Geben Sie den Vorgänger von n bzw. a an.

d) Geben Sie den fünftgrößten der im Baum enthaltenen Schlüssel an.

Lösung.

a) h

b) i, d

c) g, j

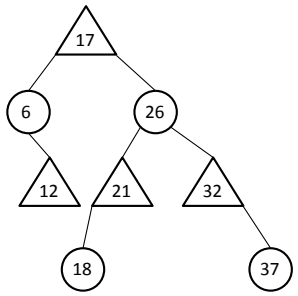
d) f

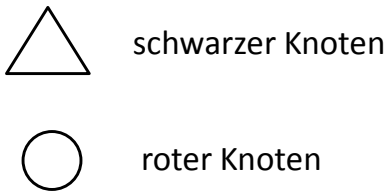
(jeweils 1 Punkt; insgesamt 6 Punkte)

K9 Rot-Schwarz-Bäume (6 Punkte)

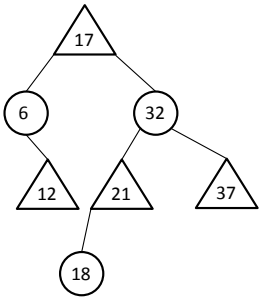
Führen Sie auf untenstehendem RB-Baum die Operationen Rb-Delete(26) und RB-Delete(37) in dieser Reihenfolge durch. Skizzieren Sie den RB-Baum nach jedem Löschvorgang und geben Sie jeweils an, ob und welche Fälle der Methode RB-Delete-Fixup aufgerufen werden.

Stellen Sie dabei rote und schwarze Knoten wie in der Abbildung gezeigt dar.



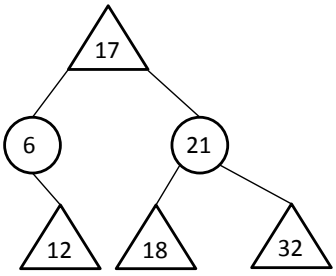


Lösung. Delete(26): 26 wird durch seinen Nachfolger 32 ersetzt. RB-Delete-Fixup wird mit dem Knoten x=37 aufgerufen. Knoten 37 wird schwarz gefärbt.



(1 Punkt für Beschreibung + 2 fürs Bild; pro Fehler (falsche Farbe etc.) ein Punkt Abzug)

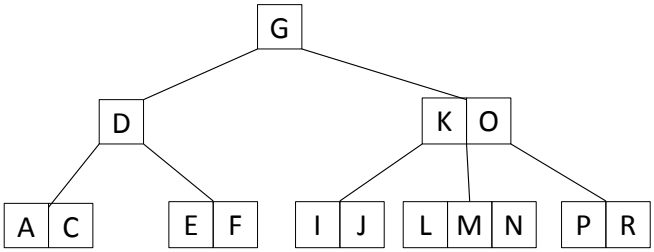
Delete(37): RB-Delete-Fixup wird mit x=NIL aufgerufen. Es tritt Fall 4 auf.



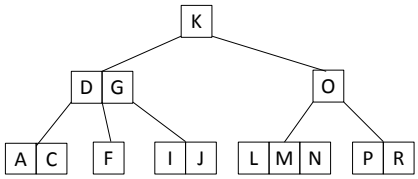
(1 Punkt für Beschreibung + 2 fürs Bild; pro Fehler (falsche Farbe etc.) ein Punkt Abzug)

K10 B-Bäume (7 Punkte)

Gegeben sei der untenstehende B-Baum vom Minimalgrad 2. Löschen Sie die Schlüssel E und O in dieser Reihenfolge aus dem Baum.
Skizzieren Sie den entstehenden B-Baum nach jedem Löschvorgang und geben Sie an, welche Fälle beim Löschen auftreten.

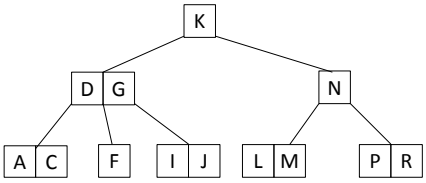


Lösung.
E löschen: Knoten D enthält nur 1 Schlüssel. Damit tritt Fall 3a auf: G kommt von der Wurzel in Knoten DG, K kommt in die Wurzel.
Anschließend tritt Fall 1 auf: E wird aus dem Blatt EF gelöscht.



(2 Punkte für Beschreibung + 2 fürs Bild)

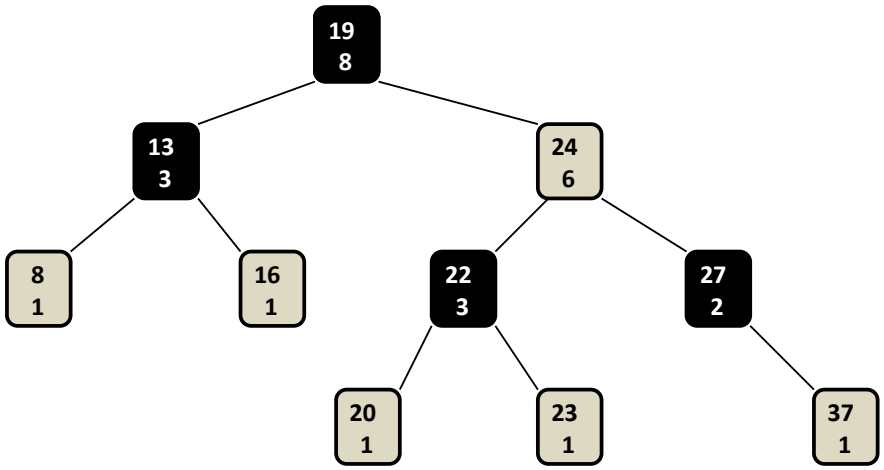
O löschen: Knoten O enthält nur 1 Schlüssel. Damit tritt Fall 2a auf: O wird durch seinen Vorgänger N ersetzt.



(1 Punkt für Beschreibung + 2 fürs Bild)

K11 OS-Bäume (7 Punkte)

Gegeben sei der untenstehende OS-Baum T mit 10 Knoten. Bestimmen Sie den Rang des Schlüssels 22 sowie den zweitgrössten der im Baum enthaltenen Schlüssel mit Hilfe der Funktionen OS-Rank und OS-Select. Geben Sie an, welcher Wert der Funktion OS-Select übergeben wird und füllen Sie die untenstehenden Tabellen aus. Geben Sie ferner die Rückgabewerte der Funktionen an.



OS-Rank(T,22)

Iteration	y	r

Rückgabewert:

OS-Select(root(T),_____)

Iteration	x	i	r

Rückgabewert:

Lösung.

OS-Rank(T,22)

Iteration	y	r
0	22	2
1	24	2
2	19	6

Rückgabewert: 6

(1/2 Punkt für erste Zeile; je 1 Punkt für Zeile 2 und 3; die erste Spalte ist nicht so wichtig; 1/2 Punkt für den Rückgabewert)

In einem Baum mit 10 Schlüsseln ist der zweitgrößte der neuntkleinste Schlüssel. Der Rang des gesuchten Schlüssels und damit der Input von OS-Select ist also 9. (1 Punkt)

OS-Select(root(T),9)

Iteration	x	i	r
0	19	9	4
1	24	5	4
2	27	1	1

Rückgabewert: 27

(1/2 Punkt für erste Zeile; je 1 Punkt für Zeile 2 und 3; die erste Spalte ist nicht so wichtig; 1/2 Punkt für den Rückgabewert)

K12 Fouriertransformation (10 Punkte)

Berechnen Sie die diskrete Fouriertransformation des Vektors $a = (1, 2, 1, 3)$.

Lösung. Da die Länge des Vektors a vier ist, benötigen wir die vierten Einheitswurzeln $\omega_4^0 = 1, \omega_4^1 = i, \omega_4^2 = -1, \omega_4^3 = -i$ (1 Punkt). Der Vektor y ergibt sich nach der diskreten Fouriertransfomation als

$$y_k = \sum_{j=0}^{n-1} a_j \omega_n^{jk} \ (k = 0, \ldots, 3). \ \ (2\ Punkte)$$

Damit ergibt sich $a_0 = \sum_{i=0}^{3} y_j = 7$. (1 Punkt)
$a_1 = 1 + 2i + 1i^2 + 3i^3 = -i$. (2 Punkte)
$a_2 = 1 + 2i^2 + 1i^4 + 3i^6 = -3$. (2 Punkte)
$a_3 = 1 + 2i^3 + 1i^6 + 3i^9 = i$. (2 Punkte)

Damit haben wir $y = (7, -1, -3, i)$.