Technische Universität Darmstadt
Fachgebiet Theoretische Informatik

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

Grundlagen der

Informatik 2

WS 2013/2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Probeklausur -Lösung

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

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

Fachbereich: .... Fachsemester: ......

Prüfungssekretariat/Büro: .... None ☐

Studienleistung erbracht? .... Wenn ja, wann? ....

Wiederholung? ☐ Versuch Nummer: ...... Jahr des letzten Versuchs: ....

Unterschrift: ....

Aufgabe	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	Σ
Mögliche Punkte	10	10	8	12	8	11	10	13	9	11	8	10	120
Erreichte Punkte													

Hinweise

- Diese Probeklausur hat den selben Umfang wie die Semestralklausur am 18. September. Sie sollten daher versuchen, die Aufgaben in einer Zeit von 120 Minuten zu bearbeiten.
- Um eine möglichst klausurnahe Atmosphäre zu erreichen, wird empfohlen, bei der Bearbeitung der Aufgaben nicht auf Bücher oder sonstige Mitschriften zurückzugreifen. Statt dessen empfehlen wir, bei dieser Probeklausur ein doppelseitig handbeschriebenes DIN-A4 Blatt zu verwenden, wie es auch bei der Semestralklausur als Hilfsmittel erlaubt ist.
- Für diese Probeklausur wird im moodle eine Musterlösung zur Selbstkorrektur bereitgestellt. Neben den Lösungshinweisen enthält diese auch eine ausführliche Beschreibung der Punktevergabe. Sie können also selbst ausrechnen, wie viele Punkte Sie erhalten hätten. Weiterhin wird ein Notenspiegel bereit gestellt, der Ihnen helfen soll, Ihre Leistung einzuschätzen.
- Anmerkung zur Bewertung: Bei der Semestralklausur werden bei der Bewertung Folgefehler berücksichtigt. Es ist allerdings nicht möglich, innerhalb dieser Musterlösung konkrete Bewertungsrichtlinien für diesen Fall anzugeben, da die Bewertung von Folgefehlern stark vom Einzelfall abhängt.

K1 Verschiedenes (10 Punkte)

a) Sei $G = (V, E)$ ein ungerichteter Graph. Nennen sie zwei Bedingungen, die jeweils zu der Aussage "G ist ein Baum“ äquivalent sind.

b) Geben Sie die worst-case Laufzeit der folgenden Algorithmen in Abhängigkeit von der Eingabelänge n an.

Insertion-Sort	
Merge-Sort	
Quicksort	

c) Vervollständigen Sie folgenden Text.

Sei x ein Knoten in einem binären Suchbaum.
Ist y ein Knoten im linken Teilbaum von x, so gilt key$[y]$ _______ key$[x]$. (Wählen Sie hierbei aus $<$, $\leq$, $=$, $\geq$ und $>$.)

Ist y ein Knoten im rechten Teilbaum von x, so gilt key$[x]$ ________ key$[y]$. (Wählen Sie hierbei aus $<$, $\leq$, $=$, $\geq$ und $>$.)

Die Zahl der Schlüssel in einem inneren Knoten eines B-Baums mit Minimalgrad t ist mindestens __________

und höchstens __________.

Die Höhe eines B-Baums mit n Knoten und Minimalgrad t ist nach oben durch ____________ beschränkt.

Lösung.

a) Es müssen zwei von 5 Möglichkeiten genannt werden

- Je zwei Knoten in G sind durch genau einen Pfad verbunden.
- G ist zusammenhängend. Aber wenn eine Kante aus G entfernt wird, ist G nicht mehr zusammenhängend.
- G ist zusammenhängend und $|E| = |V| - 1$.
- G ist azyklisch und $|E| = |V| - 1$.
- G ist azyklisch. Aber wenn eine Kante zu G hinzugefügt wird, enthält G einen Zyklus.

b)

Insertion-Sort	$O(n^2)$
Merge-Sort	$O(n \log n)$
Quicksort	$O(n^2)$

c)

Sei x ein Knoten in einem binären Suchbaum. Ist y ein Knoten im linken Teilbaum von x, so gilt $\text{key}(y) \leq \text{key}(x)$.
Ist y ein Knoten im rechten Teilbaum von x, so gilt $\text{key}(x) \leq \text{key}(y)$.
Die Zahl der Schlüssel in einem inneren Knoten eines B-Baums mit Minimalgrad t ist mindestens $t-1$ und höchstens $2t-1$.
Die Höhe eines B-Baums mit n Knoten und Minimalgrad t ist nach oben durch $h \leq \log_t \frac{n+1}{2}$ beschränkt.

K2 Asymptotik (10 Punkte)

a) Vervollständigen Sie die folgenden Definitionen.

$$f(n) = \Omega(g(n)) \Longleftrightarrow \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N} \text{ so dass } \forall n \geq n_0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$$

$$f(n) = o(g(n)) \Longleftrightarrow \forall \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \exists \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \text{ so dass } \forall n \geq n_0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_$$

b) Kreuzen Sie in der Tabelle an, ob $f = o(g)$, $f = \omega(g)$ oder $f = \Theta(g)$ gilt. In der Tabelle bezeichnet $k > 0$ eine Konstante. Bei der letzten Teilaufgabe gilt $n > 0$. In jeder Halbzeile ist nur eine Lösung richtig.

f(n)	**g(n)**	o	ω	Θ	**f(n)**	**g(n)**	o	ω	Θ
$\sqrt{\log n}$	$\log \log n$				n^n	120.000^n			
$2,5 \cdot n^3$	$e^{\ln e}$				$n^{\log k}$	$k^{\log n}$			
4^n	$n!$				$\log_2 n^n$	3^{n+2}			

Lösung.

a)

$$f(n) = \Omega(g(n)) \Longleftrightarrow \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N} \text{ so dass } \forall n \geq n_0 \;\; f(n) \geq c \cdot g(n)$$

$$f(n) = o(g(n)) \Longleftrightarrow \forall c \in \mathbb{R}_{>0} \exists n_0 \in \mathbb{N} \text{ so dass } \forall n \geq n_0 \;\; f(n) \leq c \cdot g(n)$$

b)

f(n)	**g(n)**	o	ω	Θ	**f(n)**	**g(n)**	o	ω	Θ
$\sqrt{\log n}$	$\log \log n$		x		n^n	120.000^n		x	
$2,5 \cdot n^3$	$e^{\ln e}$		x		$n^{\log k}$	$k^{\log n}$			x
4^n	$n!$	x			$\log_2 n^n$	3^{n+2}	x		

K3 Rekurrenzgleichungen (8 Punkte)

Lösen sie folgende Rekurrenzgleichungen mit Hilfe des Mastertheorems.

a) $T(n) = 5T(n/8) + n \log n$

b) $T(n) = T(3n/5) + 1$

Lösung.

a) In der Notation des Mastertheorems haben wir $a = 5$, $b = 8$ und $f(n) = n \lg n$. Damit gilt $f(n) = \Theta(n \log n) = \Omega((n^{\log_b a + \epsilon}))$ für $\epsilon = 1 - \log_8 5 \approx 0,22$. Weiterhin gilt $a \cdot f(n/b) = 5 \cdot n/8 \log(n/8) \leq 5 \cdot n/8 \log(n) = 5/8 \cdot f(n)$. Damit können wir Fall 3 des Mastertheorems anwenden und erhalten $T(n) = \Theta(f(n)) = \Theta(n \log n)$.

b) Wir haben $a = 1$, $b = 5/3$ und $f(n) = 1$. Damit gilt $f(n) = \Theta(1) = \Theta(n^{\log_b a})$. Damit können wir Fall 2 des Mastertheorems anwenden und erhalten $T(n) = \Theta(n^{\log_3 3} \log n) = \Theta(\log n)$.

K4 Sortieren (12 Punkte)

a) Vervollständigen Sie den folgenden Algorithmus zu Insertion-Sort. Der Algorithmus erhält als Eingabewert ein (unsortiertes) Array A. Beachten Sie: Das Array A beginnt mit Index 1.

Insertion-Sort(A)

for j= ____ to length(A)

  key = A[j]

  i = ________

  while i>0 and ______

    ____

    i=i-1

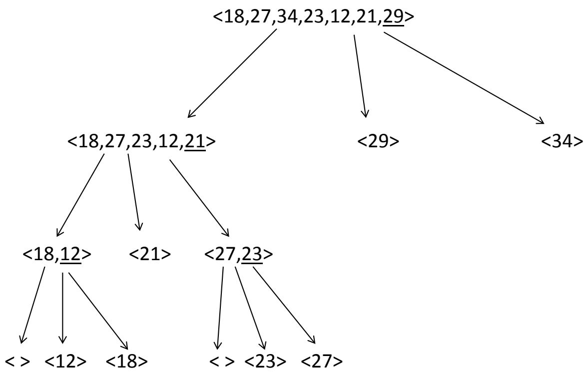
  ____

b) Demonstrieren Sie die Arbeitsweise von Quicksort anhand des Arrays $A = \langle 18, 27, 34, 23, 12, 21, 29 \rangle$. Verwenden Sie dabei immer das letzte Element eines (Teil-)Arrays als Pivotelement. Verwenden Sie die Baumstruktur, wie sie auch in den Übungen und den Vorlesungsfolien verwendet wurde. Geben Sie das sortierte Array an.

Lösung.

a) Insertion-Sort(A)
for j= **2** to length(A)
  key = A[j]
  i = j-1
  while i>0 and **A[i] $\geq$ key**
    A[i+1] =A[i]
    i=i-1
  A[i+1]=key

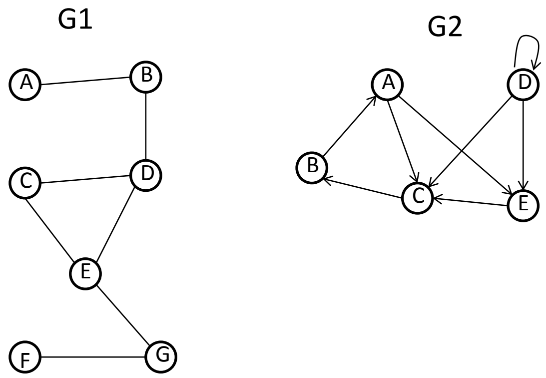
b)



Das sortierte Array hat die Form $A = \langle 12, 18, 21, 23, 27, 29, 34 \rangle$.

K5 Graphen **(8 Punkte)**

Gegeben seien die folgenden zwei Graphen.



a) Ist G1 ein Baum? Wenn nein, warum nicht? Wie kann man G1 (unter Beibehaltung der Knoten) verändern, damit es zu einem Baum wird?

b) Wie viele Möglichkeiten gibt es, G1 durch Entfernen einer Kante zu einem bipartiten Graph zu machen? Welche sind das? Geben sie jeweils die beiden disjunkten Teilmengen von Knoten an.

c) Wie viele Kanten muss man zu G2 hinzufügen, damit ein stark zusammenhängender Graph entsteht? Begründung?

Lösung.

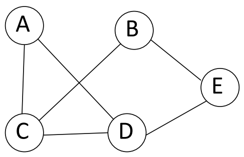
a) G1 ist kein Baum, da er nicht azyklisch ist. Durch Entfernen der Kante (C,D) kann man G1 zu einem Baum machen. (analog mit (C,E) oder (D,E)).

b) Es gibt 3 Möglichkeiten, G1 durch Entfernen einer Kante zu einem bipartiten Graph zu machen:
   - Entfernen der Kante (C,E): Es entsteht ein bipartiter Graph mit den Teilmengen (A,D,G) und (B,C,E,F).
   - Entfernen der Kante (D,E): Teilmengen (A,D,E,F) und (B,C,G).
   - Entfernen der Kante (C,D): Teilmengen (A,C,D,G) und (B,E,F).

c) Da die Knoten A,B,C und E bereits eine starke Zusammenhangskomponente bilden, muss man lediglich eine Kante hinzufügen, so dass Knoten D erreicht werden kann. Es ist egal, ob das (A,D), (B,D), (C,D) oder (E,D) ist.

K6 Breitensuche (11 Punkte)

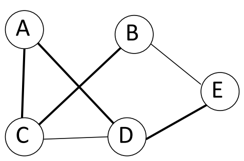
Führen Sie auf dem folgenden Graphen eine Breitensuche aus. Der Algorithmus arbeitet sämtliche Knoten nacheinander ab und färbt diese schließlich schwarz.
Tragen Sie die Werte von $d[v]$ (Abstand), $\pi[v]$ (Vorgänger) (für alle Knoten v), Q (Warteschlange) und u zu jedem Zeitpunkt, zu dem ein Knoten u schwarz gefärbt wird, in die folgende Tabelle ein. Jede Zeile entspricht dabei einem Iterationsschritt.
Benutzen Sie dabei Knoten A als Startknoten und nehmen Sie an, dass die Knoten in den Adjazenzlisten alphabetisch sortiert sind. Geben Sie den durch den Algorithmus erzeugten Vorgängerbaum an, indem Sie die entsprechenden Kanten im untenstehenden Graphen markieren.



$d[A]$	$d[B]$	$d[C]$	$d[D]$	$d[E]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	$\pi[E]$	Q	u
0	∞	∞	∞	∞	nil	nil	nil	nil	nil	$\{1\}$	–

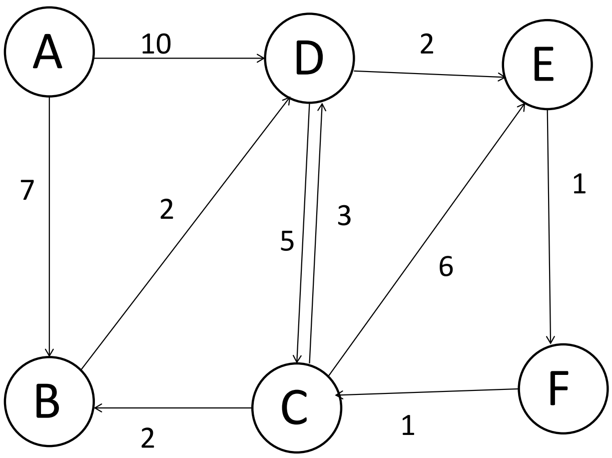
Lösung.

$d[A]$	$d[B]$	$d[C]$	$d[D]$	$d[E]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	$\pi[E]$	Q	u
0	∞	∞	∞	∞	nil	nil	nil	nil	nil	$\{A\}$	–
		1	1				A	A		$\{C,D\}$	A
	2					C				$\{D,B\}$	C
				2					D	$\{B,E\}$	D
										$\{E\}$	B
										$\emptyset$	E



K7 Dijkstra’s Algorithmus (10 Punkte)

Führen Sie Dijkstras Algorithmus auf folgendem Graphen aus. Benutzen Sie dabei den Knoten A als Startknoten. Im Verlauf des Algorithmus werden die Knoten nacheinander zur Menge S hinzugefügt. Tragen Sie die Werte von d (Abstand), π (Vorgänger) und S (bereits abschließend betrachtete Knoten) unmittelbar nachdem ein neuer Knoten zu S hinzugefügt wurde, in nachstehende Tabelle ein. Jede Tabellenzeile entspricht dabei einem Iterationsschritt.



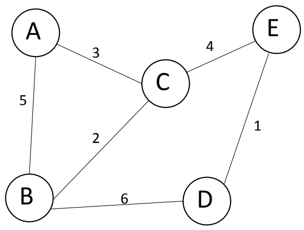
$d[A]$	$d[B]$	$d[C]$	$d[D]$	$d[E]$	$d[F]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	$\pi[E]$	$\pi[F]$	S
0	∞	∞	∞	∞	∞	nil	nil	nil	nil	nil	nil	$\emptyset$

Lösung.

$d[A]$	$d[B]$	$d[C]$	$d[D]$	$d[E]$	$d[F]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	$\pi[E]$	$\pi[F]$	S
0	∞	∞	∞	∞	∞	nil	nil	nil	nil	nil	nil	$\emptyset$
	7		10				A		A			$\{A\}$
			9						B			$\{A,B\}$
		14		11				D		D		$\{A,B,D\}$
					12						E	$\{A,B,D,E\}$
		13						F				$\{A,B,D,E,F\}$
												$\{A,B,D,E,F,C\}$

K8 Minimale aufspannende Bäume (13 Punkte)

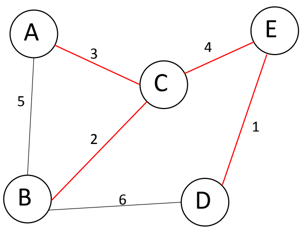
Finden Sie für den folgenden Graphen einen minimalen aufspannenden Baum. Benutzen Sie den Algorithmus von Prim. Benutzen Sie den Knoten A als Startknoten. Der Algorithmus durchläuft nacheinander alle Knoten des Graphen. Tragen Sie, immer wenn die Bearbeitung eines Knotens abgeschlossen ist, die Werte 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 unten stehenden Graphen markieren.



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

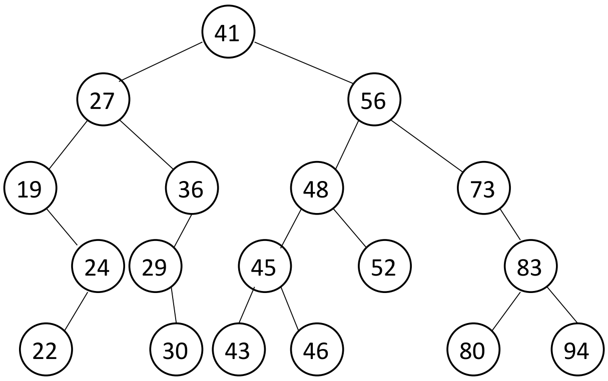
Lösung.

$key[A]$	$key[B]$	$key[C]$	$key[D]$	$key[E]$	$\pi[A]$	$\pi[B]$	$\pi[C]$	$\pi[D]$	$\pi[E]$	Q	u
0	∞	∞	∞	∞	nil	nil	nil	nil	nil	$\{A,B,C,D,E\}$	–
	5	3				A	A			$\{B,C,D,E\}$	A
	2			4		C			C	$\{B,D,E\}$	C
			6					B		$\{D,E\}$	B
			1					E		$\{D\}$	E
										$\emptyset$	D

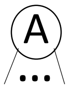


K9 Binäre Suchbäume (9 Punkte)

Betrachten Sie den folgenden binären Suchbaum.



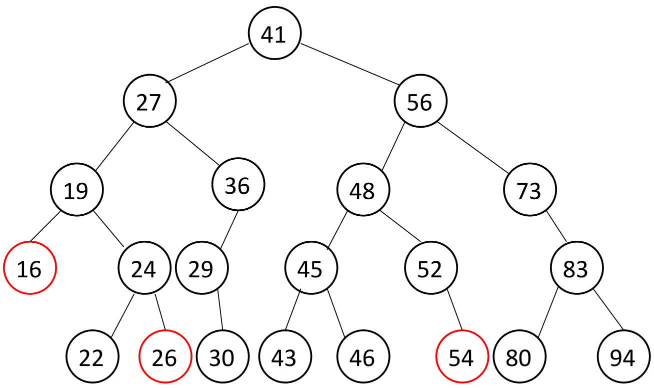
(a) Fügen Sie die Knoten z_1, z_2, z_3 mit $key[z_1] = 16, key[z_2] = 26, key[z_3] = 54$ in dieser Reihenfolge in den oben gezeigten Suchbaum ein. Skizzieren Sie den Baum, nachdem die drei Schlüssel eingefügt sind. Sie können dafür die drei Knoten in den obigen Baum einzeichnen.

(b) Löschen Sie die Knoten z_4 und z_5 mit $key[z_4] = 29$ und $key[z_5] = 48$ in dieser Reihenfolge aus dem aus a.) gewonnenen Suchbaum. Skizzieren Sie den Baum nach jedem Löschvorgang. Dabei können Sie Teilbäume mit Wurzel A, die sich nicht verändert haben, in der Form 

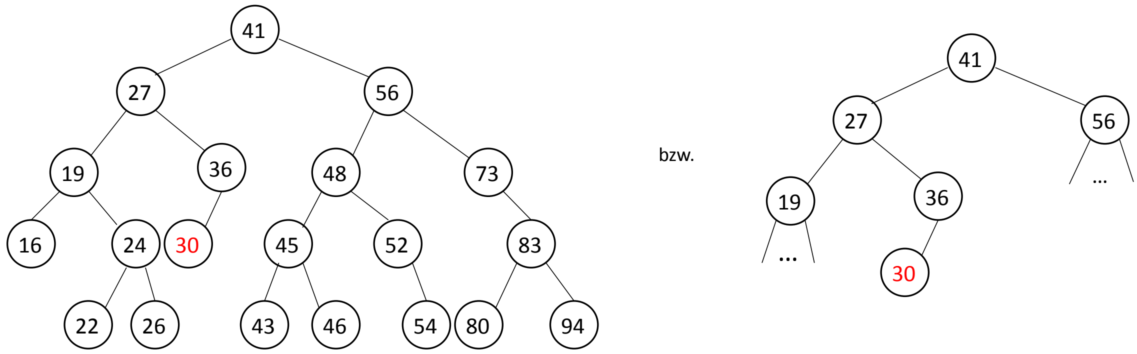
darstellen.

Lösung.

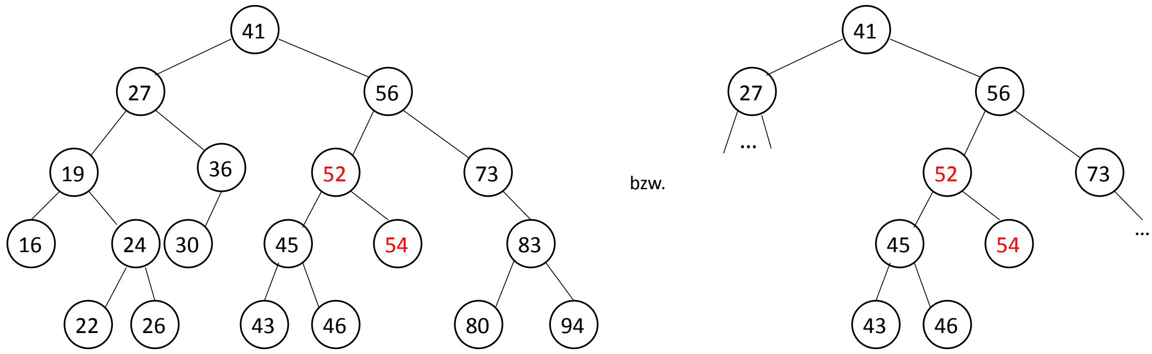
(a)



(b) 29 löschen

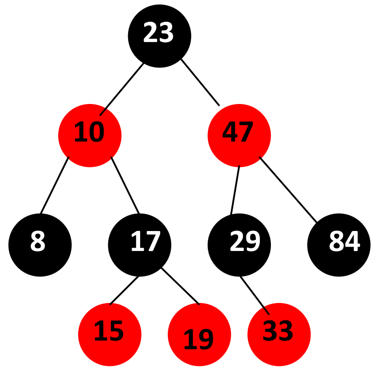


48 löschen



K10 Rot-Schwarz-Bäume (11 Punkte)

Gegeben sei der folgende Rot-Schwarz-Baum T.

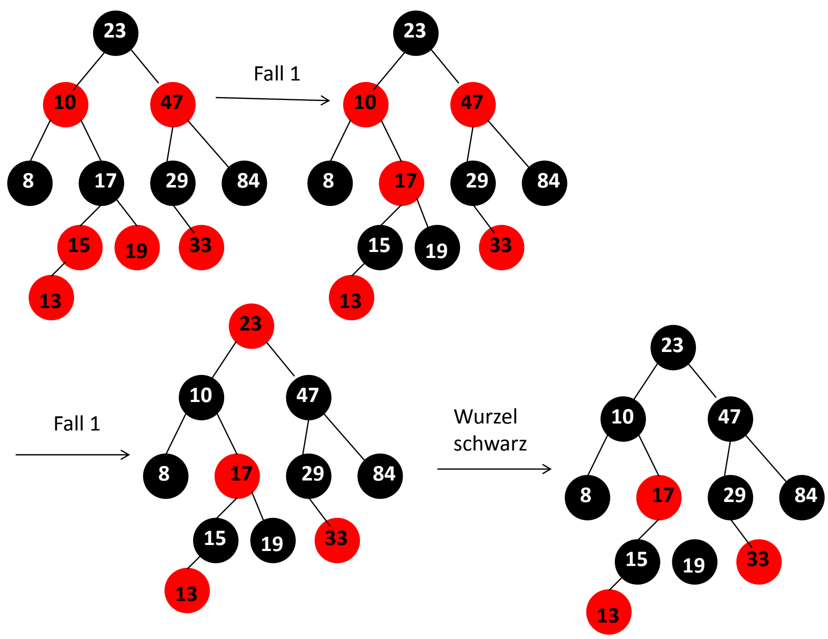


Führen Sie auf dem Baum hintereinander die Operationen RB-Insert(T,13) und RB-Delete(T,8) durch.
Geben Sie jeweils an, ob und gegebenenfalls welche Fälle von RB-Insert-Fixup bzw. RB-Delete-Fixup in welcher Reihenfolge aufgerufen werden. Skizzieren Sie den Baum nach jeder Einfüge- bzw. Löschoperation. Stellen Sie dabei rote und schwarze Knoten folgendermaßen dar.

 roter Knoten

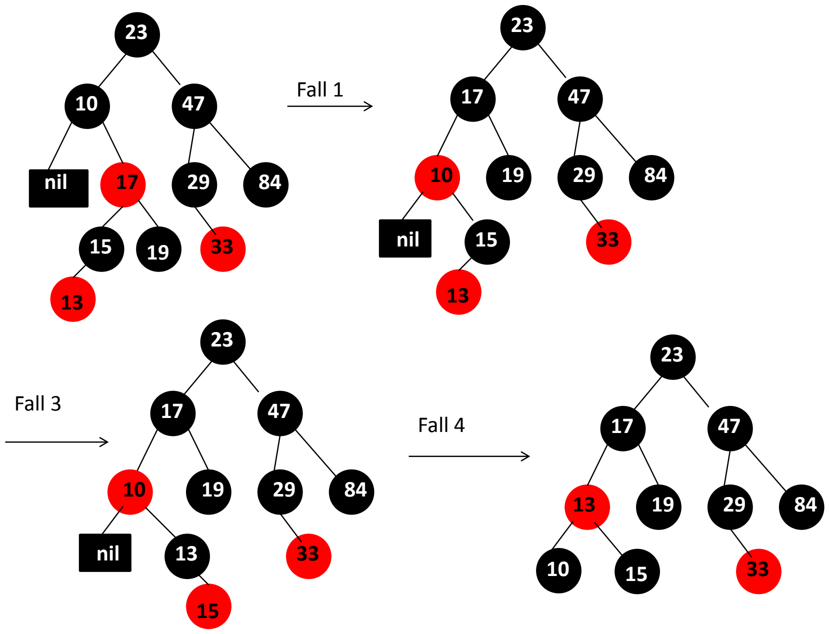
 schwarzer Knoten

Lösung. 13 einfügen



Nach dem Einfügen hat der rote Knoten 13 ein rotes Elter 15. Da sein Onkel (19) ebenfalls rot ist, wird Fall 1 von RB-Insert-Fixup aufgerufen. Anschließend hat der rote Knoten 17 ein rotes Elter 10. Da sein Onkel (47) rot ist, wird wiederum Fall 1 aufgerufen. Zuletzt muss noch die Wurzel schwarz gefärbt werden.

8 löschen



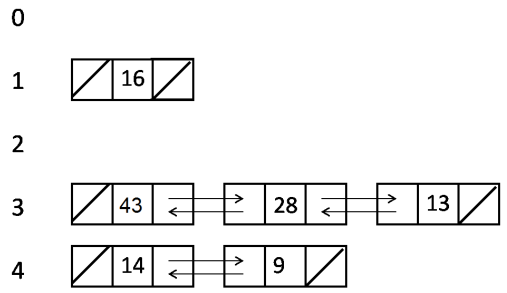
Nach dem Löschen des schwarzen Knotens 8 wird RB-Delete-Fixup mit $x = nil$ aufgerufen. Da sein Geschwister (Knoten 17) rot ist, wird zunächst Fall 1 von RB-Delete-Fixup ausgeführt. Anschließend werden noch Fall 3 und 4 durchgeführt.

K11 Hashtabellen (8 Punkte)

Fügen Sie die Schlüssel $13, 28, 16, 9, 14$ und 43 in dieser Reihenfolge in eine Hashtabelle der Länge 5 ein. Benutzen Sie doppelt verkettete Listen, um Kollisionen zu behandeln. Benutzen Sie als Hashfunktion $h(k) = k \bmod 5$.

Lösung.

$h(13) = 3$
$h(28) = 3$
$h(16) = 1$
$h(9) = 4$
$h(14) = 4$
$h(43) = 3$



K12 Diskrete Fouriertransfomation (10 Punkte)

Berechnen Sie die diskrete Fouriertransformation des Vektors $a = (0,1,2,3)$.

Lösung. Der Vektor a hat die Länge $n = 4$. Wir benötigen daher die vierten Einheitswurzeln $\omega_4^0 = 1$, $\omega_4^1 = i$, $\omega_4^2 = -1$ und $\omega_4^3 = -i$. Der Vektor $y = \mathrm{DFT}(a)$ ergibt sich aus

$$y_k = \sum_{j=0}^{3} a_j \omega_4^{kj} \ (k = 0, \ldots 3).$$

Wir haben
$y_0 = \sum_{j=0}^{3} a_j = 6$
$y_1 = 0 + i + 2i^2 + 3i^3 = -2 - 2i$
$y_2 = 0 + i^2 + 2i^4 + 3i^6 = -2$
$y_3 = 0 + i^3 + 2i^6 + 3i^9 = -2 + 2i$

Damit gilt

$$y = (6, -2 - 2i, -2, -2 + 2i).$$