Probeklausur

Informatik I - Datenstrukturen und Algorithmen

8. 8. 2002, Prof. Dr. L. Kobbelt

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Studiengang: ◯ Informatik Diplom ◯ Informatik Lehramt

◯ Sonstige _____

- Schreiben Sie auf jedes Blatt **Vorname**, **Name** und **Matrikelnummer**.
- Geben Sie Ihre Antworten in lesbarer und verständlicher Form an.
- Bitte beantworten Sie die Aufgaben auf den **Aufgabenblättern** (benutzen Sie auch die Rückseiten). Antworten auf anderen Blättern können nur berücksichtigt werden, wenn **Name, Matrikelnummer und Aufgabennummer** deutlich darauf erkennbar sind.
- Was nicht bewertet werden soll, kennzeichnen Sie bitte durch **Durchstreichen**.
- Werden Täuschungsversuche beobachtet, so wird die Klausur mit **nicht bestanden** bewertet.
- Geben Sie am Ende der Klausur **alle Blätter zusammen mit den Aufgabenblättern ab**.
- Verwenden Sie ausschließlich dokumentenechte Stifte mit schwarzer oder blauer Farbe (d.h. insbesondere sind keine Bleistifte und keine Stifte mit roter Farbe zugelassen).

	Anzahl Punkte	**Erreichte Punkte**
Aufgabe 1	**12**	
Aufgabe 2	**11**	
Aufgabe 3	**11**	
Aufgabe 4	**6**	
Aufgabe 5	**10**	
Aufgabe 6	**13**	
Aufgabe 7	**15**	
Summe	**78**	
Note	**-**	

Vorname	Name	Matr.Nr.

1
(18)

Aufgabe 1: Laufzeitanalyse

1. Betrachten Sie folgende Java-Methode:

```
public static int myst1(int b, int n) {
    int m = n;
    int p = b;
    int res = 1;
    while (m > 0) {
       if (m % 2 == 0) {
            p = p * p;
            m = m / 2;
        }
        else {
            res = res * p;
            m = m - 1;
        }
     }
     return res;
}
```

   (a) (1 Punkt) Was berechnet der Aufruf `myst1(b, n)` für natürliche Zahlen b und n? (Ohne Begründung)

   (b) (1 Punkt) Bestimmen Sie eine kleinste obere Schranke (in O-Notation) für die asymptotische Laufzeit von `myst1` für den Fall natürlicher Argumente `b` und `n`. Für elementare Operationen (arithmetische Operationen, Vergleiche, Zuweisungen) soll hierbei ein konstanter Aufwand angenommen werden. (Mit kurzer Begründung)

Vorname	Name	Matr.Nr.

2
(18)

2. Betrachten Sie folgende Java-Methode:

```
public static int myst2(int a, int b) {
   if (a == b) return a;
   else {
       if (a < b) {
          if (b - a == 1) return a;
          else return myst2(a + 1, b - 1);
       }
       else {
          if (a - b == 1) return b;
          else return myst2(a - 1, b + 1);
       }
   }
}
```

(a) (1 Punkt) Was berechnet der Aufruf `myst2(a, b)`? (Ohne Begründung)

(b) (2 Punkte) Stellen Sie eine Rekursionsgleichung auf, die die Laufzeit von `myst2` beschreibt. Für elementare Operationen (arithmetische Operationen und Vergleiche) soll hierbei ein konstanter Aufwand angenommen werden. (Ohne Begründung)

(c) (1 Punkt) Bestimmen Sie eine kleinste obere Schranke (in O-Notation) für die asymptotische Laufzeit von `myst2`. Für elementare Operationen (arithmetische Operationen und Vergleiche) soll hierbei ein konstanter Aufwand angenommen werden. (Mit kurzer Begründung)

Vorname	Name	Matr.Nr.

3
(18)

3. Seien $f, g, h : \mathbb{N} \to \mathbb{R}^+$ streng monoton wachsende Funktionen. Beweisen oder widerlegen Sie:

   (a) (2 Punkte) $f(n) = \Omega\left(g(n)\right) \implies 2^{f(n)} = \Omega(2^{g(n)})$

   (b) (2 Punkte) $f(n) = \theta\left(\sqrt{f(n^2)}\right)$

   (c) (2 Punkte) $f(n) = O\left(g(n)\right) \implies f(n)^5 = O\left(g(n)^5\right)$

Vorname	Name	Matr.Nr.

4 (18)

Aufgabe 2: Entwurfsparadigmen

Der Sattel eines Packesels hat zwei Taschen, eine links und eine rechts des Esels. Der Esel soll nun möglichst gleichmäßig mit $N \geq 1$ Gegenständen beladen werden, das heißt, der Gewichtsunterschied zwischen den beiden Taschen soll minimal werden. Wir nehmen an, dass der i-te Gegenstand ($1 \leq i \leq N$) genau $g_i \in \mathbb{N}^+$ Kilogramm wiegt.
Zur Lösung des Problems soll ein Algorithmus entwickelt werden, der auf Dynamischem Programmieren beruht. Es sei $G = \sum_{i=1}^{N} g_i$ das Gesamtgewicht aller Gegenstände. Weiter sei χ_{ij} eine binäre Variable mit $\chi_{ij} = 1$, falls sich die ersten i Gegenstände so auf die Taschen verteilen lassen, dass die Gewichtsdifferenz zwischen linker und rechter Tasche $= j$ ist, und $\chi_{ij} = 0$ sonst ($1 \leq i \leq N, -G \leq j \leq G$). Beispiel: Ist $\chi_{ij} = 1$ und wird der $(i+1)$-te Gegenstand in die linke Tasche gelegt, so folgt $\chi_{i+1,j+g_{i+1}} = 1$.

1. (1 Punkt) Geben Sie ein Beispiel für N und $g_1, \ldots, g_N$ an, bei dem sich die Gegenstände nicht so auf die Taschen verteilen lassen, dass beide Taschen gleich schwer sind. (Ohne Begründung)

2. (2 Punkte) Geben Sie die Werte für χ_{1j} an. (Ohne Begründung)

3. (3 Punkte) Geben Sie eine Rekursionsformel für χ_{ij} an. (Ohne Begründung)

Vorname	Name	Matr.Nr.

5
(18)

4. (4 Punkte) Geben Sie in Pseudocode einen Algorithmus an, der die minimale absolute Gewichtsdifferenz ausgibt. (Achtung: Es ist **nicht** nach einer optimalen Verteilung der Gegenstände selbst gefragt!)

5. (1 Punkt) Geben Sie eine kleinste obere Schranke (in O-Notation) für den asymptotischen Aufwand des Algorithmus aus Teilaufgabe 4 an. (Mit kurzer Begründung)

Vorname	Name	Matr.Nr.

6 (18)

Aufgabe 3: Bäume

Gegeben sei ein binärer Suchbaum mit ganzzahligen Schlüsseln. Reichern Sie die Datenstruktur so an, dass Sie den Mittelwert der Schlüssel in einem beliebigen Unterbaum in konstanter Zeit berechnen können.

1. (4 Punkte) Welche zusätzlichen Informationen müssen an den Knoten des Suchbaums gehalten werden und was muss beim Einfügen und Löschen von Knoten beachtet werden, damit die Information immer korrekt ist? Beschreiben Sie die Lösch- und Einfügeoperation informell.

Vorname	Name	Matr.Nr.

7
(18)

2. (3 Punkte) Jetzt werden spezielle Rot-Schwarz-Bäume betrachtet. Fügen Sie die folgenden Schlüssel in einen anfänglich leeren Rot-Schwarz-Baum ein. Auch in diesem Baum soll es möglich sein, den Mittelwert der Schlüssel in einem beliebigen Unterbaum in konstanter Zeit zu berechnen. Beachten Sie, dass bei unter Umständen auftretenden Rotationen, die entsprechenden Werte aktualisiert werden.

$$1, 12, 7, 27, 2, 5$$

Geben Sie den resultierenden Rot-Schwarz-Baum mit den zusätzlichen Informationen an. Markieren Sie dabei schwarze Knoten durch einen Kreis ◯ und rote Knoten durch ein Rechteck □.

Vorname	Name	Matr.Nr.

8
(18)

3. (4 Punkte) Beschreiben Sie die Einfüge- und Löschoperationen aus Teilaufgabe 1 jetzt für den Fall von angereicherten Rot-Schwarz-Bäumen.

Vorname	Name	Matr.Nr.

9
(18)

Aufgabe 4: Hashing

1. Erstellen Sie durch sukzessives Einfügen von Datensätzen mit den Schlüsselwerten

$$11, 23, 38, 49, 55, 25, 58, 64$$

eine Hashtabelle der Länge $m = 13$. Verwenden Sie dazu die Hashfunktion

$$h(x) = x \mod 13$$

und

(a) (1 Punkt) geschlossenes Hashing mit linearem Sondieren

0	1	2	3	4	5	6	7	8	9	10	11	12

(b) (1 Punkt) geschlossenes Hashing mit quadratischem Sondieren

0	1	2	3	4	5	6	7	8	9	10	11	12

Vorname	Name	Matr.Nr.

10 (18)

2. (3 Punkte) Gegeben sei eine Hashtabelle der Größe m. Nehmen Sie an, dass der Wertebereich W der Schlüssel mehr als $k \cdot m$ Elemente hat, $k \in \mathbb{N}^+$. Beweisen Sie: Es gibt für jede Hashfunktion h eine Teilmenge von W mit k Elementen, die alle auf den gleichen Tabelleneintrag gehasht werden.

3. (1 Punkt) Betrachten Sie die folgende Hashfunktion. Eignet sie sich für eine Hashtabelle in Zusammenhang mit der angegebenen Kollisionsfunktion? (Mit kurzer Begründung)

$$\begin{aligned} h_0(s) &= s \mod 41 \\ h_i(s) &= s \mod 41 \end{aligned}$$

Vorname	Name	Matr.Nr.

Aufgabe 5: Analyse von Algorithmen

1. Betrachten Sie den folgenden Algorithmus:

```
public static void unknown(int [] a) {
   int l, r, m, x;
   for (int i = 1; i < a.length; i++) {
      x = a[i];
      l = 0;
      r = i - 1;
      while (l <= r) {
        m = (l + r) / 2;
         if (x < a[m]) { r = m - 1; }
         else { l = m + 1; }
      }
      for (int j = i - 1; j >= l; j--) {
        a[j + 1] = a[j];
      }
      a[l] = x;
   }
 }
```

(a) (2 Punkte) Was leistet der obige Algorithmus?

(b) (2 Punkte) Wenden Sie den obigen Algorithmus auf das folgende Feld „per Hand“ an. Geben Sie den Inhalt des Feldes nach jeder äußeren `for`-Schleife an.

i=0	24	53	61	28	29	38	57	18	98	66
i=1										
i=2										
i=3										
i=4										
i=5										
i=6										
i=7										
i=8										
i=9										

Vorname	Name	Matr.Nr.

(c) (1 Punkt) Geben Sie eine kleinste obere Schranke (in O-Notation) für die asymptotische Laufzeit des obigen Algorithmus an. Für elementare Operationen (arithmetische Operationen, Vergleiche, Zuweisungen) soll hierbei ein konstanter Aufwand angenommen werden. (Mit kurzer Bergründung)

2. (5 Punkte) Betrachten Sie das folgende Problem:

Gegeben seien zwei Folgen $x_1, \ldots, x_n \in \mathbb{R}$ und $y_1, \ldots, y_n \in \mathbb{R}$. Jede Folge bestehe aus jeweils paarweise verschiedenen Zahlen.

Gesucht sind alle Indizes $i, j \in \{1, \ldots, n\}$ für die $x_i = y_j$ gilt.

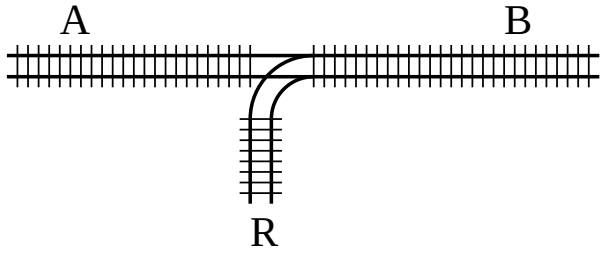
Entwerfen Sie einen $O(n \cdot \log n)$-Algorithmus in Pseudo-Code, der das oben angegebene Problem löst. Begründen Sie den Zeitaufwand Ihres Algorithmus.

Vorname	Name	Matr.Nr.

13 (18)

Aufgabe 6: Graphen

1. Die Gleise eines Rangierbahnhofs seien wie unten angegeben verlegt.



Auf Gleis A und B können beliebig viele Waggons stehen, auf dem Rangiergleis R maximal einer. Waggons können zwischen A und B, zwischen R und B, aber nicht direkt zwischen R und A verschoben werden. Weiterhin kann nur ein Waggon auf einmal verschoben werden. Es gebe zwei Sorten von Waggons: blaue (B) und grüne (G).

Zu Anfang stehe ein blauer Waggon auf Gleis A und zwei grüne Waggons auf Gleis B. Aufgabe ist es, den blauen Waggon nach Gleis B und die grünen Waggons nach Gleis A zu manövrieren.

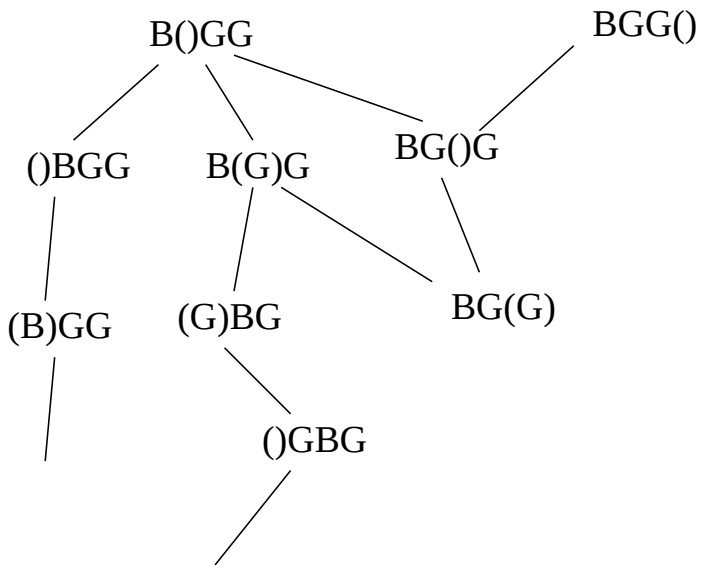
(a) (4 Punkte) Vervollständigen Sie den Graph auf der folgenden Seite.
Beachten Sie:

- Die Knoten stehen für die möglichen Waggon-Konstellationen. Bezeichnen Sie Knoten z.B. mit $G(B)G$ für „ein grüner Waggon steht auf Gleis A, der blaue Waggon auf dem Rangiergleis R und ein grüner Waggon auf Gleis B.“
- Zwei Knoten u, v sollen durch eine Kante verbunden sein, falls man durch Rangieren eines einzelnen Waggons von u nach v kommt.
- Kennzeichnen Sie Start- und Zielknoten.

Lösungshinweis: Der Graph hat insgesamt 21 Knoten.

(b) (2 Punkte) Nummerieren Sie die Knoten des Graphen in der Reihenfolge, in der sie bei einer Breitensuche mit Startknoten $B()GG$ durchlaufen würden (das ist nicht eindeutig!) und geben Sie zu jedem Knoten dessen Abstand zum Startknoten an. Welches ist der kürzeste Pfad vom Start- zum Zielknoten?

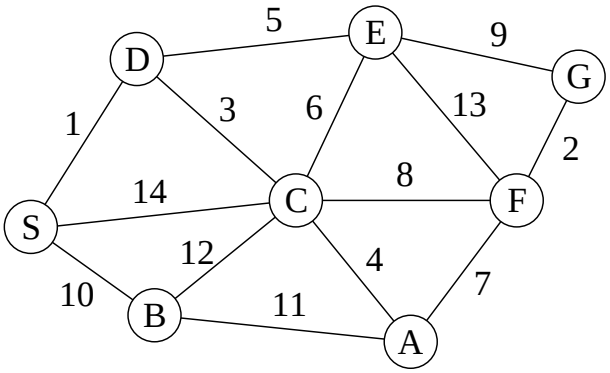
Vorname	Name	Matr.Nr.



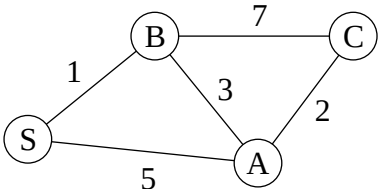
Vorname	Name	Matr.Nr.

15 (18)

2. Gegeben sei folgender gewichteter Graph G:



(a) (3 Punkte) Wenden Sie Dijkstras Algorithmus auf G mit Startknoten S an. Geben Sie die Werte aller oberen Schranken nach jedem Durchlauf der Hauptschleife von Dijkstras Algorithmus in Form einer Tabelle an.
Beispiel: Zum Graph



mit Startknoten S gehört die Tabelle

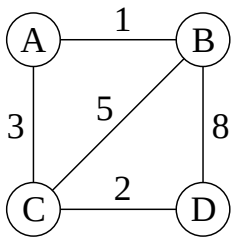
	0	1	2	3
S	0	0	0	0
A	∞	5	4	4
B	∞	1	1	1
C	∞	∞	8	6

Tragen Sie hier ihre Werte ein:

	0	1	2	3	4	5	6	7
S								
A								
B								
C								
D								
E								
F								
G								

Vorname	Name	Matr.Nr.

(b) (4 Punkte) Betrachten Sie die Menge aller kürzesten Pfade zwischen allen Knoten des folgenden Graphen G:



Ein längster dieser kürzesten Pfade heiße *kritisch*. Berechnen Sie die Länge eines kritischen Pfades in obigem Graphen, indem Sie zunächst mit Hilfe des Floyd-Warshall-Algorithmus die Längen aller kürzesten Pfade in G berechnen. Geben Sie hierzu die Werte der Distanz-Matrix nach jedem Durchlauf der Hauptschleife an. Lesen Sie aus dem Ergebnis die Länge eines kritischen Pfades ab. Wie lang ist ein kritischer Pfad?

0 :

	A	B	C	D
A				
B				
C				
D				

1 :

	A	B	C	D
A				
B				
C				
D				

2 :

	A	B	C	D
A				
B				
C				
D				

3 :

	A	B	C	D
A				
B				
C				
D				

4 :

	A	B	C	D
A				
B				
C				
D				

5 :

	A	B	C	D
A				
B				
C				
D				

Länge eines kritischen Pfades: _____

Vorname	Name	Matr.Nr.

Aufgabe 7: Multiple-Choice

Bei einigen der folgenden Fragen sind mehrere der vorgegebenen Antworten richtig. Kreuzen Sie alle richtigen Antworten an. Pro Frage erhalten Sie für eine vollständig richtige Antwort 1 Punkt. Beantworten Sie ein Frage nicht, so erhalten Sie 0 Punkte. Beantworten Sie eine Frage falsch, so erhalten Sie $-0,5$ Punkte. Insgesamt können Sie in dieser Aufgabe jedoch keine negative Punktzahl erhalten.

1. Gilt $O(\log_2 n) = O(\log_5 n)$?

   ◯ Ja ◯ Nein

2. Gegeben sei folgende Rekursionsgleichung:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= 4 \cdot T(\tfrac{n}{2}) + 2 \cdot n \quad \text{für } n > 1 \end{aligned}$$

   Wie lässt sich T dann klassifizieren?

   ◯ $T = O(n \cdot \log n)$
   ◯ $T = O(n^2)$
   ◯ $T = O(n^3)$
   ◯ Keine der obigen Klassifizierungen trifft zu.

3. Gegeben sei folgende Rekursionsgleichung:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= T(\tfrac{n}{2}) + 2 \cdot n \quad \text{für } n > 1 \end{aligned}$$

   Wie lässt sich T dann klassifizieren?

   ◯ $T = O(\log n)$
   ◯ $T = O(n)$
   ◯ $T = O(n^2)$
   ◯ Keine der obigen Klassifizierungen trifft zu.

4. InsertionSort benötigt mindestens $n - 1$ Vergleiche, um ein Array mit n Elementen zu sortieren.

   ◯ Ja ◯ Nein

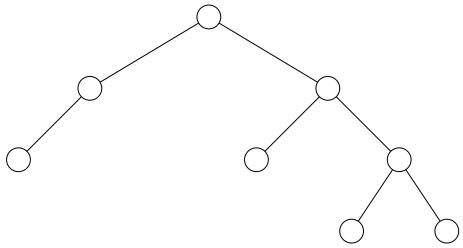
5. Bubblesort benötigt im günstigsten Fall nur $n-1$ Vergleiche, um ein Array mit n Elementen zu sortieren.

   ◯ Ja ◯ Nein

Vorname	Name	Matr.Nr.

18
(18)

6. Ist der folgende Baum höhenbalanciert?



◯ Ja ◯ Nein

7. B-Bäume heißen B-Bäume, weil sie spezielle Binärbäume sind.

◯ Ja ◯ Nein

8. Ein Knoten eines (2,4)-Baums kann u.a. die Knotengrade 1 oder 2 haben.

◯ Ja ◯ Nein

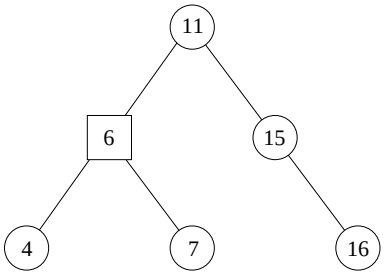
9. Die Blätter eines B-Baums unterscheiden sich nicht in ihrer Höhe.

◯ Ja ◯ Nein

10. *Memoization* ist ein anderes Wort für *Dynamisches Programmieren.*

◯ Ja ◯ Nein

11. Der folgende Baum ist ein Rot-Schwarz-Baum (Kreis = schwarzer Knoten, Quadrat = roter Knoten):



◯ Ja ◯ Nein

12. Der kleinste ungerichtete Graph, in dem alle Knoten den Grad 3 haben, hat 5 Knoten.

◯ Ja ◯ Nein

Vorname	Name	Matr.Nr.

19
(18)

13. In zusammenhängenden Graphen $G = (V, E)$ gilt $|V| = O(|E|)$.

◯ Ja ◯ Nein

14. Der Graph mit der Adjazenzmatrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

ist planar.

◯ Ja ◯ Nein

15. Zu n Knoten gibt es $O(n^2)$ mögliche ungerichtete Graphen.

◯ Ja ◯ Nein

MUSTERLÖSUNG

Probeklausur

Informatik I - Datenstrukturen und Algorithmen

8. 8. 2002, Prof. Dr. L. Kobbelt

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Studiengang: ◯ Informatik Diplom ◯ Informatik Lehramt

◯ Sonstige _____

- Schreiben Sie auf jedes Blatt **Vorname**, **Name** und **Matrikelnummer**.
- Geben Sie Ihre Antworten in lesbarer und verständlicher Form an.
- Bitte beantworten Sie die Aufgaben auf den **Aufgabenblättern** (benutzen Sie auch die Rückseiten). Antworten auf anderen Blättern können nur berücksichtigt werden, wenn **Name, Matrikelnummer und Aufgabennummer** deutlich darauf erkennbar sind.
- Was nicht bewertet werden soll, kennzeichnen Sie bitte durch **Durchstreichen**.
- Werden Täuschungsversuche beobachtet, so wird die Klausur mit **nicht bestanden** bewertet.
- Geben Sie am Ende der Klausur **alle Blätter zusammen mit den Aufgabenblättern ab**.
- Verwenden Sie ausschließlich dokumentenechte Stifte mit schwarzer oder blauer Farbe (d.h. insbesondere sind keine Bleistifte und keine Stifte mit roter Farbe zugelassen).

	Anzahl Punkte	**Erreichte Punkte**
Aufgabe 1	**12**	
Aufgabe 2	**11**	
Aufgabe 3	**11**	
Aufgabe 4	**6**	
Aufgabe 5	**10**	
Aufgabe 6	**13**	
Aufgabe 7	**15**	
Summe	**78**	
Note	**-**	

Vorname	Name	Matr.Nr.

1 (19)

Aufgabe 1: Laufzeitanalyse

1. Betrachten Sie folgende Java-Methode:

```
public static int myst1(int b, int n) {
    int m = n;
    int p = b;
    int res = 1;
    while (m > 0) {
       if (m % 2 == 0) {
            p = p * p;
            m = m / 2;
        }
        else {
            res = res * p;
            m = m - 1;
        }
     }
     return res;
}
```

(a) (1 Punkt) Was berechnet der Aufruf `myst1(b, n)` für natürliche Zahlen b und n? (Ohne Begründung)

Lösung:
b^n

(b) (1 Punkt) Bestimmen Sie eine kleinste obere Schranke (in O-Notation) für die asymptotische Laufzeit von `myst1` für den Fall natürlicher Argumente `b` und `n`. Für elementare Operationen (arithmetische Operationen, Vergleiche, Zuweisungen) soll hierbei ein konstanter Aufwand angenommen werden. (Mit kurzer Begründung)

Lösung:
$O(\log n)$
Betrachte zunächst den Fall, dass n eine Zweierpotenz ist. Dann wird `m` pro Schleifendurchlauf (bis auf den letzten) halbiert. Somit liegen insgesamt $(\log_2 n) + 1$ Schleifendurchläufe vor. Die oben angegebene Schranke ergibt sich nun aus dieser Überlegung und aus der Tatsache, dass `m` in mindestens jedem zweiten Schleifendurchlauf halbiert wird.

Vorname	Name	Matr.Nr.

2 (19)

2. Betrachten Sie folgende Java-Methode:

```
public static int myst2(int a, int b) {
   if (a == b) return a;
   else {
       if (a < b) {
          if (b - a == 1) return a;
          else return myst2(a + 1, b - 1);
       }
       else {
          if (a - b == 1) return b;
          else return myst2(a - 1, b + 1);
       }
   }
}
```

(a) (1 Punkt) Was berechnet der Aufruf `myst2(a, b)`? (Ohne Begründung)

Lösung:
$\lfloor \frac{a+b}{2} \rfloor$

(b) (2 Punkte) Stellen Sie eine Rekursionsgleichung auf, die die Laufzeit von `myst2` beschreibt. Für elementare Operationen (arithmetische Operationen und Vergleiche) soll hierbei ein konstanter Aufwand angenommen werden. (Ohne Begründung)

Lösung:
Sei c der Aufwand für eine elementare Operation und $n = |a - b|$.

$$T(n) = \begin{cases} c & \text{falls} \quad n = 0 \\ 4c & \text{falls} \quad n = 1 \\ T(n-2) + 6c & \text{sonst} \end{cases}$$

(c) (1 Punkt) Bestimmen Sie eine kleinste obere Schranke (in O-Notation) für die asymptotische Laufzeit von `myst2`. Für elementare Operationen (arithmetische Operationen und Vergleiche) soll hierbei ein konstanter Aufwand angenommen werden. (Mit kurzer Begründung)

Lösung:
$O(|a - b|)$
Ist $n = |a - b|$ gerade, so gilt nach Teilaufgabe b:

$$\begin{aligned} T(n) &= T(n-2) + 6c \\ &= T(n-4) + 2 \cdot 6c \\ &= \cdots \\ &= T(0) + \frac{n}{2} \cdot 6c \\ &= c + 3cn = O(n) \end{aligned}$$

Analog für ungerade n.

Vorname	Name	Matr.Nr.

3 (19)

3. Seien $f, g, h : \mathbb{N} \to \mathbb{R}^+$ streng monoton wachsende Funktionen. Beweisen oder widerlegen Sie:

   (a) (2 Punkte) $f(n) = \Omega(g(n)) \implies 2^{f(n)} = \Omega(2^{g(n)})$

   Lösung:
   Gegenbeispiel: $g(n) = 2n$, $f(n) = n$
   Es gilt offensichtlich $f(n) = \Omega(g(n))$, aber es ist $2^n \notin \Omega(2^{2n})$.
   Beweis durch Widerspruch:
   $2^n \in \Omega(2^{2n})$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : 2^n \geq c \cdot 2^{2n}$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : 2^n \geq c \cdot (2^n)^2$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : \frac{1}{c} \geq 2^n$
   Widerspruch zu $2^n \to \infty$ für $n \to \infty$.

   (b) (2 Punkte) $f(n) = \theta\left(\sqrt{f(n^2)}\right)$

   Lösung:
   Gegenbeispiel: $f(n) = \log_2 n$
   Es gilt
   $$\sqrt{f(n^2)} = \sqrt{\log_2(n^2)} = \sqrt{2} \cdot \sqrt{\log_2 n}$$
   und damit
   $f(n) = \theta\left(\sqrt{f(n^2)}\right)$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : \log_2 n \leq c\sqrt{2}\sqrt{\log_2 n}$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : \sqrt{\log_2 n} \leq c \cdot \sqrt{2}$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : \log_2 n \leq 2c^2$
   Widerspruch zu $\log_2 n \to \infty$ für $n \to \infty$.

   (c) (2 Punkte) $f(n) = O(g(n)) \implies f(n)^5 = O\left(g(n)^5\right)$

   Lösung:
   Beweis:
   $f(n) = O(g(n))$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
   $\Rightarrow \exists c > 0\, \exists n_0 > 0\, \forall n \geq n_0 : f(n)^5 \leq (cg(n))^5 = c^5 g(n)^5$
   $\Rightarrow f(n)^5 = O\left(g(n)^5\right)$

Vorname	Name	Matr.Nr.

Aufgabe 2: Entwurfsparadigmen

Der Sattel eines Packesels hat zwei Taschen, eine links und eine rechts des Esels. Der Esel soll nun möglichst gleichmäßig mit $N \geq 1$ Gegenständen beladen werden, das heißt, der Gewichtsunterschied zwischen den beiden Taschen soll minimal werden. Wir nehmen an, dass der i-te Gegenstand ($1 \leq i \leq N$) genau $g_i \in \mathbb{N}^+$ Kilogramm wiegt.
Zur Lösung des Problems soll ein Algorithmus entwickelt werden, der auf Dynamischem Programmieren beruht. Es sei $G = \sum_{i=1}^{N} g_i$ das Gesamtgewicht aller Gegenstände. Weiter sei χ_{ij} eine binäre Variable mit $\chi_{ij} = 1$, falls sich die ersten i Gegenstände so auf die Taschen verteilen lassen, dass die Gewichtsdifferenz zwischen linker und rechter Tasche $= j$ ist, und $\chi_{ij} = 0$ sonst ($1 \leq i \leq N, -G \leq j \leq G$). Beispiel: Ist $\chi_{ij} = 1$ und wird der $(i+1)$-te Gegenstand in die linke Tasche gelegt, so folgt $\chi_{i+1,j+g_{i+1}} = 1$.

1. (1 Punkt) Geben Sie ein Beispiel für N und $g_1, \ldots, g_N$ an, bei dem sich die Gegenstände nicht so auf die Taschen verteilen lassen, dass beide Taschen gleich schwer sind. (Ohne Begründung)

   Lösung:
   $N = 1, g_1 = 1$

2. (2 Punkte) Geben Sie die Werte für χ_{1j} an. (Ohne Begründung)

   Lösung:

$$\chi_{1j} = \begin{cases} 1 & \text{falls} \quad j = \pm g_1 \\ 0 & \text{sonst} \end{cases}$$

3. (3 Punkte) Geben Sie eine Rekursionsformel für χ_{ij} an. (Ohne Begründung)

   Lösung:

$$\chi_{ij} = \begin{cases} 1 & \text{falls} \quad -G \leq j - g_i \leq G \quad \text{und} \quad \chi_{i-1,j-g_i} = 1 \\ 1 & \text{falls} \quad -G \leq j + g_i \leq G \quad \text{und} \quad \chi_{i-1,j+g_i} = 1 \\ 0 & \text{sonst} \end{cases}$$

Vorname	Name	Matr.Nr.

4. (4 Punkte) Geben Sie in Pseudocode einen Algorithmus an, der die minimale absolute Gewichtsdifferenz ausgibt. (Achtung: Es ist **nicht** nach einer optimalen Verteilung der Gegenstände selbst gefragt!)

   Lösung:
   Die hier angegebene Lösung ist lauffähiger Java-Code. In Pseudocode müsste z.B. kein Speicher alloziiert werden, die Array-Grenzen müssten nicht überprüft werden, usw.

```
public static void best(int N, int [] g)
{
  // Summe der Gewichte aller Gegenstaende
  int G = 0;
  for (int i = 1; i <= N; ++i)
    G += g[i];

  // Speicherplatz fuer die Tabelle bereitstellen
  int chi[][] = new int[N+1][2*G+1];

  // Basisfall der Rekursion, erste Zeile der Tabelle
  for (int j = -G; j <= G; ++j)
    if (j == g[1] || j == -g[1])
      chi[1][j+G] = 1;
    else
      chi[1][j+G] = 0;

  // Auffuellen der Tabelle
  for (int i = 2; i <= N; ++i)
    for (int j = -G; j <= G; ++j)
    {
      chi[i][j+G] = 0;
      if (j+g[i] <=  G && chi[i-1][j+g[i]+G] == 1) chi[i][j+G] = 1;
      if (j-g[i] >= -G && chi[i-1][j-g[i]+G] == 1) chi[i][j+G] = 1;
    }

  // Ausgabe des Ergebnisses
  int b = 0;
  for ( ; chi[N][b+G] == 0; ++b)
    ;
  System.out.println("Bestmoegliche Gewichtsdifferenz ist +-" + b + " kg");
}
```

5. (1 Punkt) Geben Sie eine kleinste obere Schranke (in O-Notation) für den asymptotischen Aufwand des Algorithmus aus Teilaufgabe 4 an. (Mit kurzer Begründung)

   Lösung:
   Jeder Eintrag der Tabelle wird einmal mit Aufwand $O(1)$ berechnet. Der Aufwand des Algorithmus ist daher proportional zur Tabellengröße, also $O(GN)$.

Vorname	Name	Matr.Nr.

Aufgabe 3: Bäume

Gegeben sei ein binärer Suchbaum mit ganzzahligen Schlüsseln. Reichern Sie die Datenstruktur so an, dass Sie den Mittelwert der Schlüssel in einem beliebigen Unterbaum in konstanter Zeit berechnen können.

1. (4 Punkte) Welche zusätzlichen Informationen müssen an den Knoten des Suchbaums gehalten werden und was muss beim Einfügen und Löschen von Knoten beachtet werden, damit die Information immer korrekt ist? Beschreiben Sie die Lösch- und Einfügeoperation informell.

   Lösung:
   In jedem Knoten K muss die Summe g der Schlüssel und die Zahl n der Knoten des Unterbaums von K abgelegt werden. Der Mittelwert aller Schlüssel des Unterbaums von K ergibt sich dann einfach als Quotient $K.g/K.n$.

   Beim Einfügen eines Schlüssels s muss beachtet werden, dass schon beim Suchen der Einfügestelle in allen Knoten die durchlaufen werden, die Werte g bzw. n durch $g+s$ bzw. $n+1$ ersetzt werden.

   Beim Löschen eines Schlüssels s muss zunächst festgestellt werden, ob s überhaupt im Baum vorkommt. Wenn nein, ist man bereits fertig. Wenn ja, so sei K der Knoten der s enthält. Wie bei binären Suchbäumen üblich unterscheiden wir 3 Fälle:

   (a) Ist K ein Blatt, so werden für alle Knoten auf dem Weg von K zur Wurzel die Werte g bzw. n durch $g-s$ bzw. $n-1$ ersetzt und K aus dem Baum entfernt.

   (b) Hat K genau einen Sohn L, werden für alle Knoten auf dem Weg von K zur Wurzel die Werte g bzw. n durch $g-s$ bzw. $n-1$ ersetzt und K durch L ersetzt.

   (c) Hat K zwei Söhne, so sei N der Knoten der den nächstgrößeren Schlüssel $t > s$ enthält. N hat keinen linken Sohn, wir löschen daher N wie in Fall b. Nun ersetzt man s durch t und ersetzt die Werte g auf dem Weg von K zur Wurzel durch $g-s+t$.

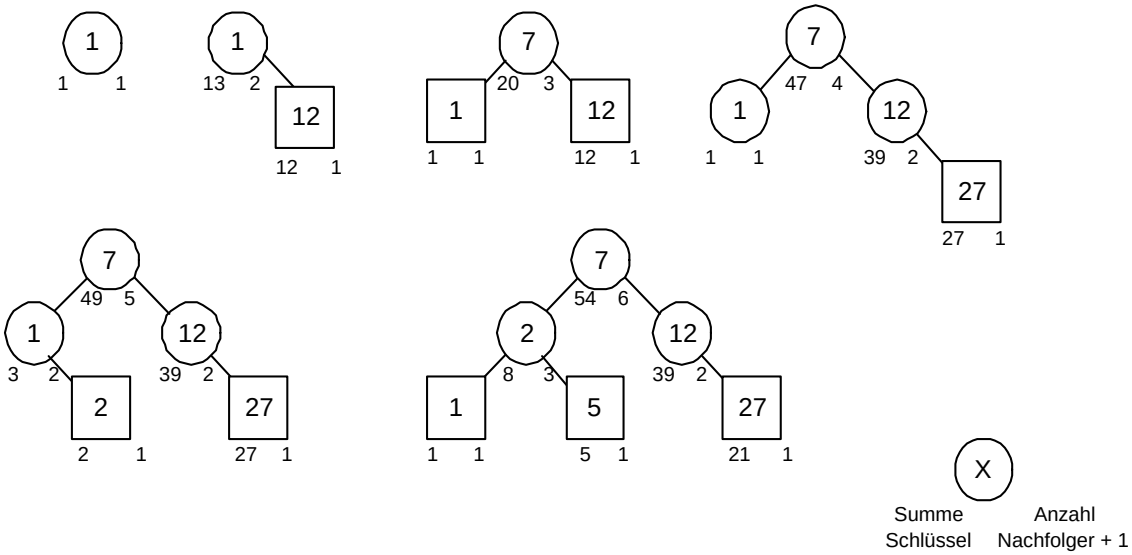
Vorname	Name	Matr.Nr.

2. (3 Punkte) Jetzt werden spezielle Rot-Schwarz-Bäume betrachtet. Fügen Sie die folgenden Schlüssel in einen anfänglich leeren Rot-Schwarz-Baum ein. Auch in diesem Baum soll es möglich sein, den Mittelwert der Schlüssel in einem beliebigen Unterbaum in konstanter Zeit zu berechnen. Beachten Sie, dass bei unter Umständen auftretenden Rotationen, die entsprechenden Werte aktualisiert werden.

$$1, 12, 7, 27, 2, 5$$

Geben Sie den resultierenden Rot-Schwarz-Baum mit den zusätzlichen Informationen an. Markieren Sie dabei schwarze Knoten durch einen Kreis ◯ und rote Knoten durch ein Rechteck □.

Lösung:

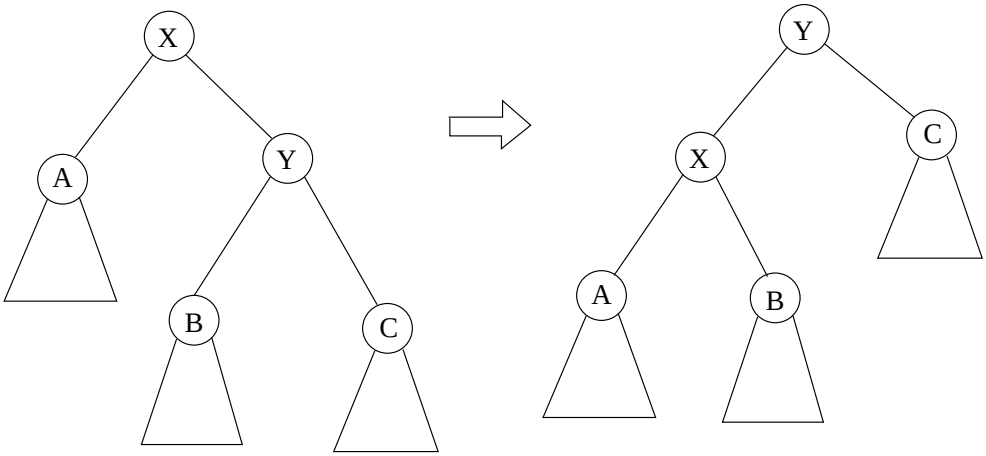


Vorname	Name	Matr.Nr.

3. (4 Punkte) Beschreiben Sie die Einfüge- und Löschoperationen aus Teilaufgabe 1 jetzt für den Fall von angereicherten Rot-Schwarz-Bäumen.

Lösung:
Zunächst werden die Einfüge- und Löschoperationen wie für einen gewöhnlichen Binärbaum behandelt, siehe Teilaufgabe 1. Danach kann es bei Rot-Schwarz-Bäumen zu Umfärbungen oder Rotationen kommen. Umfärbungen ändern nichts an den Werten g bzw. n. Kommt es zu einer Rotation, wie z.B. der unten dargestellten Links-Rotation, so müssen die Werte entsprechend angepasst werden:

$$\begin{aligned} X.g &\leftarrow X.g - C.g - Y.s \\ X.n &\leftarrow X.n - C.n - 1 \\ Y.g &\leftarrow Y.g + A.g + X.s \\ Y.n &\leftarrow Y.n + A.n + 1 \end{aligned}$$



Vorname	Name	Matr.Nr.

Aufgabe 4: Hashing

1. Erstellen Sie durch sukzessives Einfügen von Datensätzen mit den Schlüsselwerten

$$11, 23, 38, 49, 55, 25, 58, 64$$

eine Hashtabelle der Länge $m = 13$. Verwenden Sie dazu die Hashfunktion

$$h(x) = x \mod 13$$

und

(a) (1 Punkt) geschlossenes Hashing mit linearem Sondieren

Lösung:

0	1	2	3	4	5	6	7	8	9	10	11	12
49	25	64	55			58				23	11	38

(b) (1 Punkt) geschlossenes Hashing mit quadratischem Sondieren

Lösung:

0	1	2	3	4	5	6	7	8	9	10	11	12
25	49		55			58		64		23	11	38

Vorname	Name	Matr.Nr.

2. (3 Punkte) Gegeben sei eine Hashtabelle der Größe m. Nehmen Sie an, dass der Wertebereich W der Schlüssel mehr als $k \cdot m$ Elemente hat, $k \in \mathbb{N}^+$. Beweisen Sie: Es gibt für jede Hashfunktion h eine Teilmenge von W mit k Elementen, die alle auf den gleichen Tabelleneintrag gehasht werden.

   Lösung:
   Beweis durch Widerspruch. Annahme: Es gibt eine Hashfunktion h, so dass

   $$|\{w \in W : h(w) = x\}| < k$$

   für alle $x \in \{0, \ldots, m-1\}$. Dann folgt

   $$\begin{aligned} |W| &= \sum_{x=0}^{m-1} |\{w \in W : h(w) = x\}| \\ &< \sum_{x=0}^{m-1} k \\ &= mk \end{aligned}$$

   im Widerspruch zur Voraussetzung $|W| \geq km$.

3. (1 Punkt) Betrachten Sie die folgende Hashfunktion. Eignet sie sich für eine Hashtabelle in Zusammenhang mit der angegebenen Kollisionsfunktion? (Mit kurzer Begründung)

   $$\begin{aligned} h_0(s) &= s \mod 41 \\ h_i(s) &= s \mod 41 \end{aligned}$$

   Lösung:
   Sie ist ist nicht geeignet, da für alle $x \in \mathbb{N}$ und alle $j, k \in \mathbb{N}$ gilt: $h_j(x) = h_k(x)$. Somit wird durch einen Sondierungsschritt kein weiterer Tabellenplatz betrachtet und eine eventuelle Kollision nicht aufgelöst.

Vorname	Name	Matr.Nr.

Aufgabe 5: Analyse von Algorithmen

1. Betrachten Sie den folgenden Algorithmus:

```
public static void unknown(int [] a) {
   int l, r, m, x;
   for (int i = 1; i < a.length; i++) {
      x = a[i];
      l = 0;
      r = i - 1;
      while (l <= r) {
        m = (l + r) / 2;
         if (x < a[m]) { r = m - 1; }
         else { l = m + 1; }
      }
      for (int j = i - 1; j >= l; j--) {
        a[j + 1] = a[j];
      }
      a[l] = x;
   }
 }
```

(a) (2 Punkte) Was leistet der obige Algorithmus?

Lösung:
Der Algorithmus ist eine Variante von Insertion-Sort. Im Unterschied zum Original-Algorithmus wird die Stelle, an der eingefügt werden soll, durch binäre Suche bestimmt.

(b) (2 Punkte) Wenden Sie den obigen Algorithmus auf das folgende Feld „per Hand“ an. Geben Sie den Inhalt des Feldes nach jeder äußeren `for`-Schleife an.

Lösung:

i=0	24	53	61	28	29	38	57	18	98	66
i=1	24	53	61	28	29	38	57	18	98	66
i=2	24	53	61	28	29	38	57	18	98	66
i=3	24	28	53	61	29	38	57	18	98	66
i=4	24	28	29	53	61	38	57	18	98	66
i=5	24	28	29	38	53	61	57	18	98	66
i=6	24	28	29	38	53	57	61	18	98	66
i=7	18	24	28	29	38	53	57	61	98	66
i=8	18	24	28	29	38	53	57	61	98	66
i=9	18	24	28	29	38	53	57	61	66	98

Vorname	Name	Matr.Nr.

(c) (1 Punkt) Geben Sie eine kleinste obere Schranke (in O-Notation) für die asymptotische Laufzeit des obigen Algorithmus an. Für elementare Operationen (arithmetische Operationen, Vergleiche, Zuweisungen) soll hierbei ein konstanter Aufwand angenommen werden. (Mit kurzer Bergründung)

Lösung:
Der Code dieses Algorithmus entspricht im Wesentlichen dem von Insertion-Sort und hätte ohne die binäre Suche einen Aufwand von $O(n^2)$. Die binäre Suche (Aufwand $O(\log n)$) verschlechtert den Aufwand nicht, da sie n-mal ausgeführt wird und so nur zu einem Gesamtaufwand von $O(n \log n)$ führt. Tatsächlich bewirkt die binäre Suche, dass im Vergleich zu Insertion Sort weniger Vergleiche durchgeführt werden müssen, die Zahl der Zuweisungen bleibt aber $O(n^2)$.

2. (5 Punkte) Betrachten Sie das folgende Problem:

Gegeben seien zwei Folgen $x_1, \ldots, x_n \in \mathbb{R}$ und $y_1, \ldots, y_n \in \mathbb{R}$. Jede Folge bestehe aus jeweils paarweise verschiedenen Zahlen.

Gesucht sind alle Indizes $i, j \in \{1, \ldots, n\}$ für die $x_i = y_j$ gilt.

Entwerfen Sie einen $O(n \cdot \log n)$-Algorithmus in Pseudo-Code, der das oben angegebene Problem löst. Begründen Sie den Zeitaufwand Ihres Algorithmus.

Lösung:
Algorithmus zur Lösung des Problems:

```
PROCEDURE GleicheZahlen {
  Sortiere (y[1]...y[n]) mit bekanntem O(n log n) Sortierverfahren,
    d.h. erstelle eine Permutation p:{1...n}->{1...n} so dass die Folge
    z[1] = y[p[1]], z[2] = y[p[2]], ... , z[n] = y[p[n]] sortiert ist
  FOR (i = 1 TO n) {
      int j = der Index f"ur den x[i] = z[j] gilt bzw. = -1 falls
              solch ein Wert nicht existiert (kann mit Binaersuche
              in O(log n) berechnet werden)
      IF (j != -1) {
          Print(i);
          Print(p[j]);
      }
  }
}
```

Begründung:

- Sortierung von y[1]...y[n] in $O(n \cdot \log n)$ möglich
- Binärsuche in $O(\log n)$ möglich
- Binärsuche wird n-mal durchgeführt

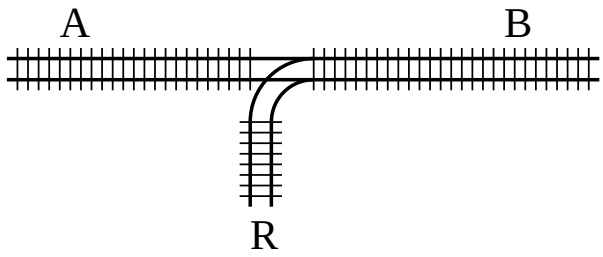
Insgesamt also: $O(n \cdot \log n)$

Vorname	Name	Matr.Nr.

13 (19)

Aufgabe 6: Graphen

1. Die Gleise eines Rangierbahnhofs seien wie unten angegeben verlegt.



Auf Gleis A und B können beliebig viele Waggons stehen, auf dem Rangiergleis R maximal einer. Waggons können zwischen A und B, zwischen R und B, aber nicht direkt zwischen R und A verschoben werden. Weiterhin kann nur ein Waggon auf einmal verschoben werden. Es gebe zwei Sorten von Waggons: blaue (B) und grüne (G).

Zu Anfang stehe ein blauer Waggon auf Gleis A und zwei grüne Waggons auf Gleis B. Aufgabe ist es, den blauen Waggon nach Gleis B und die grünen Waggons nach Gleis A zu manövrieren.

(a) (4 Punkte) Vervollständigen Sie den Graph auf der folgenden Seite.
Beachten Sie:

- Die Knoten stehen für die möglichen Waggon-Konstellationen. Bezeichnen Sie Knoten z.B. mit $G(B)G$ für „ein grüner Waggon steht auf Gleis A, der blaue Waggon auf dem Rangiergleis R und ein grüner Waggon auf Gleis B."
- Zwei Knoten u, v sollen durch eine Kante verbunden sein, falls man durch Rangieren eines einzelnen Waggons von u nach v kommt.
- Kennzeichnen Sie Start- und Zielknoten.

Lösungshinweis: Der Graph hat insgesamt 21 Knoten.

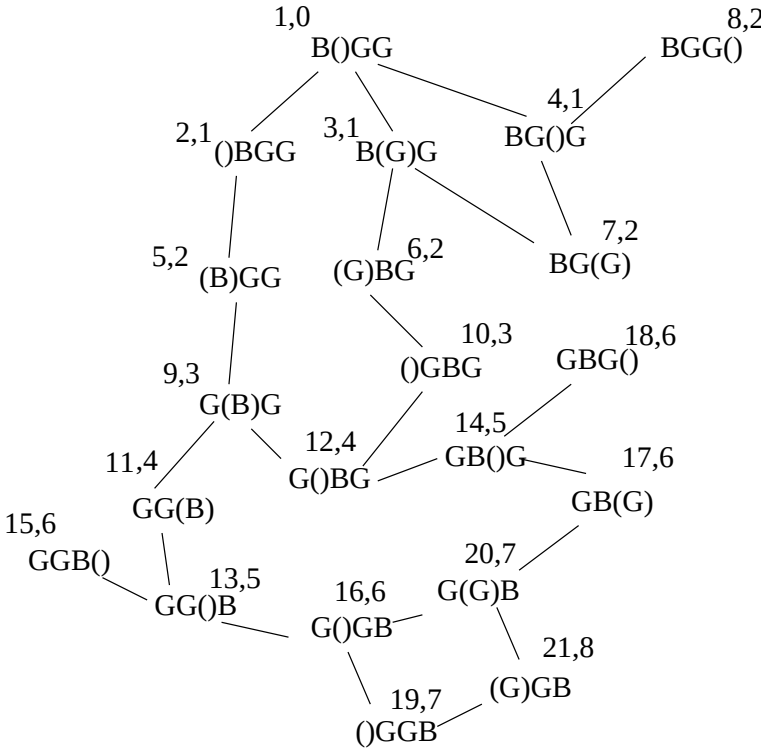
(b) (2 Punkte) Nummerieren Sie die Knoten des Graphen in der Reihenfolge, in der sie bei einer Breitensuche mit Startknoten $B()GG$ durchlaufen würden (das ist nicht eindeutig!) und geben Sie zu jedem Knoten dessen Abstand zum Startknoten an. Welches ist der kürzeste Pfad vom Start- zum Zielknoten?

Lösung:
Kürzester Pfad: $B()GG$, $()BGG$, $(B)GG$, $G(B)G$, $GG(B)$, $GG()B$

Vorname	Name	Matr.Nr.

14 (19)

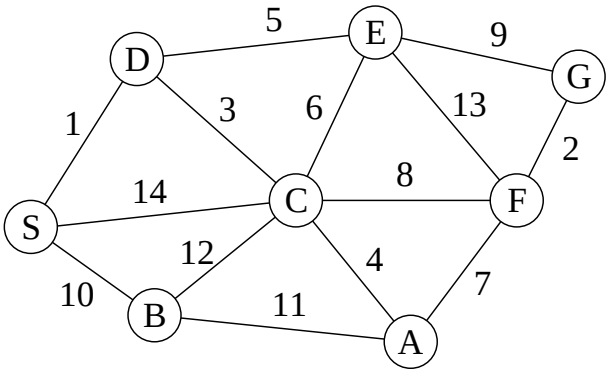
Lösung:
Nummerierung: (Index, Tiefe)



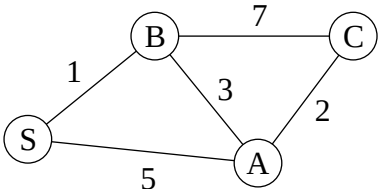
Vorname	Name	Matr.Nr.

15 (19)

2. Gegeben sei folgender gewichteter Graph G:



(a) (3 Punkte) Wenden Sie Dijkstras Algorithmus auf G mit Startknoten S an. Geben Sie die Werte aller oberen Schranken nach jedem Durchlauf der Hauptschleife von Dijkstras Algorithmus in Form einer Tabelle an.
Beispiel: Zum Graph



mit Startknoten S gehört die Tabelle

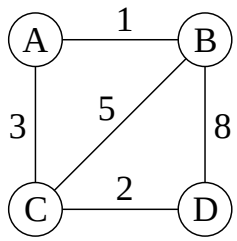
	0	1	2	3
S	0	0	0	0
A	∞	5	4	4
B	∞	1	1	1
C	∞	∞	8	6

Lösung:

	0	1	2	3	4	5	6	7
S	0	0	0	0	0	0	0	0
A	∞	∞	∞	8	8	8	8	8
B	∞	10	10	10	10	10	10	10
C	∞	14	4	4	4	4	4	4
D	∞	1	1	1	1	1	1	1
E	∞	∞	6	6	6	6	6	6
F	∞	∞	∞	12	12	12	12	12
G	∞	∞	∞	∞	15	15	15	14

Vorname	Name	Matr.Nr.

(b) (4 Punkte) Betrachten Sie die Menge aller kürzesten Pfade zwischen allen Knoten des folgenden Graphen G:



Ein längster dieser kürzesten Pfade heiße *kritisch*. Berechnen Sie die Länge eines kritischen Pfades in obigem Graphen, indem Sie zunächst mit Hilfe des Floyd-Warshall-Algorithmus die Längen aller kürzesten Pfade in G berechnen. Geben Sie hierzu die Werte der Distanz-Matrix nach jedem Durchlauf der Hauptschleife an. Lesen Sie aus dem Ergebnis die Länge eines kritischen Pfades ab. Wie lang ist ein kritischer Pfad?

Lösung:

0 :

	A	B	C	D
A	0	1	3	∞
B	1	0	5	8
C	3	5	0	2
D	∞	8	2	0

1 :

	A	B	C	D
A	0	1	3	∞
B	1	0	4	8
C	3	4	0	2
D	∞	8	2	0

2 :

	A	B	C	D
A	0	1	3	9
B	1	0	4	8
C	3	4	0	2
D	9	8	2	0

3 :

	A	B	C	D
A	0	1	3	5
B	1	0	4	6
C	3	4	0	2
D	5	6	2	0

4 :

	A	B	C	D
A	0	1	3	5
B	1	0	4	6
C	3	4	0	2
D	5	6	2	0

5 :

	A	B	C	D
A				
B				
C				
D				

Länge eines kritischen Pfades: 6

Vorname	Name	Matr.Nr.

Aufgabe 7: Multiple-Choice

Bei einigen der folgenden Fragen sind mehrere der vorgegebenen Antworten richtig. Kreuzen Sie alle richtigen Antworten an. Pro Frage erhalten Sie für eine vollständig richtige Antwort 1 Punkt. Beantworten Sie ein Frage nicht, so erhalten Sie 0 Punkte. Beantworten Sie eine Frage falsch, so erhalten Sie $-0,5$ Punkte. Insgesamt können Sie in dieser Aufgabe jedoch keine negative Punktzahl erhalten.

1. Gilt $O(\log_2 n) = O(\log_5 n)$?

   ⊗ Ja ○ Nein

2. Gegeben sei folgende Rekursionsgleichung:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= 4 \cdot T(\tfrac{n}{2}) + 2 \cdot n \quad \text{für } n > 1 \end{aligned}$$

   Wie lässt sich T dann klassifizieren?

   ○ $T = O(n \cdot \log n)$
   ⊗ $T = O(n^2)$
   ⊗ $T = O(n^3)$
   ○ Keine der obigen Klassifizierungen trifft zu.

3. Gegeben sei folgende Rekursionsgleichung:

$$\begin{aligned} T(1) &= 2 \\ T(n) &= T(\tfrac{n}{2}) + 2 \cdot n \quad \text{für } n > 1 \end{aligned}$$

   Wie lässt sich T dann klassifizieren?

   ○ $T = O(\log n)$
   ⊗ $T = O(n)$
   ⊗ $T = O(n^2)$
   ○ Keine der obigen Klassifizierungen trifft zu.

4. InsertionSort benötigt mindestens $n - 1$ Vergleiche, um ein Array mit n Elementen zu sortieren.

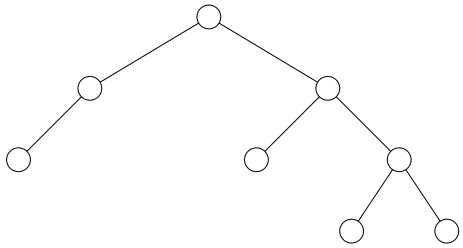
   ⊗ Ja ○ Nein

5. Bubblesort benötigt im günstigsten Fall nur $n-1$ Vergleiche, um ein Array mit n Elementen zu sortieren.

   ○ Ja ⊗ Nein

Vorname	Name	Matr.Nr.

6. Ist der folgende Baum höhenbalanciert?



⊗ Ja ○ Nein

7. B-Bäume heißen B-Bäume, weil sie spezielle Binärbäume sind.

○ Ja ⊗ Nein

8. Ein Knoten eines (2,4)-Baums kann u.a. die Knotengrade 1 oder 2 haben.

○ Ja ⊗ Nein

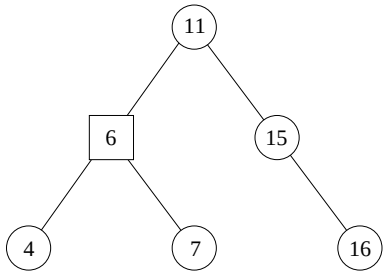
9. Die Blätter eines B-Baums unterscheiden sich nicht in ihrer Höhe.

⊗ Ja ○ Nein

10. *Memoization* ist ein anderes Wort für *Dynamisches Programmieren.*

○ Ja ⊗ Nein

11. Der folgende Baum ist ein Rot-Schwarz-Baum (Kreis = schwarzer Knoten, Quadrat = roter Knoten):



○ Ja ⊗ Nein

12. Der kleinste ungerichtete Graph, in dem alle Knoten den Grad 3 haben, hat 5 Knoten.

○ Ja ⊗ Nein

Vorname	Name	Matr.Nr.

19
(19)

13. In zusammenhängenden Graphen $G = (V, E)$ gilt $|V| = O(|E|)$.

⊗ Ja ○ Nein

14. Der Graph mit der Adjazenzmatrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

ist planar.

○ Ja ⊗ Nein

15. Zu n Knoten gibt es $O(n^2)$ mögliche ungerichtete Graphen.

○ Ja ⊗ Nein