C. Sohler
A. Krivošija, A. Rey, H. Sandvoß

DAP2 – Probeklausur

Datum: 24. Juli 2017

Matrikelnummer	
Vorname	
Nachname	

Diese Klausur besteht aus **acht** Aufgaben mit insgesamt **50** Punkten. Zum Bestehen dieser Klausur sind **25 Punkte** notwendig.

Benutzen Sie **keinen Bleistift**. Schreiben Sie unbedingt **leserlich**, da unleserliche Antworten nicht gewertet werden können.

Schreiben Sie oben auf **jede** Seite **lesbar** Ihren Namen und Ihre Matrikelnummer.

Als Hilfsmittel ist ein handbeschriebener DIN-A4-Zettel zugelassen. Legen Sie Ihren Rucksack unten im Hörsaal ab, am Platz sind **nur** der handbeschriebene Zettel, Schreibmaterial, Ihr Lichtbildausweis und Studentenausweis und Verpflegung zulässig. Bei Täuschungsversuchen wird Ihre Klausur als mangelhaft (5.0) bewertet.

Wenn der Platz unter der Aufgabenstellung nicht ausreicht, schreiben Sie zuerst auf die Rückseite und anschließend auf weitere Blätter der gleichen Aufgabe. Sollte der Platz nicht ausreichen, können Sie von der Aufsicht Zusatzblätter erhalten.

Werden zu einer Aufgabe zwei Lösungen abgegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für eine Lösung.

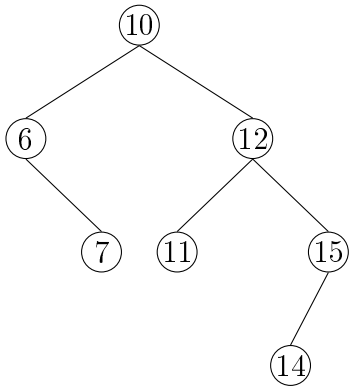
Ergebnisse, Algorithmen und Datenstrukturen aus der Vorlesung dürfen zitiert werden. Ergebnisse aus den Übungen dürfen nicht verwendet werden, sondern müssen selbst hergeleitet werden.

Die Bearbeitungszeit beträgt 180 Minuten (diese Klausur richtet sich an Studierende nach der Bachelorordnung Informatik).

Mit dem Beginn des Ausfüllens der Klausur gilt die Prüfungsfähigkeit als bestätigt.

Aufgabe 1 (4 Punkte): (AVL-Bäume)

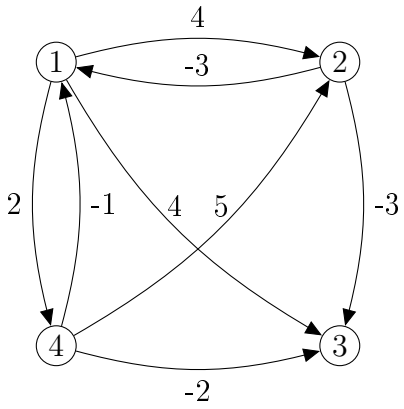
Betrachten Sie den folgenden AVL-Baum:



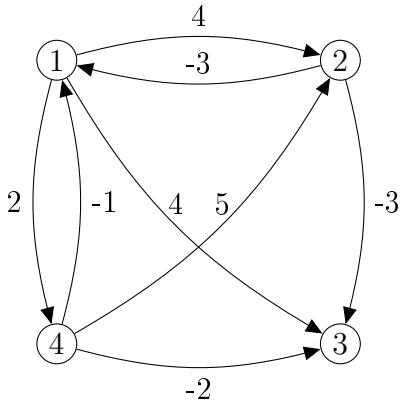
Fügen Sie in diesen Baum nacheinander Knoten mit den Schlüsseln $1, 9, 2, 4, 3$ ein. Geben Sie den AVL-Baum nahc jeder Einfügeoperation an. Geben Sie außerdem für jede Rotation an, an welchem Knoten rotiert wird und in welche Richtung. Löschen Sie aus dem entstandenen Baum anschließend erst den Schlüssel 11 und dann den Schlüssel 10.

Aufgabe 2 (4 Punkte): (Floyd-Warshall)

Gegeben sei der folgende gewichtete und gerichtete Graph (G, w):



Die Matrizen $D^{(0)}$, $D^{(1)}$, $D^{(2)}$, $D^{(3)}$ und $D^{(4)}$ enthalten die kürzesten Distanzen zwischen den Knoten des Graphen G nach Initialisierung und nach jeder von vier Iterationen des Algorithmus von Floyd und Warshall. Tragen Sie den Inhalt der Matrizen $D^{(i)}$, für $i = \{1, 2, 3, 4\}$ in die auf der nächsten Seite bereitgestellte Vorlage ein.



Nach Initialisierung:

$$D^{(0)} = \begin{pmatrix} 0 & 4 & 4 & 2 \\ -3 & 0 & -3 & \infty \\ \infty & \infty & 0 & \infty \\ -1 & 5 & -2 & 0 \end{pmatrix}$$

Nach Iteration 1:

$$D^{(1)} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Nach Iteration 2:

$$D^{(2)} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Nach Iteration 3:

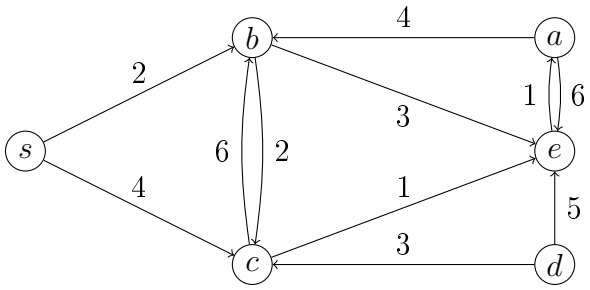
$$D^{(3)} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Nach Iteration 4:

$$D^{(4)} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

Aufgabe 3 (4 Punkte): (Dijkstra Algorithmus)

Führen Sie den Dijkstra-Algorithmus für den folgenden Graphen durch. Geben Sie dabei nach jedem Durchlauf der **while**-Schleife die Knoten in der Prioritätenschlange Q und die Distanzwerte der Knoten an. Der Startknoten ist mit s gezeichnet.



Aufgabe 4 (6 Punkte): (Schleifeninvariante)

Betrachten Sie den folgenden Algorithmus:

```
NewAlgo(int n):
1 x ← 1
2 for i ← 1 to n do
3     x ↔ x · 5
4 end
5 return x
```

a) Stellen Sie eine Schleifeninvariante für a auf, die **nach** jeder Iteration gelten soll.

b) Beweisen Sie die Richtigkeit ihrer Schleifeninvariante durch Induktion.

Aufgabe 5 (8 Punkte): (Gierige Algorithmus)

Gegeben seien zwei Mengen A und B. Jede dieser Mengen enthält n natürliche Zahlen, die paarweise verschieden voneinander sind. Sie können die Elemente in A und B in eine beliebige Reihenfolge bringen, so dass a_i das i-te Element in A und b_i das i-te Element in B ist. Es sei $Z = \prod_{i=1}^{n} a_i^{b_i}$.

a) Beschreiben Sie einen gierigen Algorithmus, der den Wert Z maximiert und Z ausgibt. Setzen Sie diese auch in Pseudocode an.
Hinweis: Sie können der Einfachheit halber annehmen, dass A und B durch Arrays dargestellt werden. Es soll also die Reihenfolge der Elemente innerhalb der Arrays neu angeordnet werden.

b) Analysieren Sie die Laufzeit Ihres Algorithmus.

c) Beweisen Sie die Korrektheit Ihres Algorithmus.

Aufgabe 6 (8 Punkte): (Teile und Herrsche)

Gegeben sei ein Array $A[1 \ldots n]$ über einem Datentyp, für den keine Ordnung seiner Elemente vorliegt. Wohl aber ist ein Vergleich $A[i] = A[j]$ in konstanter Zeit möglich. Ein solches Array $A[1 \ldots n]$ besitzt nun ein majorisierendes Element, falls mehr als die Hälfte seiner Elemente denselben Wert besitzt.

a) Beschreiben Sie einen *Teile und Herrsche* Algorithmus, der bei Eingabe des Arrays $P[1..n]$ feststellt, ob ein majorisierendes Element vorhanden ist und, falls dies der Fall ist, dieses auch ausgibt. Geben Sie den Algorithmus auch in Pseudocode an. Für die volle Punktzahl wird ein Algorithmus erwartet, dessen Worst-Case-Laufzeit durch $O(n \cdot \log n)$ beschränkt ist.

b) Analysieren Sie die Laufzeit Ihres Algorithmus.

c) Beweisen Sie, dass Ihr Algorithmus ein majorisierendes Element (wenn vorhanden) korrekt findet.

Aufgabe 7 (8 Punkte): (Graphenalgorithmen)

Ein Graph heißt bipartit genau dann, wenn er keine Kreise ungerader Länge enthält.

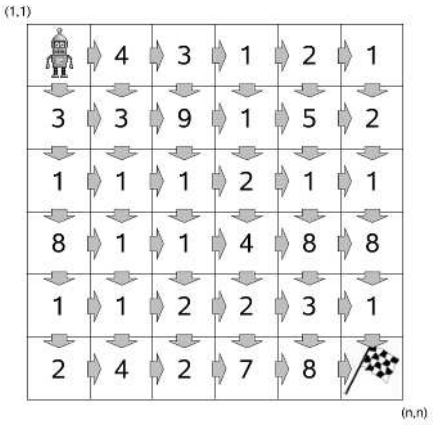
a) Entwickeln Sie einen Algorithmus, der für einen gegebenen Graphen $G(V, E)$ entscheidet, oder dieser bipartit ist oder nicht. Geben Sie zunächst eine knappe umgangssprachliche Beschreibung des Algorithmus an und setzen Sie diese dann in Pseudocode um. Für die volle Punktzahl wird ein Algorithmus mit Laufzeit $O(|V| + |E|)$ erwartet.

b) Analysieren Sie die Laufzeit Ihres Algorithmus.

c) Beweisen Sie die Korrektheit Ihres Algorithmus.

Aufgabe 8 (8 Punkte): (Dynamische Programmierung)

Die europäische Weltraumorganisation ESA beauftragt Sie mit dem Entwurf eines Algorithmus zur Steuerung ihres neuesten Marsroboters. Der Roboter soll sich auf der Marsoberfläche von einem Start- zu einem Zielsektor bewegen. Dabei soll er unterwegs möglichst wertvolle Gesteinsproben sammeln und analysieren. Zu diesem Zweck wird die ESA zuvor via Satellit eine zweidimensionale Karte der Marsoberfläche erstellen. Die Karte teilt den Mars in $n \times n$ gleichgroße quadratische Sektoren ein und bewertet jeden Sektor mit einer positiven Zahl, die den Wert der enthaltenen Gesteinsproben bezeichnet. Wir können uns die Karte als doppelt indiziertes Array $A[1..n, 1..n]$ vorstellen, wobei $A[i, j]$ dem Gesteinsprobenwert im Sektor (i, j) entspricht. Die Startposition des Roboters befindet sich in Sektor $(1, 1)$, sein Ziel liegt in Sektor (n, n). Um Energie zu sparen und sein Ziel so schnell wie möglich zu erreichen, darf der Roboter sich lediglich entlang einer der beiden Koordinatenachsen auf den Zielsektor zu (d.h. "nach rechts" bzw. "nach unten") bewegen. Diagonale Züge sind nicht erlaubt. Ferner darf der Roboter die Karte natürlich nicht verlassen, und er bewegt sich auch nicht rückwärts (Schritte "nach links" oder "oben" sind also nicht erlaubt). Das folgende Bild zeigt ein Beispiel.



a) Bezeichne $M[i, j]$ den maximalen Gesamtwert der Gesteinsproben, den ein Roboter auf einer durch A gegebenen Karte entsprechend der oben genannten Bewegungsregeln auf dem Weg von Sektor $(1, 1)$ bis Sektor (i, j) einsammeln kann. Geben Sie eine rekursive Formulierung für $M[i, j]$ für alle $1 \leq i, j \leq n$ an. Achten Sie darauf, die Rekursion vollständig und inklusive Rekursionsabbruch anzugeben.

b) Geben Sie in Pseudocode einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung bei Eingabe A den maximalen Gesamtwert der Gesteinsproben eines Weges von Sektor $(1, 1)$ bis (n, n) bestimmt. Der optimale Weg selbst muss dabei nicht explizit bestimmt werden.

c) Analysieren Sie die Laufzeit Ihres Algorithmus.

d) Beweisen Sie die Korrektheit der rekursive Formulierung aus Teilaufgabe a).