Algorithmen und Datenstrukturen 1

SoSe 2022 · Prof. Dr. M. Hoefer
14. Oktober 2022 (Zweitklausur)
Klausurnummer: **0001**

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift)**:** _____

Vorname (Druckschrift)**:** _____

Matrikelnummer: _____

Studiengang: _____

Lösungsskizze

Die folgenden Hinweise sind zu beachten:

- Schreiben Sie Ihren Namen **nur auf dieses Titelblatt**. Sollten Sie Zusatzpapier bekommen, vermerken Sie darauf unbedingt Ihre Klausurnummer **0001**.
- Merken oder notieren Sie sich Ihre Klausurnummer **0001**, da nur unter dieser Nummer die Ergebnisse veröffentlicht werden.
- Diese Klausur ist für die Prüfungsvariante **Algo-1 (beide Teile)** und besteht aus den Aufgaben **1 – 10**.
- Legen Sie Ihre **Goethe-Card** deutlich sichtbar an Ihren Platz, damit wir während der Klausur Ihre Identität überprüfen können.
- Überprüfen Sie bitte, ob die Klausur aus **?** durchnummerierten **Vorder- und Rückseiten** besteht. Beachten Sie, dass die Aufgabenstellungen sowohl auf den Vorder- als auch auf den Rückseiten stehen.
- Es dürfen nur **dokumentenechte Stifte** in den Farben blau und schwarz verwendet werden. Insbesondere ist die Nutzung von Tintenlöschern und Tipp-Ex untersagt. Zugelassene Hilfsmittel:
  1 Blatt DIN A4 mit handschriftlichen Notizen (beidseitig).
- Das Mitbringen nicht zugelassener Hilfsmittel stellt eine Täuschung dar und führt zum Nichtbestehen der Klausur. **Schalten Sie bitte deshalb alle elektronischen Geräte, insbesondere Handys und Smartwatches, vor Beginn der Klausur aus.**
- Sollte der Platz unter einer Aufgabe nicht ausreichen, **nutzen Sie bitte die Zusatzblätter am Ende**. Weitere Blätter sind bei Bedarf erhältlich.
- Wenn sich Ihre Lösung zu einer Aufgabe teilweise oder ganz auf Zusatzblättern befindet, vermerken Sie dies entsprechend bei der Aufgabe und auf dem Zusatzblatt.
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung. Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies explizit verlangt.
- Die Klausur ist mit Sicherheit bestanden, wenn mindestens **50%** der Höchstpunktzahl (ohne Bonifikation aus den Übungen) erreicht wird. Die Bearbeitungszeit beträgt **180 Minuten**.

Viel Erfolg! ☘

1 0012211 00011

Diese Seite ist nur für den internen Gebrauch bestimmt.
Bitte nicht beschreiben.

Lösungsskizze

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Diese Seite ist nur für den internen Gebrauch bestimmt.
Bitte nicht beschreiben.

Lösungsskizze

Aufgabe	1	2	3	4	5	6	7	8	9	10	11
Erreichbar	22	17	14	10	9	16	12	16	17	14	13
Erreicht											

Klausur	Bonifikation

1 0003221 100015

Aufgabe 1: Asymptotische Notation und Codeanalyse — $6 + 6 + 6 + 4 = 22$ Punkt(e)

a) Geben Sie jeweils eine Funktion $f : \mathbb{N} \mapsto \mathbb{R}$ an, welche die angegebenen Wachstumseinschränkungen einhält.

$f \in \Omega(n)$ $\quad$ $f \in o(n \log n)$ $\quad$ $f(n) =$ n

$f \in \mathcal{O}(\sqrt{n})$ $\quad$ $f \in \omega\left((\log n)^2\right)$ $\quad$ $f(n) =$ $\sqrt{n}$

$f \in \Theta(2^f)$ $\quad$ $f(n) =$ 1

Lösungsskizze: Siehe Textfelder.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

b) Lösen Sie die folgenden Rekursionsgleichungen auf. Für alle Rekursionsgleichungen gilt $T(1) = 1$. Nehmen Sie vereinfachend an, dass n so gewählt ist, so dass sämtliche Divisionen restfrei aufgehen.

$T(n) = 2 \cdot T(\frac{n}{2}) + 1$ $\quad$ $T(n) = \Theta($ n $)$

$T(n) = 8 \cdot T(\frac{n}{2}) + n^4$ $\quad$ $T(n) = \Theta($ n^4 $)$

$T(n) = 4 \cdot T(\frac{2}{4}n) + \sum_{i=0}^{n} i$ $\quad$ $T(n) = \Theta($ $n^2 \log n$ $)$

Lösungsskizze: Siehe Textfelder.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

Lösungsskizze

c) Geben Sie für die drei folgenden Algorithmen in Pseudo-Code jeweils die Laufzeit in Abhängigkeit von n in Θ-Notation an.

```
for i = 1...n do
    j = 2n;
    while j ≥ i do
        j = j − 2;
```

$\Theta($ n^2 $)$

```
if n ≤ 2022 then
    i = 1; j = n;
    while i < j do
        i = 2 * i;
        j = j − 1;
```

$\Theta($ 1 $)$

```
for i = 1...n do
    j = 1;
    while j < i do
        j = 2 * j;
```

$\Theta($ $n \log n$ $)$

Lösungsskizze: Siehe Textfelder.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

d) Sei `A[1...n]` ein Array mit Zahlen. Die Rekursion `rek` wird mit folgendem Algorithmus in Pseudo-Code berechnet:

```
rek(int i, int j) {
    if(i >= j) return;
    int h = A[i];
    A[i] = A[j];
    A[j] = h;
    rek(i+1, j-1);
}
```

i) **(2 Punkte)** Beschreiben Sie kurz, welche Modifikation durch den Aufruf `rek(1,n)` am Array `A` vorgenommen wird.

Lösungsskizze: Die Reihenfolge der Elemente wird invertiert.

ii) **(2 Punkte)** Welche der folgenden Rekursionsgleichungen beschreibt die Laufzeit des Aufrufs `rek(1,n)`? Kreuzen Sie genau eine Antwort an.

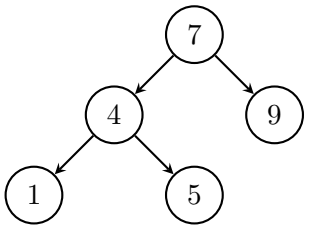
- ☐ $T(n) = 2 \cdot T(n/2) + O(n)$
- ☐ $T(n) = T(n-1) + O(n)$
- ☐ $T(n) = T(n/2) + O(1)$
- ☐ $T(n) = T(n-2) + O(1)$ ✓

Lösungsskizze: Siehe oben.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Wie viele verschiedene Folgen von Einfügeoperationen führen zu dem folgenden binären Suchbaum? Gehen Sie davon aus, dass der Suchbaum am Anfang leer ist und jeder Schlüsselwert genau einmal vorliegt.



Antwort: 8

Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Sei das Array $A = [9, 5, 3, 1, 4]$ ein *Max*-Heap. Geben Sie A nach dem Einfügen der Priorität 12 an.

Antwort: $A = [$ 12, 5, 9, 1, 4, 3 $]$

Lösungsskizze: Siehe Textfeld

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Die Prioritäten $1, ..., n$ werden mittels `insert` in folgender Reihenfolge in einen anfangs leeren Max-Heap eingefügt: $n, n-1, n-2, ..., 3, 2, 1$. Welche asymptotische Gesamtlaufzeit ergibt sich für die n Einfügeoperationen in diesem Fall?

$\Theta($ n $)$

Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

d) Ein binärer Baum wird als *vollständig* bezeichnet, wenn alle Blätter die gleiche Tiefe haben und jeder innere Knoten genau zwei Kinder besitzt.

Sei B ein vollständiger binärer Suchbaum mit $2^k - 1$ ganzzahligen und paarweise verschiedenen Schlüsseln aus dem Intervall $[1, 2^k]$. Welche Schlüssel können an der Wurzel von B gespeichert sein?

Antwort: 2^{k-1}, $2^{k-1}+1$

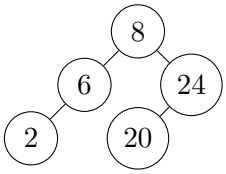
Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Lösungsskizze

1 0006221 100016

e) Gegeben sei der folgende AVL-Baum:



Fügen Sie einen geeigneten ganzzahligen Schlüssel $x > 0$ ein, so dass der resultierende Baum die AVL-Eigenschaft direkt behält und keine Reorganisationen bzw. Reparaturen notwendig sind. Zeichnen Sie den resultierenden Baum.

Schlüssel $x =$ 7 oder Wert $x \geq 25$

Resultierender Baum:

Lösungsskizze: Schlüssel mit Wert $x = 7$ muss rechtes Kind von Schlüssel 6 sein. Schlüssel mit Wert $x \geq 25$ muss rechtes Kind von Schlüssel 24 sein.

⇑⇑⇑ _____ / 5 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Sei T ein (a, b)-Baum mit $b = 2a - 1$, also ein B-Baum. Die Tiefe von T habe sich nach Entfernen eines Schlüssels von 2 auf 1 gesenkt.

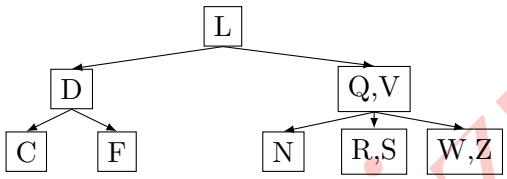
Wie viele Schlüssel befinden sich nach dem Entfernen *mindestens* im Baum? Geben Sie eine exakte untere Schranke in Abhängigkeit von a an.

Antwort: $2(a+1)(a-1)$

Lösungsskizze: Siehe Textfeld.

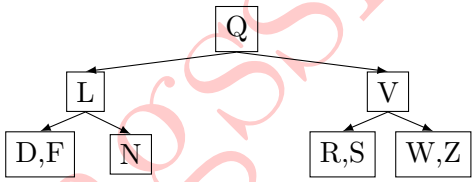
⇑⇑⇑ _____ / 5 Punkt(e) ⇑⇑⇑

b) Betrachten Sie den folgenden (2,3)-Baum:



Führen Sie die Operation `remove(C)` aus. Geben Sie den resultierenden (2,3)-Baum in graphischer Darstellung an.

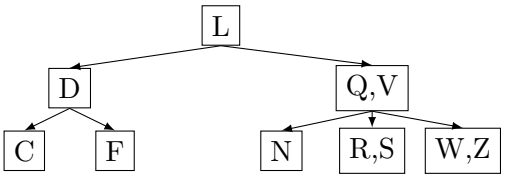
Lösungsskizze:



⇑⇑⇑ _____ / 5 Punkt(e) ⇑⇑⇑

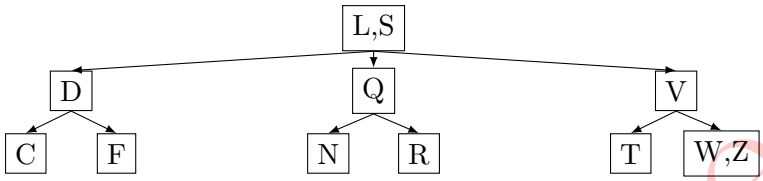
Lösungsskizze

c) Betrachten Sie den folgenden (2,3)-Baum:



Führen Sie die Operation `insert(T)` aus. Geben Sie den resultierenden (2,3)-Baum in graphischer Darstellung an.

Lösungsskizze:



⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Gegeben sei die folgende Hashtabelle der GröSSe $m = 7$.

0	1	2	3	4	5	6
70	22	51	64	39	47	11

Es wird Hashing mit linearem Austesten, $h_i(x) = (f(x) + i) \mod 7$ (für $i = 0, 1, ..., 6$), mit der Hashfunktion $f(x) = x \mod 7$ verwendet.

Fügen Sie die folgenden Schlüssel in der gegebenen Reihenfolge in die obige Hashtabelle ein:

$$64, 39, 11.$$

Lösungsskizze: Siehe Hashtabelle.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

b) Sei nun die TabellengröSSe $m = 11$ und das Universum $U = \{0, 1, 2, \ldots, 28\}$. Welche der folgenden Klassen von Hashfunktionen ist c-universell für das angegebene c? Kreuzen Sie genau eine Antwort an.

☐ $H = \{h_i \mid 0 \leq i < 11 \ , \ h_i(x) = (x + i) \mod 11\}, \quad c = 2$

☐ $H = \{h_{a,b} \mid 0 \leq a, b < 29 \ , \ h_{a,b}(x) = \big((ax + b) \mod 29\big) \mod 11\}, \quad c = \frac{3}{2}$ ✓

☐ $H = \{h_i \mid 0 \leq i < 11 \ , \ h_i(x) = (i \cdot x) \mod 11\}, \quad c = 2$

☐ $H = \{h_{a,b} \mid 0 \leq a, b < 29 \ , \ h_{a,b}(x) = (ax + b) \mod 11\}, \quad c = 1$

Lösungsskizze: Siehe oben.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Lösungsskizze

Gegeben sei ein Min-Heap H mit n Zahlen. Erweitern Sie die Heap-Datenstruktur um eine Operation `smaller(x)`, die alle Zahlen in H ausgibt, die strikt kleiner als `x` sind. Die Laufzeit soll $\mathcal{O}(k)$ sein, wobei k die Anzahl der Elemente in H ist, die strikt kleiner als `x` sind.

Geben Sie Ihr Verfahren in Pseudo-Code an und begründen Sie, warum die Laufzeitschranke eingehalten wird.

Lösungsskizze:

Pseudo-Code:

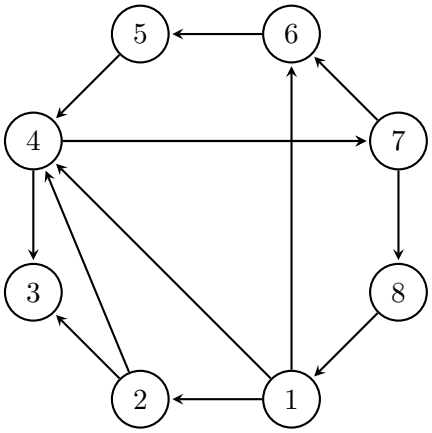
```
void output(int i, int x) {
    if(i > n) return;
    if(H[i] < x) {
        print H[i];
        output(2*i, x);
        output(2*i+1, x);
    }
}

void smaller(int x) {
    output(1, x);
}
```

Laufzeit: Für jedes der k Elemente strikt kleiner x werden nur konstant viele Operationen ausgeführt. Somit ist die Laufzeit $\mathcal{O}(k)$.

Lösungsskizze

Betrachten Sie den folgenden gerichteten Graphen $G = (V, E)$:



a) i) **(2 Punkte)** Welche Kante $e \in E$ muss aus G entfernt werden, damit eine topologische Sortierung *aller* Knoten des Graphen $G' = (V, E')$ mit $E' = E \setminus \{e\}$ möglich ist?

Antwort: $(4, 7)$

Lösungsskizze: Siehe Textfeld.

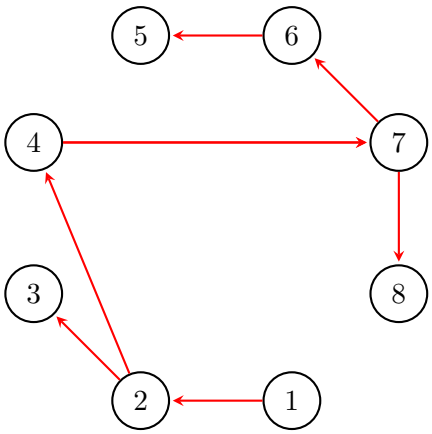
ii) **(2 Punkte)** Geben Sie eine entsprechende topologische Sortierung der Knoten in G' an. Falls in einem Schritt mehrere Knoten gewählt werden können, wählen Sie den kleinsten.

Antwort: 7, 8, 1, 2, 6, 5, 4, 3

Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) i) **(4 Punkte)** Bestimmen Sie den Baum der Tiefensuche für G, wie in der obigen Abbildung gegeben, wenn die Tiefensuche in Knoten 1 gestartet wird. Dabei sollen die unbesuchten Nachbarn eines jeden Knotens in aufsteigender Reihenfolge besucht werden. Sie können die folgende Vorlage verwenden.



Lösungsskizze

Lösungsskizze: Siehe oben.

ii) **(5 Punkte)** Geben Sie alle Quer-, Rückwärts, und Vorwärtskanten an.

Querkanten: $(4, 3)$

Rückwärtskanten: $(5, 4), (8, 1)$

Vorwärtskanten: $(1, 4), (1, 6)$

Lösungsskizze:

⇑⇑⇑ _____ / 9 Punkt(e) ⇑⇑⇑

c) Auf einem gerichteten, stark zusammenhängenden Graphen $G = (V, E)$ wird eine Tiefensuche ausgeführt. Geben Sie die Summe der Anzahl aller Vorwärts-, Rückwärts-, und Querkanten in Abhängigkeit von $|E|$ und $|V|$ an.

Antwort: $|E| - |V| + 1$

Lösungsskizze: $|E| - (|V| - 1) = |E| - |V| + 1$ (beide Ausdrücke ok).

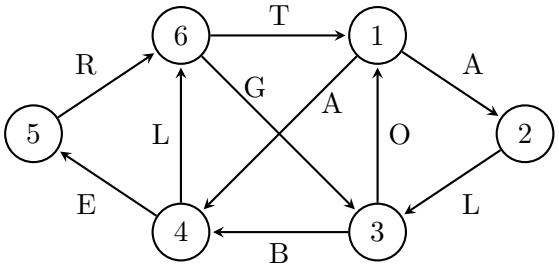
⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Lösungsskizze

Gegeben sei ein gerichteter Graph $G = (V, E)$ in Adjazenzlistendarstellung. Jede Kante des Graphen hat als Beschriftung einen GroSSbuchstaben.

Ziel ist es, einen Weg der Länge vier zu finden, dessen Kantenbeschriftungen mit der Zeichenkette `ALGO` übereinstimmen, oder zu verifizieren, dass es keinen solchen Weg gibt.

a) Betrachten Sie folgende Instanz des Problems:



Geben Sie einen Weg an, dessen Kantenbeschriftungen mit der Zeichenkette `ALGO` übereinstimmen.

Antwort: $(1, 4, 6, 3, 1)$

Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Lösungsskizze

b) Entwerfen Sie einen Algorithmus, der entscheidet, ob in einem gegebenen Graphen G ein zum String `ALGO` passender Weg existiert oder nicht. Die asymptotische Laufzeit soll $\mathcal{O}(|V| + |E|)$ nicht übersteigen.

Beschreiben Sie Ihren Algorithmus in Worten. Begründen Sie auch, warum die Laufzeitschranke eingehalten wird.

Lösungsskizze:
Idee: Führe für jeden Buchstaben `i=1,2,3,4` eine Tiefensuche auf G aus und markiere dabei alle Knoten mit `i`, zu denen ein Pfad führt, dessen Beschriftung mit dem Präfix von `ALGO` bis zum `i`-ten Buchstaben übereinstimmt. Ist am Ende mindestens ein Knoten mit `4` markiert, existiert ein Pfad, sonst nicht.

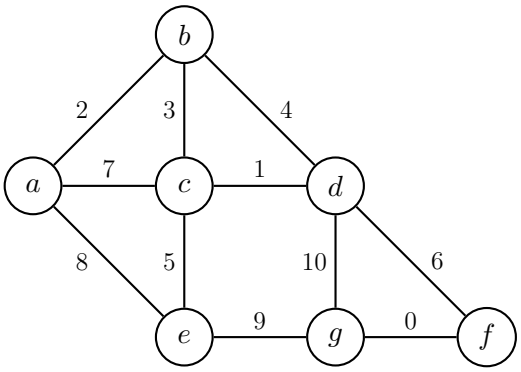
Algorithmus: Anfangs sind alle Knoten mit `0` markiert. Für jeden Buchstaben `i=1,2,3,4` des Strings wird eine Tiefensuche auf G ausgeführt. In der `i`-ten Tiefensuche werden die Knoten in G wie folgt markiert: Wird eine mit dem `i`-ten Buchstaben beschriftete Kante $(u, v) \in E$ besucht, und ist u mit `i-1` markiert, dann wird v mit `i` markiert. Wichtig ist, dass Markierungen auch überlappen können, das heiSSt, ein Knoten kann mehrere Markierungen aus $\{1, 2, 3, 4\}$ gleichzeitig haben. Da es nur konstant viele Markierungen gibt, kann deren Abfrage in Konstantzeit erfolgen.

Laufzeit: Die Laufzeit der einzelnen Tiefensuchen entspricht $\mathcal{O}(|V| + |E|)$, da durch das Markieren der Knoten jede Operation nur um konstanten Aufwand erhöht wird. Da nur vier solcher Tiefensuchen durchgeführt werden, ist die Laufzeit $\mathcal{O}(|V| + |E|)$.

⇑⇑⇑ _____ / 9 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Gegeben sei der folgend dargestellte ungerichtete, gewichtete Graph G:



Falls benötigt, sei a der Startknoten. Für die beiden folgenden Teilaufgaben i), ii) und iii) ist die Reihenfolge, in der gleichwertige Kanten/Knoten ausgewählt werden, beliebig.

i) **(5 Punkte)** Welche Kante wird vom DJP-Algorithmus als *viertes* in den minimalen Spannbaum von G eingefügt?

Antwort: $\{c, e\}$

Lösungsskizze: Siehe Textfeld.

ii) **(5 Punkte)** Welche Kante wird von Kruskals Algorithmus als erste *nicht* in den minimalen Spannbaum von G eingefügt?

Antwort: $\{b, d\}$

Lösungsskizze: Siehe Textfeld.

iii) **(3 Punkte)** Geben Sie die maximale Tiefe eines Baums in der Union-Find Datenstruktur bei der Anwendung von Kruskals Algorithmus auf dieser Instanz an. Hierbei soll in jeder Union-Operation die Wurzel des kleineren Baums unter die Wurzel des gröSSeren Baums gehängt werden. Dabei ist der Baum mit mehr Knoten der gröSSere. Bei gleicher Knotenzahl ist der Baum mit lexikographisch kleinerer Wurzel der gröSSere.

Antwort: 2

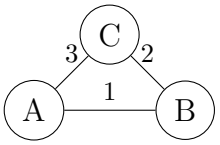
Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 13 Punkt(e) ⇑⇑⇑

Lösungsskizze

b) Gegeben ist ein ungerichteter, zusammenhängender Graph, der kein Baum ist. Ist es möglich, dass der DJP-Algorithmus die schwerste im minimalen Spannbaum enthaltene Kante als erstes hinzufügt? Geben Sie ein begründetes Beispiel an oder begründen Sie, warum das nicht möglich ist.

Lösungsskizze: Wenn Prims Algorithmus im Knoten C startet, wird die Kante $\{A, C\}$ verworfen und die Kante $\{B, C\}$ ausgewählt. Diese ist gleichzeitig die schwerste Kante im minimalen Spannbaum.



Alternativ: Startknoten hat nur eine inzidente Kante, die das gröSSte Gewicht hat, und zum Rest des Graphen führt.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Sei $G = (V, E)$ ein beliebiger ungerichteter Graph mit strikt positiven Kantengewichten. Wir transformieren die Gewichte von G: Das Gewicht $w(e)$ einer Kante $e \in E$ ändert sich zu $w'(e)$.

In welchem der folgenden Fälle bleibt ein minimaler Spannbaum durch die Transformation *nicht immer* minimal? Kreuzen Sie genau eine Antwort an.

☐ $w'(e) = \big(w(e) - 2\big)^2$ ✓

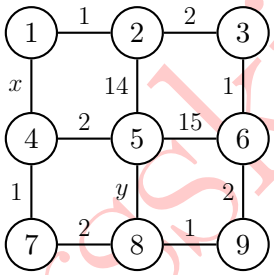
☐ $w'(e) = 3 \cdot (w(e) + 1)$

☐ $w'(e) = \sqrt{w(e)}$

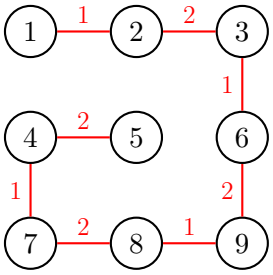
☐ $w'(e) = \log_2(w(e))$

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Gegeben sei der folgende Graph G mit variablen Kantengewichten $x, y \geq 0$:



i) **(4 Punkte)** Bestimmen Sie einen Baum kürzester Wege ausgehend von Knoten 1 für $x = 12$ und $y = 6$. Nutzen Sie folgende Vorlage:



ii) **(3 Punkte)** Sei $D \subseteq \mathbb{R}^2$ die Menge *aller* Paare (x, y), so dass der Baum aus i) weiterhin ein Baum kürzester Wege für Startknoten 1 ist. Kreuzen Sie die richtige Antwort an:

☐ $D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \geq 99, y \geq 99\}$

☐ $D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \geq 12, y \geq 6\}$

☐ $D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \leq 10, y \leq 5\}$

☐ $D = \{(x, y) \subseteq \mathbb{R}^2 \mid x \geq 10, y \geq 5\}$ ✓

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

Lösungsskizze

c) Im Folgenden wird *Stabiles Matching* für zwei Mengen $A = \{a_1, a_2\}$ und $B = \{b_1, b_2\}$ betrachtet.

Gegeben seien folgende Präferenzlisten:

$a_1 : b_1 \succ b_2$ $\quad$ $b_1 : a_2 \succ a_1$

$a_2 : b_2 \succ b_1$ $\quad$ $b_2 : a_1 \succ a_2$

Geben Sie *alle* stabilen Matchings für obige Instanz an.

Antwort: $\{(a_1, b_1), (a_2, b_2)\}$ und $\{(a_1, b_2), (a_2, b_1)\}$

Lösungsskizze: Siehe Textfeld.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

d) Ist $a \to 1$, $b \to 0$, $c \to 11$, $d \to 10$ ein Huffman-Code für folgenden Text: `"aaabaabacd"` ?

☐ Ja

☐ Nein ✓

Lösungsskizze: Siehe oben.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Lösungsskizze

Entlang einer *EinbahnstraSSe* befinden sich die Bushaltestellen $0, 1, \ldots, n$. An *jeder* Haltestelle kann entweder das Ticket WEIT oder das Ticket NAH gekauft werden. Die Preise für die Tickets sind jeweils abhängig von der Haltestelle, wobei Ticket WEIT auch günstiger sein kann als Ticket NAH. An Haltestelle i kostet das Ticket WEIT $W(i)$ Euro und das Ticket NAH $N(i)$ Euro. Mit dem Ticket WEIT kann bis zu *zwölf* Haltestellen weiter gefahren werden, mit dem Ticket NAH nur *eine*.

Ziel ist es, mit möglichst geringen Kosten von Haltestelle 0 zu Haltestelle n mit dem Bus zu fahren.

Um das Problem zu lösen, werden Teilprobleme für $i \in \{0, 1, \ldots, n\}$ wie folgt definiert:

$$\text{opt}(i) = \text{minimale Kosten, um von Haltestelle 0 zu Haltestelle } i \text{ zu gelangen.}$$

a) Geben Sie eine Rekursionsgleichung sowie Basisfälle für obige Teilprobleme an.

Basisfälle:

Lösungsskizze: $opt(0) = 0$

Rekursionsgleichung:

Lösungsskizze:
Für $i > 0$:

$$opt(i) = \min\left\{opt(i-1) + N(i-1),\ \min_{j=1}^{\min\{12,i\}} \big(opt(i-j) + W(i-j)\big)\right\}$$

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

Lösungsskizze

b) Entwickeln Sie aus Ihrer Rekursionsgleichung ein dynamisches Programm zur Berechnung der minimalen Kosten, um von Haltestelle 0 zu Haltestelle n mit dem Bus zu fahren. Ihr Algorithmus soll Laufzeit $\mathcal{O}(n)$ haben.

Geben Sie Ihren Algorithmus in Pseudo-Code an und analysieren Sie dessen Laufzeit.

Lösungsskizze:

$opt(0) = 0$;
for $i = 1 \ldots n$ **do**
    $nah = opt(i-1) + N(i-1)$;
    $weit = \infty$;
    $maxj = 12$;
    if $i < 12$ **then**
        $maxj = i$;
    for $j = 1 \ldots maxj$ **do**
        if $opt(i-j) + W(i-j) < weit$ **then**
            $weit = opt(i-j) + W(i-j)$;
    $opt(i) = \min\{nah, weit\}$;
Gebe $opt(n)$ aus;

Laufzeit: Die äuSSere Schleife wird n mal durchlaufen. Die innere Schleife wird maximal 12 mal durchlaufen. Insgesamt ergibt sich also $\Theta(n)$ (die Angabe $\mathcal{O}(n^2)$ genügt).

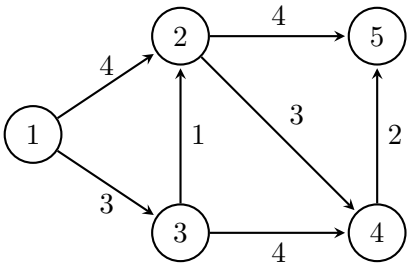
⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

Lösungsskizze

Im Problem *Single-Source-Min-Max-Paths (SSMMP)* ist ein gerichteter Graph $G = (V, E)$ und ein Startknoten $s \in V$ gegeben. Jede Kante $e \in E$ hat ein Gewicht $w(e) > 0$. Gesucht ist die *Min-Max-Distanz* von Knoten s zu allen anderen Knoten $v \neq s$. Die Min-Max-Distanz $\mathrm{MMD}(u, v)$ von Knoten u zu Knoten v ist dabei definiert als

$$\mathrm{MMD}(u, v) = \min_{P \text{ ist Pfad von } u \text{ nach } v} \max_{e \text{ ist Kante des Pfades } P} w(e)$$

a) Geben Sie die Min-Max-Distanz von Knoten 1 zu Knoten 5 für folgenden Graphen an:



MMD(1, 5) = 3

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Entwerfen Sie einen möglichst effizienten Algorithmus, der das Problem für beliebige Instanzen löst. Sie können davon ausgehen, dass der Graph in Adjazenzlistendarstellung gegeben ist. Ausgabe Ihres Algorithmus sollen die Min-Max-Distanzen von s zu allen anderen Knoten sein.

Beschreiben Sie Ihren Algorithmus in Worten und analysieren Sie seine worst-case Laufzeit. Begründen Sie, warum Ihr Verfahren funktioniert.

Lösungsskizze:
Algorithmus: Es kann Dijkstras Algorithmus mit geänderten Prioritäten im Heap verwendet werden. Anstatt die minimale Summe der Kantengewichte eines S-Weges zu den Knoten aus $V \setminus S$ als Priorität zu verwenden, wird nun das Gewicht der gröSSten Kante eines besten S-Weges benutzt. Ein bester S-Weg ist dabei ein S-Weg, welcher die maximale Kante auf dem Weg minimiert. Wird an einer Stelle im Algorithmus ein S-Weg um eine Kante verlängert, wird anstatt das Gewicht der neuen Kante zu addieren, das Maximum des Weges ggf erhöht (wenn die Kante gröSSer ist als das bisherige Maximum). Dies ist nach wie vor in Konstantzeit möglich.

Laufzeit: Die Laufzeit entspricht der von Dijkstra und damit $\mathcal{O}\big((|V| + |E|) \log |V|\big)$, da lediglich das Aktualisieren von Weglängen verändert wurde, und dies wie oben beschrieben nach wie vor in Konstantzeit erfolgt.

Korrektheit: Wir zeigen, dass unter der Annahme, dass die berechneten Distanzen zu allen bisher gewählten Knoten S korrekt sind, die Distanz zum nächsten gewählten Knoten $w \in V \setminus S$ ebenfalls korrekt ist. Wir nehmen an, es sei nicht so. Dann gibt es einen besseren Weg p von s zu w als den gewählten. Dieser Weg p muss einen Knoten v aus $V \setminus S$ enthalten, sonst wäre p vom Algo gewählt worden. Die maximale Kante auf dem Teilweg von s zu v ist dann aber höchstens so groSS wie die maximale Kante auf dem gesamten Weg p. Damit hat v einen geringeren Distanzwert als w, und wäre daher vom Algo anstatt w gewählt worden. Widerspruch.

⇑⇑⇑ _____ / 9 Punkt(e) ⇑⇑⇑

Lösungsskizze

1 0222211 100016

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Hinweis auf Seite 23 platziert wurde.

Lösungsskizze

Notizseite

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Hinweis auf Seite 24 platziert wurde.

Lösungsskizze