Algorithmen und
Berechnungskomplexität I
Wintersemester 2017/18

**Abgabe: 09.01.2018, 12.00 Uhr**
**Besprechung: KW 3**

UNIVERSITÄT BONN

PD Dr. Elmar Langetepe
David Kübel
*Institut für Informatik*

# Übungsblatt 11

*Hinweis:* Dieses Übungsblatt ist zur Vorbereitung auf die Klausur konzipiert. Wie auch in der Klausur, gibt es 120 Punkte. Jeder Minute der Bearbeitungszeit entspricht also ein Punkt. Für die Anrechnung dieser Probeklausur als Übungsblatt gilt der Umrechnungsschlüssel **5 Probeklausurpunkte = 1 Übungspunkt**.

## Aufgabe 11.1: (3+3+3+2+6+6 = 23 Punkte)

(a) Nennen Sie 3 grundlegende algorithmische Paradigmen/Entwurfsmuster. Geben Sie zu jedem Paradigma einen in Vorlesung bzw. Übung besprochenen Algorithmus an, welchem dieses Paradigma zu Grunde liegt.

(b) Betrachten Sie die folgenden Sortieralgorithmen, wie sie in der Vorlesung vorgestellt wurden. Sei $n$ die Anzahl zu sortierender Werte. Ordnen Sie jedem Algorithmus den asymptotischen Speicherbedarf zu, welcher *zusätzlich* zur Eingabe benötigt wird. Machen Sie dazu in der entsprechenden Spalte einer Zeile *genau* ein Kreuz.

| | $O(1)$ | $O(\log(n))$ | $O(n)$ | $O(n \log n)$ | $O(n^2)$ |
|---|---|---|---|---|---|
| Insertionsort | | | | | |
| Mergesort | | | | | |
| Heapsort | | | | | |
| Quicksort | | | | | |
| Radixsort | | | | | |
| Bucketsort | | | | | |

(c) Ordnen Sie die folgenden Funktionen aufsteigend nach asymptotischem Wachstum:

- $f(x) = x^3 - 3 \cdot \log(x) + x \cdot \sqrt{x}$
- $g(x) = x \cdot \log(x)$
- $h(x) = 0.5^x + 4$
- $i(x) = e^{\ln(2^x)}$

(d) Ist die Aussage $\frac{n^{15}}{3^n} \in O(1)$ korrekt? Begründen Sie!

(e) Zeigen Sie, dass $\log n! \in \Theta(n \log n)$ ist.

(f) Betrachten Sie folgende Rekursionsgleichung $T : \mathbb{N} \to \mathbb{N}$ mit

$$T(n) = \begin{cases} 1, & \text{falls } n = 0, \\ T(n-1) + 3n + 5, & \text{falls } n \geq 1. \end{cases}$$

Stellen Sie eine Vermutung für eine geschlossene Form dieser Rekursionsgleichung auf und beweisen Sie diese per Induktion.

## Aufgabe 11.2: ( = 14 Punkte)

Seien $(a_n, \ldots, a_1)$ und $(b_n, \ldots, b_1)$ die Binärdarstellungen zweier natürlicher Zahlen $a$ und $b$ der Länge $n$, wobei $n$ eine Zweierpotenz ist. Wir wollen die Binärdarstellung $(c_{2n}, \ldots, c_1)$ des Produktes $c = a \cdot b$ bestimmen.

(a) Geben Sie die Laufzeit der klassischen Schulmethode zur Multiplikation von $a$ und $b$ an.

(b) Geben Sie einen Divide-and-Conquer-Algorithmus zur Multiplikation von $a$ und $b$ an, der mit vier Multiplikationen von Zahlen der Länge $n/2$ auskommt. Welche Laufzeit hat diese Methode?

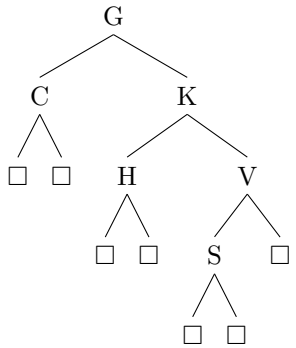
(c) Geben Sie einen Divide-and-Conquer-Algorithmus zur Multiplikation von $a$ und $b$ an, der mit drei Multiplikationen von Zahlen der Länge $n/2$ auskommt. Welche Laufzeit hat diese Methode?

## Aufgabe 11.3: (4+3+3+3+6 = 19 Punkte)

Diese Aufgabe behandelt die Datenstruktur AVL-Baum, wie sie in der Vorlesung definiert wurde. Zur Erinnerung: Die Wurzel eines Baumes liegt in Höhe 0 und jeder Blattknoten ist leer.

(a) Geben Sie die in der Vorlesung verwendete Definition eines binären Such- bzw. Blattsuchbaums.

(b) Begründen Sie, warum es sich bei folgendem Baum nicht um einen AVL-Baum handelt.




(c) Angenommen der Baum aus Teilaufgabe b) ist eine Zwischenkonfiguration während der Einfüge-Operation von Schlüssel S in einen AVL-Baum. Führen Sie den Einfüge-Vorgang durch entsprechende Rotation(en) zu Ende, sodass der Baum schließlich wieder um ein AVL-Baum ist.

(d) Zeigen Sie, dass der Binärbaum aus Teilaufgabe b) ein Rot-Schwarz Baum ist, d. h. eine Färbung besitzt, die die Rot-Schwarz Baum Eigenschaft erfüllt.

(e) Beweisen Sie, dass in jedem AVL-Baum gilt: Der längste direkte Pfad von der Wurzel zu einem seiner Blätter ist höchstens doppelt so lange wie der kürzeste.

## Aufgabe 11.4: (5+7+4+6 = 22 Punkte)

Gegeben sei eine Menge $\mathcal{M}$ von $n$ Münzen mit Werten $v_1, v_2, ..., v_n \in \mathbb{N}$. Sie sollen nun einen bestimmten Betrag $S$ mit Hilfe dieser Münzen zusammensetzen, d.h eine Teilmenge $U \subseteq 1, 2, ..., n$ finden, so dass $S = \sum_{i \in U} v_i$. Jede Münze darf dabei nur einmal verwendet werden.

Mittels Dynamischer Programmierung soll nun die minimale Anzahl an Münzen aus $\mathcal{M}$ bestimmt werden, die dazu nötig ist.

(a) Vervollständigen Sie die folgende rekursive Funktion $M(i, b)$. Diese gibt die minimale Anzahl von Münzen an, die nötig ist um unter Verwendung der ersten $i$ Münzen den Betrag $b$ zu erzielen, falls dies überhaupt möglich ist.

$$M(b, i) = \begin{cases} 0 & : \text{falls } b = 0 \\ \infty & : \text{falls } b < 0 \vee (i = 0 \wedge b > 0) \\ & \\ & : \text{falls} \end{cases}$$

(b) Formulieren Sie mit Hilfe der Rekursionsformel aus Teil a) *in Stichworten* einen möglichst effizienten Algorithmus, der $M(S, |\mathcal{M}|)$ mittels Dynamischer Programmierung berechnet. Geben Sie dabei Eingabe und Ausgabe an und denken Sie an die Initialisierung verwendeter Datenstrukturen und Variablen.

(c) Geben Sie Laufzeit und Speicherplatzbedarf Ihres Algorithmus aus Aufgabenteil b) an. Begründen Sie kurz Ihre Angaben!

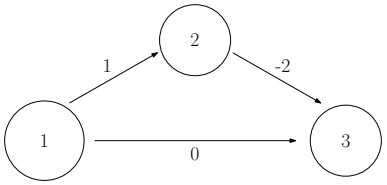
(d) Ein Kommilitone hat zur Lösung des obigen Problems einen Greedy-Algorithmus vorgeschlagen. Dieser Algorithmus fügt in jedem Schritt die Münze zur Lösung hinzu, deren Wert $v$ gerade noch kleiner gleich dem Betrag $b$ ist. Anschließend entfernt er die Münze aus $\mathcal{M}$ und wiederholt den Schritt mit den verbleibenden Münzen und dem Betrag $b - v$.

Wir gehen nun davon aus, dass die Menge $\mathcal{M}$ so beschaffen ist, dass der Greedy-Algorithmus für jeden Betrag $S$ eine Lösung findet.

- Zeigen Sie anhand eines Beispiels, dass die Lösung des Greedy-Algorithmus nicht immer optimal ist.
- Beweisen Sie, dass der Greedy-Algorithmus für kein $c > 0$ eine $c$-Approximation liefert.

**Aufgabe 11.5:** (4+4+6+8 = 22 Punkte)

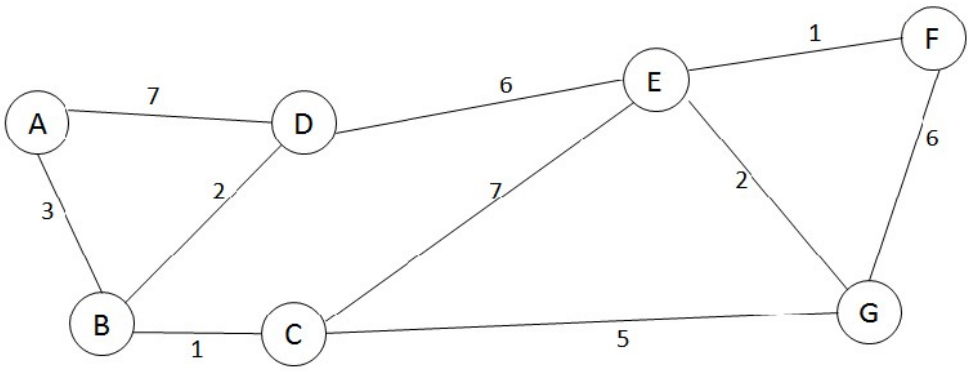
(a) Obwohl der nachfolgend abgebildete Graph negative Kantengewichte aufweist, würde der Algorithmus von Dijkstra in der Form, in der er in der Vorlesung vorgestellt wurde, die korrekten Distanzen vom Knoten 1 zu 2 und 3 bestimmen. Erweitern Sie den Beispielgraphen um genau einen Knoten, so dass mindestens ein Knoten nicht mehr die korrekte Distanz zugewiesen bekommt.




(b) Argumentieren Sie, ob die Korrektheit des Dijkstra-Algorithmus für negative Kantengewichte wiederhergestellt werden kann, indem vor seiner Anwendung zunächst das kleinste Kantengewicht ermittelt wird, sodann sämtliche Kantengewichte um dessen Betrag inkrementiert werden, und nach der Anwendung des Algorithmus diese Änderungen wieder rückgängig gemacht werden.

(c) Sei $G = (V, E)$ ein Graph und $u, v \in V$ zwei seiner Knoten. Geben Sie die Definition der Vorlesung, für einen Weg bzw. kürzesten Weg zwischen $u$ und $v$ in $G$ und den Kürzesten-Wege-Baum mit Wurzel $u$.

(d) Gegeben sei der nachfolgend abgebildete Graph $G$. Berechnen Sie unter Verwendung des Algorithmus von Dijkstra einen kürzesten Weg vom Knoten $A$ zum Knoten $F$. Erweitern Sie die nachfolgende Tabelle, um alle Zwischenschritte anzugeben.



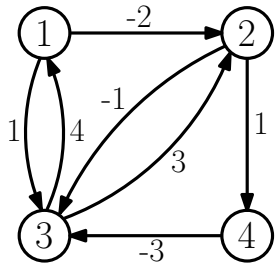

| Schritt | Besuchter Knoten | A | B | C | D | F | G | S |
|---|---|---|---|---|---|---|---|---|
| 0. | - | 0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\emptyset$ |
| 1. | | | | | | | | |
| 2. | | | | | | | | |
| 3. | | | | | | | | |
| 4. | | | | | | | | |
| 5. | | | | | | | | |
| 6. | | | | | | | | |
| 7. | | | | | | | | |
| 8. | | | | | | | | |

**Aufgabe 11.6:** (4+4+6+6 = 20 Punkte)

(a) Der Floyd-Warshall-Algorithmus erzeugt auf dem unten abgebildeten Graphen eine Sequenz von Matrizen, die wir mit $D^{(0)}, D^{(1)}, \ldots, D^{(4)}$ bezeichnen. Die ersten drei dieser Matrizen sind gegeben durch

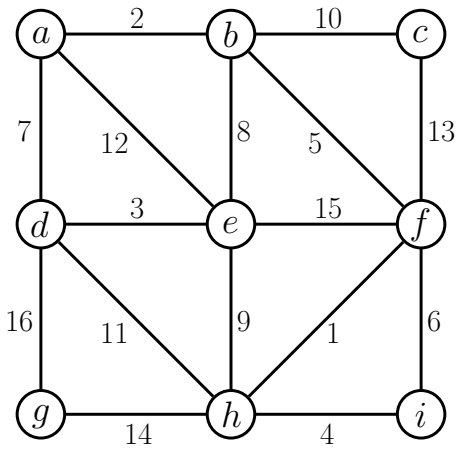
$$D^{(0)} = \begin{bmatrix} 0 & -2 & 1 & \infty \\ \infty & 0 & -1 & 1 \\ 4 & 3 & 0 & \infty \\ \infty & \infty & -3 & 0 \end{bmatrix}, \quad D^{(1)} = \begin{bmatrix} 0 & -2 & 1 & \infty \\ \infty & 0 & -1 & 1 \\ 4 & 2 & 0 & \infty \\ \infty & \infty & -3 & 0 \end{bmatrix} \quad \text{und} \quad D^{(2)} = \begin{bmatrix} 0 & -2 & -3 & -1 \\ \infty & 0 & -1 & 1 \\ 4 & 2 & 0 & 3 \\ \infty & \infty & -3 & 0 \end{bmatrix}.$$

Geben Sie die Matrix $D^{(3)}$ an.




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

(b) Bestimmen Sie mithilfe des Algorithmus von Kruskal einen minimalen Spannbaum auf dem unten abgebildeten Graphen.




(c) Sei $G = (V, E)$ ein zusammenhängender ungerichteter Graph mit einer Kantengewichtung $w\colon E \to \mathbb{R}_{>0}$ derart, dass keine Kantengewichte mehrfach vorkommen. Die Kanten von $G$ seien so mit $e_1, \ldots, e_m$ durchnummeriert, dass $w(e_1) < w(e_2) < \ldots < w(e_m)$ gilt. Des Weiteren sei $T$ ein minimaler Spannbaum von $G$ bezüglich der Kantengewichtung $w$. Geben Sie für $i = 1$, für $i = 2$ und für $i = 3$ eine sowohl notwendige als auch hinreichende Bedingung an, unter der $e_i \in T$ gilt. Begründen Sie Ihre Antwort.

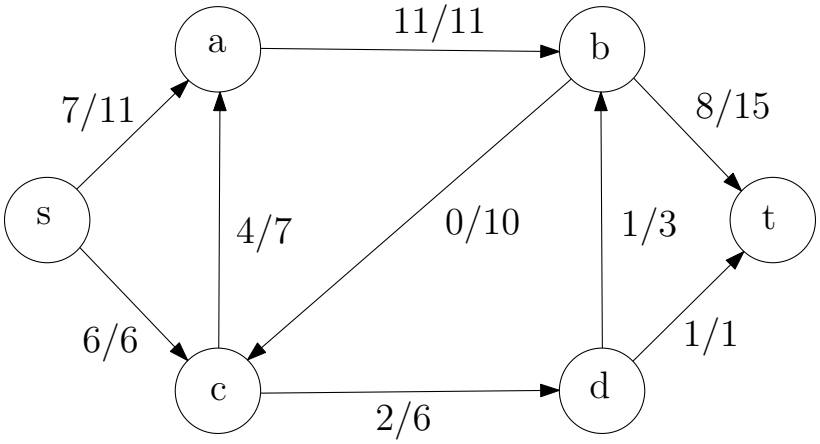
*Hinweis*: Da der Algorithmus von Kruskal nur einen möglichen minimalen Spannbaum liefert, können Sie nicht ohne Weiteres die Vorgehensweise des Algorithmus von Kruskal zur Argumentation heranziehen.

(d) Gegeben seien ein gerichteter Graph $G = (V, E)$ in Adjazenzlistendarstellung mit einer Kantengewichtung $w\colon E \to \{1, 2\}$ und ein Startknoten $s \in V$. Geben Sie einen Algorithmus an, der in Zeit $O(|V| + |E|)$ den Abstand von $s$ zu jedem Knoten $v \in V$ berechnet. Begründen Sie, warum Ihr Algorithmus korrekt ist.

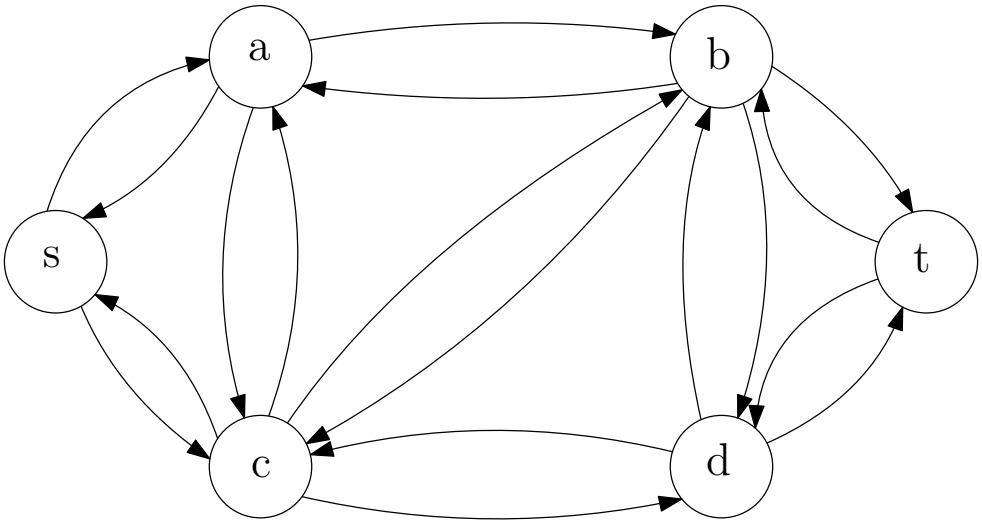
*Hinweis*: Überlegen Sie sich zunächst, wie Sie vorgehen würden, wenn alle Kantengewichte gleich 1 wären.

**Aufgabe 11.7:** (5+5+4 = 14 Zusatzpunkte)

Betrachten Sie folgendes Flussnetzwerk $G$ mit Quelle $s$, Senke $t$ und dem vorgegebenen Fluss $f$ in $G$. Eine Kante mit Beschriftung $X/Y$ hat dabei Kapazität $Y$ und der Fluss $f$ leitet $X$ Einheiten über diese Kante.




(a) Tragen sie in folgende Abbildung die Kapazität der Kanten im Restnetzwerk $G_f$ ein.




(b) Geben Sie einen flussvergrößernden Pfad von $s$ nach $t$ an. Um wieviele Einheiten kann der Fluss über diesen Pfad maximal erhöht werden? Ist der Fluss nach der Erhöhung maximal? Wenn ja, begründen Sie Ihre Anwort. Wenn nicht, geben Sie einen maximalen Fluss in diesem Netzwerk an.

(c) In der Vorlesung haben Sie einen Algorithmus kennengelernt, der einen maximalen Fluss in einem Flussnetzwerk mit $n$ Knoten, $m$ Kanten und ganzzahligen Kantenkapazitäten $c(e) \leq K$ berechnet. Unter welchem Namen ist der Algorithmus bekannt und welches Paradigma liegt diesem Algorithmus zu Grunde? Geben Sie außerdem die asymptotische Worst-Case-Laufzeit des Algorithmus in O-Notation an.