Prof. Dr. Christian Sohler
Nicole Funk, Matthijs Ebbens

Sommersemester 2023

Probeklausur zur Vorlesung

Grundzüge der Informatik 1

Besprechung in der Vorlesung in Kalenderwoche 28

Die Probeklausur ist nicht zulassungsrelevant! Die Bepunktung der Aufgaben dient zur Visualisierung des Bepunktungskonzept der eigentlichen Klausur.

Bitte notieren Sie die folgenden Angaben in **deutlicher Blockschrift**:

Name	
Vorname	
Matrikelnummer	
Studiengang	

Wichtige Hinweise für die Klausur:

- Die Klausur besteht aus 9 Aufgaben. Sollte Ihr Exemplar weniger Aufgaben enthalten, melden Sie sich bitte umgehend bei der Klausuraufsicht.
- Geben Sie das Titelblatt zusammen mit Ihren Lösungen ab und entfernen Sie nicht die Klammerung der Klausur.
- Schreiben Sie auf jedes Blatt Ihren Namen und Ihre Matrikelnummer.
- Sollten Sie weiteres Papier benötigen, wenden Sie sich an die Klausuraufsicht. Beschriften Sie jedes weitere Blatt ebenfalls sofort mit Ihrem Namen und Ihrer Matrikelnummer.
- Sie dürfen nur von uns zur Verfügung gestelltes Papier und dokumentenechte Stifte (also insbesondere keine Bleistifte) verwenden. Verwenden Sie ebenfalls keine Stifte mit roter Farbe. Neben einem handschriftlich beidseitig beschriebenen DIN A4 Blatt sind keine weiteren Hilfsmittel erlaubt. Sollten Sie ein solches beschriebenes Blatt nutzen, geben Sie dieses ebenfalls nach der Bearbeitung der Klausur mit ab.
- Die Bearbeitungszeit beträgt 180 Minuten.

Viel Erfolg!

Aufgabe	1	2	3	4	5	6	7	8	9	Gesamt	Note
maximal	4	4	4	4	5	5	8	8	8	50	
erreicht											

Aufgabe 1 (4) **Punkte**

Wenden Sie den Algorithmus `GreedyLoadBalancing` aus der Vorlesung auf die folgende Eingabe an.

Gegeben seien vier identische Maschinen und zehn Aufgaben mit den Längen

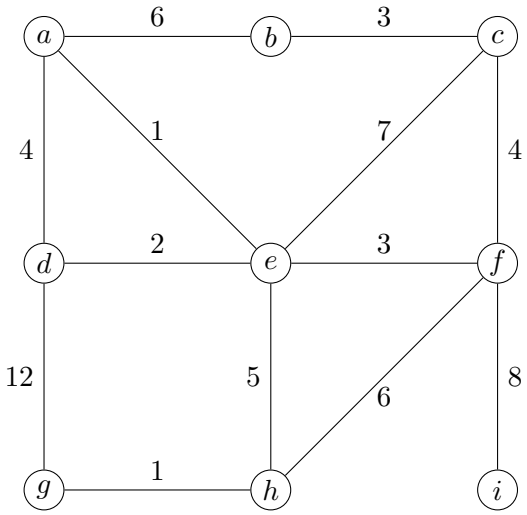
$$t[1..10] = [12, 33, 6, 18, 16, 23, 14, 17, 15, 12]\,.$$

Geben Sie in der Tabelle unten an, in welcher Reihenfolge welche Aufgabe welcher Maschine zugewiesen wird. Falls eine Aufgabe mehreren Maschinen mit gleichem Effekt auf den Makespan zugewiesen werden kann, weisen Sie die Aufgabe der Maschine mit kleinstem Index zu.

1	
2	
3	
4	

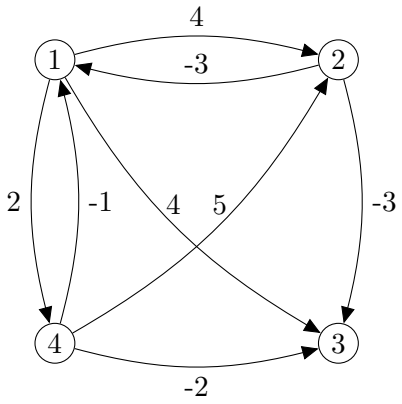
Aufgabe 2 (4) **Punkte**

Bestimmen Sie in dem folgenden ungerichteten Graph mit Hilfe des Algorithmus von Prim aus der Vorlesung einen minimalen Spannbaum, wobei Sie den Knoten a als Startknoten verwenden. Markieren Sie dazu die Kanten, die der Algorithmus von Prim auswählt. Geben Sie außerdem die Reihenfolge an, in der die Kanten ausgewählt werden.

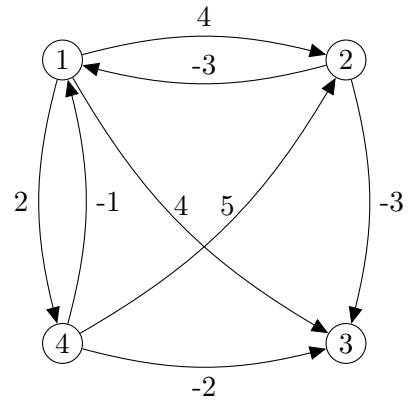


Aufgabe 3 (4) **Punkte**

Gegeben sei der folgende gewichtete und gerichtete Graph G:



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 4 (4) **Punkte**

Betrachten Sie eine anfangs leere Union-Find Datenstruktur, die durch verkettete Listen implementiert ist. Führen Sie nun die folgenden Operationen auf der Datenstruktur aus. Gehen Sie bei der UNION-Operation so vor, dass die Operation möglichst schnell arbeitet und bei gleicher Mengengröße die Menge mit lexikographisch kleinerem Repräsentanten als die kleinere Menge interpretiert wird.

1. MAKE-SET(A)
2. MAKE-SET(B)
3. MAKE-SET(C)
4. MAKE-SET(D)
5. MAKE-SET(E)
6. UNION(B,C)
7. UNION(C,E)
8. UNION(A,D)
9. UNION(D,C)

Zeichnen Sie den Zustand der Datenstruktur nach den Schritten 5, 6, 7, 8 und 9.

Aufgabe 5 (5) **Punkte**

Betrachten Sie den folgenden Algorithmus, welcher als Eingabe eine natürliche Zahle n erhält:

BerechneWert(n):

1. $m = 1$
2. **for** $i = 1$ **to** n **do**
3. $\quad m = m \cdot n$
4. **return** m

a) Stellen Sie eine Behauptung auf, welchen Wert der Algorithmus in Abhängigkeit der Eingabezahl n ausgibt.

b) Formulieren Sie eine Schleifeninvariante, die zu Beginn jeder Iteration der *for*-Schleife (Zeilen 2 bis 5) für die Variable x gilt. Beweisen Sie diese mittels vollständiger Induktion.

c) Verwenden Sie die Schleifeninvariante aus Aufgabenteil b), um zu zeigen, dass die Behauptung aus Aufgabenteil a) korrekt ist.

Aufgabe 6 (5) **Punkte**

Gegeben ist die folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 1 & \text{falls } n = 1 \\ 16T(n/4) + n^3 & \text{sonst} \end{cases}$$

Sie können im Folgenden annehmen, dass n eine Viererpotenz ist.

a) Bestimmen Sie eine möglichst gute Schranke $f(n)$, für die $T(n) \in O(f(n))$ gilt.

b) Zeigen Sie für die in Aufgabenteil a) angegebene Funktion f, dass $T(n) \in O(f(n))$ gilt. Nutzen Sie dazu einen Induktionsbeweis.

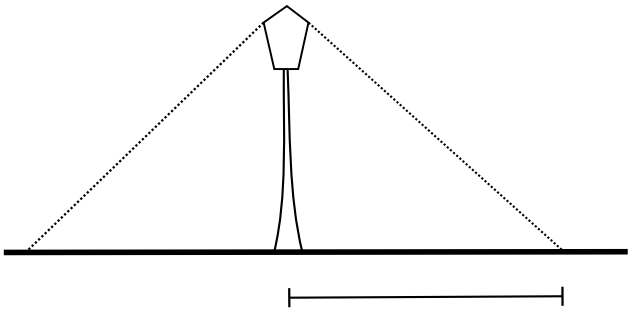
Aufgabe 7 (8) **Punkte**

Das Dorf Kitamrofni möchte seine Straßen beleuchten und hat dafür $n \in \mathbb{N}$ Laternen eingekauft.

Allerdings besitzen die Laternen unterschiedliche Leuchtstärken. Für Laterne i, $1 \leq i \leq n$, beschreibt ℓ_i den Radius (in Metern), der beleuchtet wird. Laterne i deckt also $2\ell_i$ Meter einer Straßenlänge ab. Ein Array $L[1..n]$ enhalte diese Daten.

Es soll nun die Hauptstraße der Länge m, mit $0 < m \leq 2\sum_{i=1}^{n} \ell_i$ (in Metern), ausgestattet werden. Ziel ist es, dabei möglichst wenige Laternen zu verwenden. Jede Laterne kann nur einmal verwendet werden.

Formal wird ein gieriger Algorithmus gesucht, der für eine Eingabe eines Arrays L, der Arraylänge n und der Straßenänge m die minimale Anzahl der Laternen ausgibt, die die Straße der Länge m beleuchten.



Beispiel: Gegeben seien $n = 4$ Laternen mit $\ell_1 = 10$, $\ell_2 = 20$, $\ell_3 = 30$, $\ell_4 = 40$ und eine Straße der Länge $m = 110$. Dann decken Laternen 1, 2 und 3 mit $2 \cdot 10 + 2 \cdot 20 + 2 \cdot 30 = 120 \geq 110$ die Straße ab.

a) Beschreiben Sie einen *gierigen* Algorithmus, der bei Eingabe der Straßenlänge m und dem Array L **die minimale Anzahl an benötigten Laternen** berechnet und zurückgibt Geben Sie den Algorithmus auch in Pseudocode an.

b) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus.

c) Beweisen Sie, dass Ihr Algorithmus eine optimale Lösung berechnet.

Aufgabe 8 (8) **Punkte**

Alice hat ein neues Videospiel. Ein Level dieses Spiels ist ein 2-dimensionales Array A natürlicher Zahlen. Das Array repräsentiert eine Mine, die Alice durchgraben muss, um Erz zu sammeln. Die Werte $A[i][j] = a_{ij}$ in A, $1 \leq i \leq k$ und $1 \leq j \leq \ell$ mit $\ell > 1$, geben dabei die Menge des Erzes in dieser Position an. Das Array A könnte zum Beispiel wie folgt aussehen.

$$A = \begin{pmatrix} 5, & 6, & 12, & 9, & 43 \\ 44, & 16, & 55, & 8, & 1 \\ 101, & 12, & 106, & 18, & 2 \\ 17, & 19, & 22, & 41, & 6 \end{pmatrix}$$

Alice kann von einer beliebigen Position a_{1j}, $1 \leq j \leq \ell$, der ersten (obersten) Reihe beginnen. Danach kann sie sich immer nur in die nächst tiefere Reihe bewegen. Von der Position a_{ij}, $1 \leq i \leq k$ und $1 \leq j \leq \ell$, erreicht sie entweder die Position $a_{(i+1)j}$ unter ihr oder eine diagonal anliegende, also $a_{(i+1)(j+1)}$ oder $a_{(i+1)(j-1)}$, sofern diese existieren. Wegen undurchdingbarem Gestein ist es nicht möglich, die Seiten der Mine zu überschreiten. Ihr Ziel ist es, in der unteren Reihe anzukommen und dabei auf dem Weg möglichst viel Erz zu sammeln.

Lösen Sie das Problem mithilfe von dynamischer Programmierung und gehen Sie dabei wie folgt vor:

a) Geben Sie eine Rekursionsgleichung für eine Funktion $G(i, j)$, $1 \leq i \leq k$ und $1 \leq j \leq \ell$, an, sodass $G(i, j)$ die maximale Menge Erz auf einem Weg ist, der

- die oben beschriebenen Eigenschaften erfüllt,
- in der ersten Reihe beginnt und
- in a_{ij} endet.

b) Zeigen Sie die Korrektheit Ihrer Rekursionsgleichung.

c) Entwerfen Sie mithilfe der Rekursionsgleichung in a) einen Algorithmus, der mit dynamischer Programmierung bei Eingabe des Arrays A die maximale Menge Erz, die Alice sammeln kann, berechnet.

d) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus.

Aufgabe 9 (8) **Punkte**

Entwerfen Sie eine Datenstruktur, die alle folgenden Operationen unterstützt. Es ist **nicht** vorher bekannt wie viele Elemente in die Datenstruktur eingefügt werden.

- `Einfügen(x)`: Fügt ein Element x in die Datenstruktur ein.
- `LöscheNeuestes`: Löscht unter allen Elementen in der Datenstruktur das, was zuletzt eingefügt wurde.
- `LeseNeuestes`: Gibt unter allen Elementen in der Datenstruktur das, was zuletzt eingefügt wurde, zurück.
- `LöscheÄltestes`: Löscht unter allen Elementen in der Datenstruktur das, was am frühesten eingefügt wurde.
- `LeseÄltestes`: Gibt unter allen Elementen in der Datenstruktur das, was am frühesten eingefügt wurde, zurück.
- `Zähle`: Gibt zurück, wie viele Elemente aktuell in der Datenstruktur sind.

Für die volle Punktzahl wird erwartet, dass **alle Operationen eine Worst-Case Laufzeit von $\mathcal{O}(1)$ haben.**

Beschreiben Sie in wenigen kurzen Sätzen, wie Ihre Datenstruktur aufgebaut ist und wie die angegeben Operationen realisiert werden. Wenn notwendig, beschreiben Sie auch ihre Verbundstypen. Hierbei ist kein Pseudocode gefordert. Machen Sie deutlich, dass die Datenstruktur korrekt arbeitet und alle Operationen Worst-Case-Laufzeit O(1) haben.