Wissensfragen: Aufgabe 1

(1) Nennen Sie zwei in der Vorlesung vorgestellte Algorithmen, die nach dem Greedy-Prinzip arbeiten. Begründen Sie kurz, warum.

(2) Nennen Sie einen Vorteil und einen Nachteil von Mergesort gegenüber Quicksort.

(3) Was bedeutet Stabilität bei Sortieralgorithmen? Nennen Sie eine Situation, in der man die Stabilität ausnutzen kann.

(4) Binäre Suche hat auf jeder Datenstruktur eine bessere Laufzeit als Lineare Suche - Beweisen oder widerlegen sie diese Aussage.

(5) Der zweitbeste MST ist eindeutig - Beweisen oder Widerlegen sie diese Aussage

(6) Ein Algorithmus Dijkstra2 arbeitet wie Dijkstra, mit der einzigen Abwandlung, dass die Elemente in der Prioritätswarteschlange absteigend statt wie in Dijkstra aufsteigend sortiert sind und der Relaxierungsschritt als

if $w_{neu}(e) > w_{alt}(e)$ **then** $w_{alt}(e) = w_{neu}(e)$

definiert ist.

Beweisen oder widerlegen sie die Behauptung: Dijkstra2 findet den längsten Pfad in einem Graphen ohne negative Kantengewichte.

Laufzeit: Aufgabe 2

(1) Gegeben sei ein Datentyp, dessen Elemente bezüglich $<$ bzw. $>$ vergleichbar sind. Im folgenden betrachten wir Datenstrukturen für Elemente dieses Typs.

   (a) Warum lässt sich in einem Heap das Minimum nicht in $\Theta(1)$ entfernen, so dass die entstehende Struktur immer noch ein Heap ist?
   Tipp: Analysieren Sie die Laufzeit des sich ergebenden Heapsorts.

   (b) Geben Sie eine Datenstruktur an, die die entsprechende Eigenschaft aus (a) hat.

   (c) Warum ist dies kein Widerspruch zur Überlegung aus (a)?

(2) Lösen Sie folgende Rekurrenzgleichungen:

   (a) $T(n) = T(\frac{n}{2}) + n$

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

   (c) $T(n) = T(\frac{n}{2}) + 1$

(3) Warum lässt sich das Mastertheorem nicht auf folgende Gleichungen anwenden?

   (a) $T(n) = T(\frac{n}{2}) + \log(n)$

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

(c) $T(n) = T(\frac{n}{2}) + n \cdot (2 - \cos(n))$

(d) $T(n) = T(n) + 1$

(4) Seien $f, g, h, j : \mathbb{N} \to \mathbb{N}_0$ monoton wachsend. Zeigen oder widerlegen Sie:

(a) $f \in \Theta(g)$ und $h \in \Theta(j) \Rightarrow f \circ h \in \Theta(g \circ j)$

(b) $f \in \Theta(g)$ und $h \in \Theta(j) \Rightarrow f \cdot h \in \Theta(g \cdot j)$

(c) $f \in \Theta(g)$ und $h \in \omega(j) \Rightarrow f + h \in \Theta(g + j)$

(d) $f \in o(g) \Rightarrow h \circ f \in o(h \circ g)$

(e) $f \in O(g) \Rightarrow h \circ f \in O(h \circ g)$

(f) $f \in o(g) \Rightarrow h \circ f \in O(h \circ g)$

(g) $f \in O(f^2)$

(h) $f \in o(f^2)$

(i) $f \in \Theta(f^2)$

Sortieren: Aufgabe 3

(1) Gegebenen sei der Sortieralgorithmus *Cocktail-Sort*. Um ein Array zu sortieren, durchschreitet der Algorithmus den Array zunächst von vorne nach hinten und vertauscht dabei zwei benachbarte Werte, wenn sie in der falschen Reihenfolge stehen. Ist der Algorithmus am Ende des Arrays angekommen, durchschreitet er das Array jetzt in umgekehrter Reihenfolge von hinten nach vorne (also „zurück“).

(a) Wenden Sie Cocktail-Sort auf folgendes Array an. Geben Sie dabei das Array nach jedem Schritt an.

4	2	1	3	7	3

(b) Geben Sie eine naive Implementierung von *Cocktail-Sort* in Pseudocode an.

(c) Geben Sie eine *Familie* von Zahlenarrays an, die von Bubblesort in $O(n^2)$ sortiert werden, aber von Cocktail-Sort in $O(n)$.

Rot-Schwarz-Bäume: Aufgabe 4

(1) Schreibe die Methode $bh(x, T)$, die für jeden Knoten x seine Schwarzhöhe zurückgibt.

(2) Schreibe eine Methode $check\_b\_r\_tree(T)$, die einen Rot-Schwarz-Baum als Parameter nimmt und überprüft, ob dieser ein gültiger Rot-Schwarz-Baum ist.

(3) (a) Ein Rot-Schwarz-Baum habe die Höhe h. Berechne die maximale Differenz zwischen Entfernungen von der Wurzel zu den NIL-Knoten (mit kurzer Begründung).

(b) Konstruiere einen solchen Rot-Schwarz-Baum für $h = 3$.

(4) (a) Gegeben sei ein Rot-Schwarz-Baum der Höhe h mit genau einem roten Knoten. Dieser habe den Abstand d von der Wurzel. Berechne für jedes $n \in \mathbb{N}$ die Anzahl $a(n)$ der NIL-Knoten, die den Abstand n zur Wurzel haben.

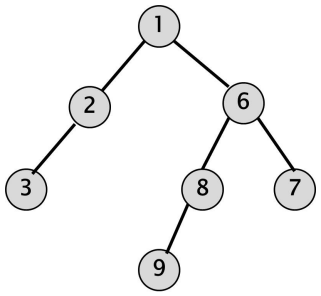
(b) Ist es möglich, dass $a(h) = 2^h$? Begründe!

Datenstrukturen: Aufgabe 5

(1) (a) Geben sie für $n \in \{0, 1, 2, 3, 4\}$ alle Möglichkeiten an, die Zahlen $1, \ldots, n$ in einen Min-Heap einzuordnen. Der Heap soll minimale Höhe haben und ein rechter Unterbaum darf nicht höher sein als ein linker Unterbaum desselben Knotens.

(b) Welche Heaps aus (a) haben in der Arraydarstellung keine Lücken?

(c) Stellen Sie den folgenden Heap als Array dar. Verwenden Sie $\perp$, um Lücken zu kennzeichnen:



(2) Gesucht ist eine Datenstruktur, in der Einfügen in konstanter Zeit ebenso möglich ist wie die Rückgabe des minimalen Werts.

(a) Beschreiben Sie kurz eine mögliche Datenstruktur.

(b) Schreiben Sie Pseudocode für das Einfügen. Es genügt, den für die spätere Rückgabe des Minimums relevanten Teil zu betrachten.

(c) Schreiben Sie Pseudocode, der das Minimum zurückgibt.

(d) Beschreiben Sie kurz, was beim Entfernen eines Elements passieren muss. Hat dies Auswirkung auf die Laufzeit des Löschens?

(3) Für die folgenden Anwendungsfälle soll jeweils eine geeignete Datenstruktur ausgewählt werden. Geben Sie jeweils eine passende Datenstruktur an und begründen Sie Ihre Wahl:

(a) In einer Musiksammlung sollen häufig neue Musikstücke hinzugefügt, gelöscht und gesucht werden. Über die zukünftige Größe der Sammlung kann beim Anlegen noch keine Aussage gemacht werden.

(b) An einen Datenbankserver werden zu manchen Zeitpunkten so viele Anfragen geschickt, dass er sie nicht sofort bearbeiten kann. Daher soll er die Möglichkeit bekommen, die Anfragen zwischenspeichern zu können, bis er wieder genügend freie Ressourcen besitzt.

(c) Ein Prozess-Scheduler in einem Betriebssystem arbeitet mit unterschiedlich hohen Prioritäten. Bei jedem Aufruf soll jeweils der Prozess mit der höchsten Priorität ausgeführt werden, wobei die Auswahlgeschwindigkeit für die Leistungsfähigkeit des Betriebssystems eine entscheidende Rolle spielt.

(4) Zeigen Sie, wie eine Warteschlange mit einer **einfach verketteten** Liste sowie zwei Zeigern K (ältestes Element) und E (neuestes Element) implementiert werden kann. Hierbei sollen sowohl ENQUEUE(X) als auch DEQUEUE() die Laufzeit O(1) besitzen.
Gehen Sie dafür folgendermaßen vor:

- Stellen Sie mit einer Skizze dar, wie die beiden Zeiger in der Liste positioniert werden müssen.
- Geben Sie den Pseudocode für beide Warteschlangenoperationen an. Stellen Sie dabei sicher, dass Ihre Implementierung auch korrekt mit einer leeren Warteschlange umgehen kann.

Graphen: Aufgabe 6

(1) Zeigen oder widerlegen Sie: Ein kürzester-Wege-Baum ist auch ein minimaler Spannbaum.

(2) (a) Zeigen Sie: Dijkstra liefert bei negativen Kanten in gerichteten Graphen auch ohne negative Kreise im Allgemeinen ein falsches Ergebnis.

(b) Geben Sie einen stark zusammenhängenden, gerichteten Graphen mit mindestens einer negativen Kante an, so dass Dijkstra für mindestens einen Startknoten ein korrektes Ergebnis liefert.

(c) Warum funktioniert Dijkstra nicht auf zusammenhängenden, ungerichteten Graphen mit mindestens einer negativen Kante?

(3) Ein Graph G = (V,E) sei durch die folgende Adjazenzfelddarstellung gegeben:

V :=

1	3	4	6	9	11

E :=

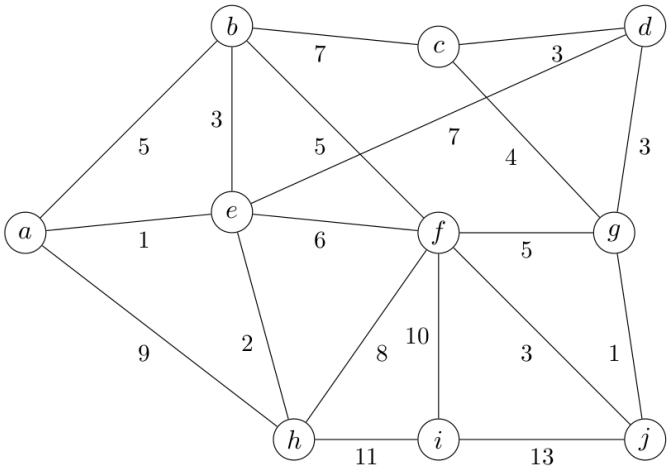
2	3	4	2	5	2	3	6	1	6	5

(a) Zeichnen Sie den Graphen.

(b) Geben Sie den Graphen in Adjazenzlistendarstellung an.

(c) Welche dritte Möglichkeit zur Darstellung von Graphen wurde in der Vorlesung vorgestellt? Beschreiben Sie sie kurz und nennen Sie sowohl einen Vor- als auch einen Nachteil dieser Methode.

(4) Wenden Sie die Algorithmen von Prim und Kruskal auf den folgenden Graphen an. Geben Sie dabei jeweils die Reihenfolge der betrachteten Kanten bzw. Knoten an.



Dynamische Programmierung: Aufgabe 7

(1) Was ist die grundlegende Idee hinter der Methode der dynamischen Programmierung?

(2) Die Fakultätsfunktion $n!$ lässt sich folgendermaßen rekursiv definieren:
$0! = 1$
$n! = n \cdot (n-1)!$ für $n \geq 1$

(a) Geben Sie ein Programm in Pseudocode an, welches $n!$ mittels dynamischer Programmierung und Bottom-Up-Ansatz berechnet.

(b) Das Programm soll nun so modifiziert werden, dass es nach einmaligem Berechnen von $n!$ jeden Aufruf $k!$ mit $k \leq n$ in $O(1)$ bearbeiten kann. Beschreiben Sie eine Möglichkeit dafür.

(3) Der Binomialkoeffizient $\binom{n}{k}$ kann folgendermaßen rekursiv berechnet werden:

$$\binom{n}{k} = \begin{cases} 0, & \text{falls } k > n; \\ 1, & \text{falls } k = 0 \text{ oder } n = k; \\ \binom{n-1}{k-1} + \binom{n-1}{k}, & \text{sonst.} \end{cases}$$

Geben Sie ein Programm in Pseudocode an, welches $\binom{n}{k}$ mittels dynamischer Programmierung und Bottom-Up-Ansatz berechnet. Hinweis: Verwenden Sie eine Matrix (d.h. ein 2-dimensionales Array), um die Lösungen der Teilprobleme zu speichern.