13. Übungsblatt zu Grundzüge von Algorithmen und Datenstrukturen, WS 15/16

Prof. Markus Bläser
http://www-cc.cs.uni-saarland.de/course/50/

Abgabe: nie

- Dieses Blatt enthält einige Beispiel-Klausuraufgaben.
- Sollten Ihnen noch Punkte fehlen, so können Sie durch Bearbeiten des Blattes Zusatzpunkte erwerben. Bitte klären Sie dies mit Ihrem Übungsleiter.
  (Die Punkte, die an den Aufgaben angegeben sind, sind den Klausuren entnommen und entsprechen den relativen Schwierigkeiten. Für die Klausurzulassung können Sie jeweils maximal 4 Punkte pro richtig bearbeitete Aufgabe erhalten.)
- Der Umfang der Klausur wird geringer sein als dieser Zettel.

Aufgabe 13.1 Betrachten Sie folgenden rekursiven Algorithmus:
Input: Array a, Grenzen i und j, $i \leq j$
Output: $a[i..j]$ sortiert

```
1: if a[i] > a[j] then
2:     h := a[i]; a[i] := a[j]; a[j] := h
3: if i + 1 ≥ j then
4:     return
5: k := ⌊(j − i + 1)/3⌋
6: Crazy-sort(a, i, j − k)
7: swap(a[i + k .. j − k], a[j − k + 1 .. j])
8: Crazy-sort(a, i, i + 2k − 1)
9: Crazy-sort(a, i + k, j)
```

swap$(a[i+k \,..\, j-k], a[j-k+1 \,..\, j])$ vertauscht im Array a die beiden angegebenen Bereiche und braucht dafür $\Theta(\max(j-i-2k, k))$ Schritte.

(a) *(5 Punkte)* Zeigen Sie, dass Crazy-Sort das Array $a[i..j]$ sortiert.

(b) *(5 Punkte)* Stellen Sie eine Rekursionsgleichung für die Laufzeit von Crazy-Sort auf und bestimmen Sie deren asymptotisches Wachstum.

Aufgabe 13.2 *(12 Punkte)* Gegeben sei der in der Abbildung gezeichnete AVL-Baum. Die virtuellen Blätter sind in dieser Abbildung nicht gezeichnet und müssen von Ihnen ebenfalls nicht gezeichnet werden. Führen Sie die unten angegebenen Operationen nacheinander auf dem Baum aus. Beachten Sie dabei die nachfolgenden zwei Punkte:

- Geben Sie jeweils den Baum nach dem Einfügen/Löschen an und geben Sie die Werte fü r die Balancen an.

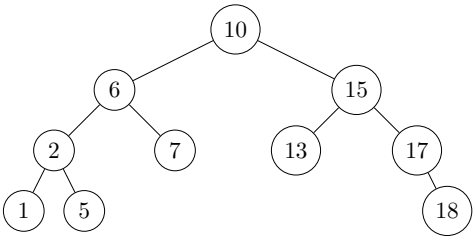
- Sollten Sie Einfach- oder Doppel-Rotationen brauchen, so geben Sie den Baum zusätzlich nach der Rotation mit den jeweiligen Balancen an.

a) Fügen Sie 4 ein.

b) Fügen Sie 19 ein.

c) Löschen Sie 6.

d) Löschen Sie 18.



Aufgabe 13.3 a) *(4 Punkte)* Sei T ein heap-geordneter Binomial-Baum mit 2^i Elementen und sei $m \leq 2^i$. Zeigen Sie: T lässt sich so in zwei Mengen von heap-geordneten Binomial-Bäumen zerlegen, dass die eine Menge genau m Elemente hat (und die andere dann natürlich $2^i - m$) und in den beiden Mengen zusammen jeweils für jedes j maximal zwei Bäume mit 2^j Elementen vorhanden sind.

b) *(3 Punkte)* Geben Sie einen Algorithmus an, der, gegeben einen ein heap-geordneten Binomial-Baum T mit $n = 2^i$ Elementen und $m \leq 2^i$, zwei *Binomial-Heaps* H_1 und H_2 ausgibt, so dass H_1 und H_2 die gleichen Elemente wie T speichern und H_1 m Elemente enthält. Die Laufzeit Ihres Algorithmus soll $O(\log n)$ sein. Erläutern Sie Ihren Algorithmus und begründen Sie die Laufzeit.

c) *(4 Punkte)* Entwerfen Sie eine Prozedur Binomial-Heap-Split(H, r), die einen Binomial Heap H mit n Elementen in Heaps H_1, H_2 aufteilt, so dass H_1 r Elemente enthält und H_2 $n - r$ Elemente enthält. Die Laufzeit soll $O(\log n)$ betragen. Erläutern Sie Ihren Algorithmus und begründen Sie die Laufzeit. Sie können annehmen, dass $n \geq r \geq 0$ ist.

Aufgabe 13.4 Das gewichtete Intervall-Scheduling-Problem ist wie folgt: Gegeben n Intervalle $I_1 = [s_1, f_1), \ldots, I_n = [s_n, f_n)$ mit Gewichten $w_i \in \mathbb{N}$ und $s_i, f_i \in \mathbb{N}$. Für jedes Intervall gilt, dass $s_i < f_i$. Wir nennen zwei Intervalle I_1, I_2 kompatibel wenn $I_1 \cap I_2 = \emptyset$ ist. Wir nennen eine Indexmenge $S \subseteq \{1, \ldots, n\}$ kompatibel, falls für alle $i, j \in S$ gilt: Wenn $i \neq j$ dann ist I_i kompatibel mit I_j. Wir nennen das Gewicht einer Indexmenge S, die Summe über das Gewicht der Intervalle, d.h. $\sum_{i \in S} w_i$. Wir wollen nun eine kompatible Indexmenge S mit maximalem Gewicht finden. Wir nennen solch eine Lösung optimal.

a) *(4 Punkte)* Nehmen Sie hier an, dass die Elemente aufsteigend nach f_i sortiert sind. Angenommen Sie haben eine Funktion $p(j)$ gegeben, die das größte $i < j$ angibt, für das $f_i \leq s_j$. Sei Opt(j) der Wert eines optimalen Schedules für die Intervalle $I_1, \ldots, I_j$.

Drücken Sie Opt(j) durch Opt(k) aus für $k < j$. Beweisen Sie ihre Gleichung.

b) *(4 Punkte)* Benutzen Sie a) und $\mathrm{Opt}(0) = 0$, um einen Algorithmus für das Gewicht des gewichteten Intervall-Scheduling-Problems mittels Dynamischer Programmierung zu konstruieren. Beweisen Sie die Korrektheit und zeigen Sie die Laufzeit ihrer Implementation. Sie können $p(j)$ wieder als gegeben annehmen.

c) *(3 Punkte)* Wie können Sie zusätzlich ein optimales Schedule ausrechnen?

Aufgabe 13.5 (a) *(3 Punkte)* Sei $G = (V, E)$ ein ungerichteter zusammenhängender Graph und (V_1, V_2) ein Schnitt von G. Zeigen Sie: jeder minimale Spannbaum T von G enthält eine Kante e, die von V_1 nach V_2 führt und unter allen diesen Kanten minimales Gewicht hat.

Wir betrachten nun einen ungerichteten, gewichteten Graph $G = (V, E)$ mit Gewichtsfunktion $w : E \to \mathbb{N}$, einen minimalen Spannbaum T von G und eine Kante e aus T.

(b) *(1 Punkt)* Geben Sie ein Beispiel an, in dem eine Gewichtsänderung von e dafür sorgt, dass T nicht mehr minimaler Spannbaum ist.
Hinweis: Es gibt eine Lösung G mit weniger als fünf Knoten.

(c) *(1 Punkt)* Geben Sie ein Beispiel an, in dem T trotz jeder möglichen Gewichtsänderung von e ein minimaler Spannbaum bleibt.
Hinweis: Es gibt eine Lösung G mit weniger als fünf Knoten.

(d) *(7 Punkte)* Geben Sie einen Algorithmus (in Pseudopseudocode) an, der bestimmt, ob eine Gewichtsänderung von e dafür sorgen kann, dass T kein minimaler Spannbaum mehr ist, und gegebenenfalls auch bestimmt, ab welcher Gewichtsänderung das der Fall ist. Begründen Sie, warum Ihr Algorithmus korrekt ist.