Datenstrukturen (DS)

Sommersemester 2017

Prof Dr. Martin Hoefer
Niklas Hahn

GOETHE UNIVERSITÄT
FRANKFURT AM MAIN

Institut für Informatik
Algorithmen und Komplexität

Übung 5

Ausgabe: 13.06.2017
Abgabe: 27.06.2017

Aufgabe 5.1. *Binäre Suchbäume vs Heaps* (3+3 = 6 Punkte)

In der Vorlesung haben Sie binäre Suchbäume mit den Operationen `insert(x)` und `remove(x)` sowie Heaps mit den Operationen `insert(x)` und `remove(wo)` kennen gelernt. Gegeben sind die folgenden Operationen, die auf einen anfänglich leeren binären Suchbaum bzw. Max-Heap angewandt werden: `insert(13), insert(2), insert(7), insert(15),`
`remove(2), insert(4), insert(6), remove(13)`

a) Geben Sie für die obige Folge von Operationen jeden Zwischenschritt sowie den resultierenden Binären Suchbaum in graphischer Darstellung an.

b) Geben Sie für die obige Folge von Operationen jeden Zwischenschritt sowie den resultierenden Heap in graphischer Darstellung an. Die Operation `remove(x)` soll hierbei den Schlüssel `x` entfernen.

Eine weitere Begründung ist nicht gefordert.

Aufgabe 5.2. *k-näre Heaps* (2+3+4+3 = 12 Punkte)

In der Vorlesung haben Sie sich mit binären Heaps beschäftigt. Betrachten Sie nun *k-näre Heaps* für $k \geq 3$.

Ein k-närer Heap ist die natürliche Erweiterung von binären Heaps auf mehr als 2 Kinder pro Knoten. Die Heap-Ordnung gilt wie vorher, nur die Heap-Struktur ändert sich entsprechend. Wenn der Heap Tiefe t hat, dann hat jeder Knoten der Tiefe höchstens $t-2$ genau k Kinder. Auf Tiefe $t-1$ gibt es einen Knoten v mit höchstens k Kindern. Alle Knoten auf Tiefe $t-1$ links von v haben genau k Kinder. Alle Knoten auf Tiefe $t-1$ rechts von v sind Blätter.

a) Zeigen Sie, dass die Tiefe des zugehörigen Baums mit n Knoten $\Theta(\log_k(n))$ beträgt.

b) Welche Position im Array haben die Kinder von Knoten i für $1 \leq i \leq n$ (sofern sie existieren)? Welche Position im Array hat der Elternknoten von Knoten j für $2 \leq j \leq n$?

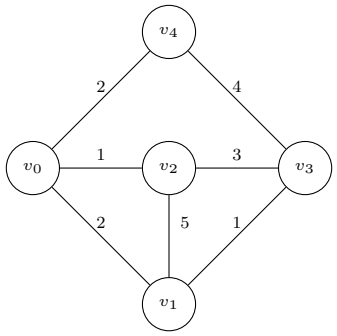
c) Wie muss man `Repair_down()` verändern, damit die Prozedur für einen k-nären Heap funktioniert? Beschreiben Sie die geänderte Prozedur in **Pseudocode** und analysieren Sie die Laufzeit Ihres Verfahrens asymptotisch in k und n.

d) Wie muss man `Repair_up()` verändern, damit die Prozedur für einen k-nären Heap funktioniert? Beschreiben Sie die geänderte Prozedur in **Pseudocode** und analysieren Sie die Laufzeit Ihres Verfahrens asymptotisch in k und n.

Bitte wenden!

Aufgabe 5.3. *Dijkstra, Prim und Kruskal* (3+3+3+3 = 12 Punkte)

Gegeben sei folgender ungerichteter, gewichteter Graph $G = (V, E)$:



a) Bestimmen Sie mithilfe von Dijkstras Algorithmus die kürzesten Wege von Knoten v_1 zu allen anderen Knoten. Hierbei sind die eingetragenen Kantengewichte für beide Richtungen der Kante gültig. Geben Sie nach Hinzunahme eines Knotens in die Menge S die aktualisierten Distanzwerte aller Knoten aus $V \setminus S$ sowie den *Baum der kürzesten Wege* an. Bei gleicher Distanz wird der Knoten mit kleinerem Index zuerst bearbeitet.

b) Bestimmen Sie mithilfe von Prims Algorithmus den minimalen Spannbaum von G und zeichnen Sie diesen. Geben Sie dabei in jedem Schritt den Zustand des Heaps an und notieren Sie, welche *kürzesten S-kreuzenden Kanten* zur Auswahl stehen. Starten Sie wieder bei Knoten v_1.

c) Bestimmen Sie mithilfe von Kruskals Algorithmus den minimalen Spannbaum von G. Bei Kanten mit gleichem Gewicht erfolgt die Sortierung gemäß der Knotenindizes, wobei die Kante, die den kleinsten Index enthält, stets zuerst gewählt werden soll. Skizzieren Sie für jeden Schritt, in graphischer Darstellung, den Zustand der *Union-Find*-Datenstruktur. Bei Vereinigung zweier Bäume gleicher Größe soll dabei stets der Knoten mit dem kleineren Index die Wurzel des neuen Baumes sein.

d) Zeigen oder widerlegen Sie: In jedem gewichteten ungerichteten Graphen sind minimaler Spannbaum und Baum der kürzesten Wege gleich.

Aufgabe 5.4. *Bin-Packing* (6 Punkte)

Wir betrachten das folgende Problem:
Gegeben n Objekte mit Gewichten $g_1, ..., g_n \in (0, 1]$, die in möglichst wenig Behälter („Bins“) verteilt werden sollen, wobei alle Behälter maximale Kapazität 1 besitzen.
Ein möglicher Ansatz ist es, die Objekte der Reihe nach (also 1,2,3,...,n) jeweils in den Behälter mit der größten Restkapazität zu legen. Wenn das aktuelle Objekt in keinen der bereits angefangenen Behälter hineinpasst, wird ein neuer Behälter dazugenommen.

Beschreiben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(n \log n)$, der n Objekte nach obigem Verfahren verteilt.

Beschreiben Sie zuerst ihre Idee, geben dann den Algorithmus an und analysieren Sie seine Laufzeit und begründen seine Korrektheit.