5. Arbeitsblatt Priority Queues – Teil 2

Aufgabe 7 (Heap bauen in linearer Zeit)

Der Aufwand vom Algorithmus von Floyd ist O(n). Um dies zu zeigen, zählen wir die Anzahl Vertauschungen, die während des Algorithmus maximal gemacht werden. Weil so ein Tausch in konstanter Zeit möglich ist, ergibt sich daraus die Laufzeit. Zentral für unsere Analyse ist folgende Beobachtung:

- Ein siftDown in einem Teilbaum der Höhe h macht höchstens h-1 Vertauschungen, weil auf einem Pfad von der Wurzel in ein Blatt maximal h-1 andere Elemente liegen.

Die Anzahl Vertauschungen kann also aus der Summe der Höhen aller Teilbäume berechnet werden. Eine solche Summe lässt sich am besten rekursiv aufschreiben. Der Einfachheit halber nehmen wir für unsere Analyse an, der Baum sei vollständig. Ist er es nicht, fallen ein paar Vertauschungen weg. Wir beweisen also eine obere Schranke.

Sei F(h) die maximale Anzahl Vertauschungen in einem Binärbaum der Höhe h. Dann gilt

$$\mathrm{F}(h) = \begin{cases} 0 & falls\ \ h = 1 \\ (h-1) + 2 \cdot F(h-1) & falls\ \ h > 1 \end{cases}$$

a) Erklären Sie die obige Rekursionsgleichung in Worten.

Die obige Rekursionsformel löst sich auf zu

$$F(h) = 2^h - h - 1.$$

Nun müssen wir von der Höhe des Heaps zu den Anzahl Elementen darin kommen. Wir wissen, dass die Höhe eines Heaps mit n Knoten $h = 1 + \lfloor \log_2 n \rfloor$ ist. Ersetzen wir h durch diesen Term erhalten wir

$$\begin{array}{rcl} 2^h - h - 1 & = & 2^{1+\lfloor \log_2 n \rfloor} - (1 + \lfloor \log_2 n \rfloor) - 1 \\ & \leq & 2 \cdot n \quad - \log_2 n \quad - 1 \\ & = & O(n) \end{array}$$

Der Algorithmus von Floyd schafft es tatsächlich, in linearer Zeit aus einem ungeordneten Array einen Heap zu bauen. Wenn wir uns überlegen, dass bereits die Überprüfung der Ordnungseigenschaft O(n) Zeit braucht, können wir daraus schliessen, dass es keinen schnelleren Algorithmus geben kann.

b) Zeigen Sie mittels Teleskopieren, dass sich die Rekursionsgleichung zu $\mathbf{F(h) = 2^h - h - 1}$ auflöst. Benützen Sie die folgenden beiden geometrischen Reihen, um Summen aufzulösen:

$$\sum_{i=0}^{k} c^i = \frac{c^{k+1} - 1}{c - 1} \qquad \sum_{i=0}^{k} i c^i = \frac{k c^{k+2} - (k+1) c^{k+1} + c}{(c-1)^2}$$

Aufgabe 8 (Heapsort von Hand)

Führen Sie Heapsort (in-place) von Hand auf folgendem Array durch. Zeichnen Sie den Array nach jedem siftDown.