Prof. Dr. A. Goerdt
Julian Pape-Lange

TU Chemnitz
Wintersemester 2021/2022
01.12.2021

Theoretische Informatik I

7. Übung

Abgabe: Lösen Sie Aufgabe **3** handschriftlich. Ihre Lösungen geben Sie bitte entweder

- bis zum 08.12.2021 um 13:00 Uhr per Mail
  an `julian.pape-lange@informatik.tu-chemnitz.de`
  mit *Betreff:* `TI1 Hausaufgaben` oder
- bis zum 08.12.2021 um 13:00 Uhr im Briefkasten der Professur Theoretische Informatik (vor Raum A10.266.4)

ab.

1. Aufgabe:
Implementieren Sie den Heap mit Knotenindex aus der Vorlesung, wie er für Dijkstra gebraucht wird.
Es werden also drei Arrays mit einer Länge/Kapazität von $|V|$ gebraucht. Ein Array für den Heap, ein Array für die Schlüsselwerte und ein Array für den Index.
Bei dem Dijkstra-Algorithmus muss jeweils nur der Wurzelknoten gelöscht werden. Überlegen Sie sich trotzdem, wie Sie auch andere Knoten im Heap löschen können

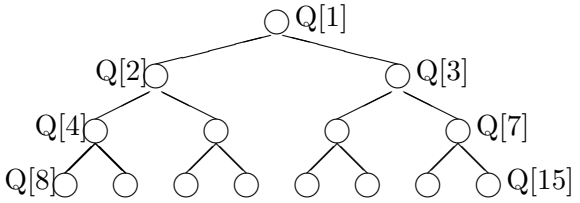
2. Aufgabe:
Beim Löschen der Wurzel im Heap muss erst der Wurzelknoten mit dem letzten Element getauscht werden und dann muss das (jetzt oben stehende) letzte Element wieder heruntersickern.
Überlegen Sie sich, warum man nicht direkt die Wurzel löschen kann und die entstehenden Leerstellen durch „Hochrücken“ von Kindknoten füllen kann.

3. Aufgabe: (6+4)
Zeigen Sie, dass

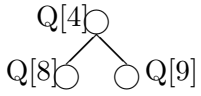
(a) der folgende Algorithmus die Laufzeit $O(n)$ hat und

(b) der durch den Algorithmus erzeugte „Heap“ tatsächlich alle Bedingungen des Heaps erfüllt.

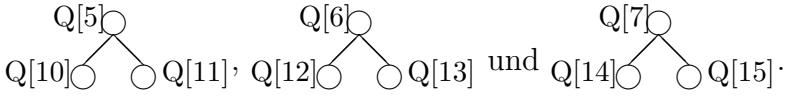
Gegeben sei ein Feld $Q[1..n]$ von Elementen. Wir wollen aus Q einen *Heap* aufbauen. Wir nehmen $n = 2^k - 1$ an und betrachten Q als Baum (z. B. $k = 4$):



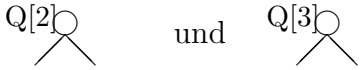
Der Heap wird nun *bottom-up* aufgebaut. Das heißt, zuerst wird in dem Teil



die Heap-Eigenschaft hergestellt, indem $Q[4]$ an die richtige Stelle sickert. Genauso verfahren wir mit den Teilen



Danach lassen wir $Q[2]$ bzw. $Q[3]$ an die richtige Stelle sickern, um in den Teilbäumen



die Heapeigenschaft herzustellen. Schließlich entsteht durch Sickern von $Q[1]$ ein korrekter Heap.

Hinweis: Überprüfen Sie, wie oft ein Knoten aus Ebene i maximal sickern kann, bis er an der richtigen Stelle steht. Summieren Sie diesen Wert über alle Knoten. Die entstehende Summenformel kann auf eine geometrische Reihe zurückgeführt werden, wenn man die Ungleichung $i < 1,5^i$ benutzt.

4. Aufgabe:
In der Vorlesung haben wir einen Algorithmus gesehen, der von allen Knoten zu allen Knoten den kürzesten Weg in $\mathcal{O}(n^4)$ berechnet.
Der Algorithmus berechnet für jede Zahl i mit $0 \leq i \leq n-2$ alle kürzesten Wege mit maximal i Zwischenknoten.
Passen Sie den Algorithmus an, sodass nicht alle i gebraucht werden und der Algorithmus nur noch $\mathcal{O}(n^3 \log n)$ Zeit braucht.