Algorithmen und Datenstrukturen
WS 2025
von Luxburg/Bordt

Präsenzübung 3

Besprechung: 27.10.25 - 31.10.25

Aufgabe 1: Heaps (mündlich, keine Punkte)

Betrachten Sie die folgenden drei in Array-Schreibweise gegebenen Binärbäume (diese Reihenfolge der Indizierung heißt auch "Level-Order"):

$$T_1 = \boxed{6}\boxed{3}\boxed{9}\boxed{2}\boxed{1}\boxed{7} \quad T_2 = \boxed{9}\boxed{7}\boxed{5}\boxed{8}\boxed{3}\boxed{6}\boxed{2} \quad T_3 = \boxed{9}\boxed{6}\boxed{8}\boxed{3}\boxed{5}\boxed{1}\boxed{4}\boxed{2}$$

Nur einer davon (T_a) erfüllt die Max-Heap-Eigenschaft. Ein zweiter (T_b) verletzt die Max-Heap-Eigenschaft an genau einer Stelle. Der dritte (T_c) kann nur durch mehrere Operationen in einen gültigen Max-Heap umgewandelt werden.

a) Bestimmen Sie $a, b, c \in \{1, 2, 3\}$.

b) Führen Sie auf T_b die HEAPIFY-Operation an der Fehlstelle aus. Geben Sie Zwischenschritte und das Resultat in Level-Order an.

c) Führen Sie auf T_a nacheinander die folgenden Operationen aus: Fügen Sie eine 7 hinzu mit INSERT(7), extrahieren Sie das Maximum mit EXTRACTMAX, und verringern Sie die 8 auf 0 mit DECREASE($8 \mapsto 0$). Geben Sie nach jeder Operation das Ergebnis in Level-Order an.

d) Erstellen Sie aus T_c einen Max-Heap mittels der Operation BUILDMAXHEAP. Geben Sie das Resultat in Level-Order an.

e) In BUILDMAXHEAP wird die HEAPIFY-Operation auf dem zugrundeliegenden Array von rechts nach links gehend angewandt. Kann man stattdessen auch von links nach rechts gehen (d.h. man beginnt an der Wurzel und geht dann in jedem darauffolgenden Level von links nach rechts)? Beweisen oder widerlegen Sie!

f) Erstellen Sie aus den originalen Array-Daten von T_c einen *ternären* Heap, indem sie ganz analog zum binären Fall die Einträge Level für Level von links nach rechts in einen ternären Baum schreiben. Erfüllt das Ergebnis die Max-Heap-Eigenschaft?

Aufgabe 2: Stacks (mündlich, keine Punkte)

Angenommen Sie möchten eine Stack-Datenstruktur mit den üblichen Funktionen PUSH($\cdot$) und POP($\cdot$) bereitstellen. Dazu steht Ihnen genau eine einfach verkettete Liste zur Verfügung (also keine Arrays oder anderes).

1. Wie kann die Stack-Datenstruktur mittels der einfach verketteten Liste implementiert werden? Beschreiben Sie eine Implementierung in Worten und in Pseudocode.
2. Was können Sie über die worst-case-Laufzeit für eine PUSH($\cdot$)- bzw. POP($\cdot$)-Operation Ihrer Implementierung sagen?
3. Nehmen Sie nun an, dass Ihnen nicht die einfach verkettete Liste zur Verfügung steht, sondern genau ein Array. Wie kann die Stack-Datenstruktur mittels des Arrays implementiert werden? Beschreiben Sie eine Implementierung in Worten und in Pseudocode. *Hinweis:* Sie dürfen zwar keine zusätlichen Datenstrukturen benutzen, Pointer aber schon.
4. Was können Sie über die worst-case-Laufzeit für eine PUSH($\cdot$)- bzw. POP($\cdot$)-Operation der Array-Implementierung sagen?
5. Wie unterscheiden sich die beiden Implementierungen des Stacks? Diskutieren Sie Vor- und Nachteile.
6. Ist es möglich drei verschiedene Stacks mit genau einem Array zu implementieren?

Aufgabe 3: Queues (mündlich, keine Punkte)

Angenommen Sie möchten eine Queue-Datenstruktur mit den üblichen Funktionen ENQUEUE(·) und DEQUEUE(·) bereitstellen, wozu Ihnen "intern" genau eine doppelt verkettete Liste zur verfügung steht (aber keine Arrays, Stacks oder anderes).

1. Wie kann die Queue-Datenstruktur mittels der doppelt verketteten Liste implementiert werden? Beschreiben Sie eine Implementierung in Worten und in Pseudocode.

2. Was können Sie über die worst-case-Laufzeit für eine PUSH(·)- bzw. POP(·)-Operation Ihrer Implementierung sagen?

3. Kann die Queue-Datentstruktur auch mit genau einer einfach verketteten Liste implementiert werden? Beschreiben Sie eine Implementierung in Worten und in Pseudocode. *Hinweis:* Sie dürfen einen zusätzlichen Pointer verwenden.