9,5 / 10 P.

Exercise Sheet 5 - Gruppe 4

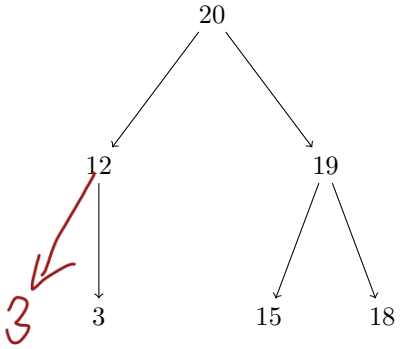
Jeudyl Robles Pidiache

December 3, 2023

Exercise 1: 4 / 4P.

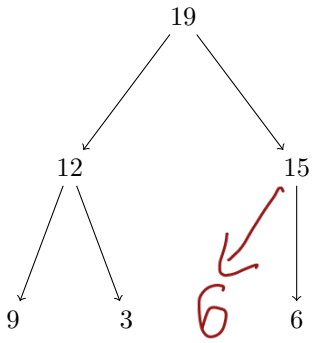
a) 1 / 1P.

A.)



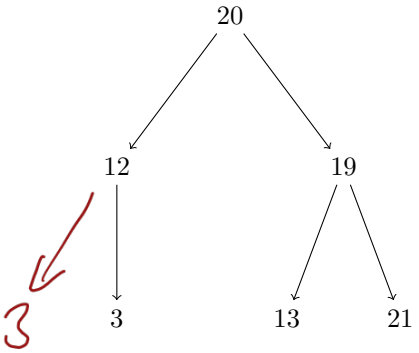
- Elternknot (12) hat nur als Kind Knot (3), d.h es gibt eine Lücke (Loch). nicht links-vollständig.
- Fazit: Baum B ist kein Heap. ✓

B.)



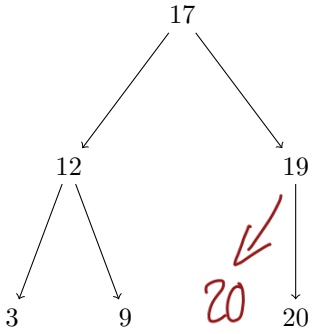
- Der Wurzelknoten (19) ist größer als beide Kinder (12 und 15). Der linke Kindknoten (12) ist größer als beide seine Kinder (9 und 3). Der rechte Kindknoten (15) ist größer als sein einziges Kind (6).
- Fazit: Baum B ist ein Heap.

C.)



- Nein, das rechte Kind der Elternknot (19) hat ein Kind (21), dass größer als (19) ist und Elternknot (12) hat nur (3) als kind, d.h es gibt eine Lücke (Loch).
- Fazit: Baum C ist kein Heap.

D.)



- Nein, der Elternteil (19) hat ein Kind (20), das größer als (19) ist und die Wurzel (17) hat ein Kind (19), das größer als (17) ist.
- Fazit: Baum D ist kein Heap.

b)

Die Array $A = [10, 5, 0, 3, 11, 7, 9, 8, 12, 4, 6]$ wurde mit dem Heapsort-Verfahren sortiert. Hier ist der Zustand des Arrays nach jedem Aufruf der `sink`-Funktion

und nach jedem Extraktionsschritt, wobei die Teilung zwischen dem Heap- und dem sortierten Teil des Arrays angezeigt wird:

1. Heap nach `sink(4)`: $[10, 5, 0, 3, 11, 7, 9, 8, 12, 4, 6]$
2. Heap nach `sink(3)`: $[10, 5, 0, 12, 11, 7, 9, 8, 3, 4, 6]$
3. Heap nach `sink(2)`: $[10, 5, 9, 12, 11, 7, 0, 8, 3, 4, 6]$
4. Heap nach `sink(1)`: $[10, 12, 9, 8, 11, 7, 0, 5, 3, 4, 6]$
5. Heap nach `sink(0)`: $[12, 11, 9, 8, 10, 7, 0, 5, 3, 4, 6]$

Nach dem Aufbau des Heaps:

6. Heap nach Extraktion 1: $[11, 10, 9, 8, 6, 7, 0, 5, 3, 4]$ — Sortiert: $[12]$
7. Heap nach Extraktion 2: $[10, 8, 9, 5, 6, 7, 0, 4, 3]$ — Sortiert: $[11, 12]$
8. Heap nach Extraktion 3: $[9, 8, 7, 5, 6, 3, 0, 4]$ — Sortiert: $[10, 11, 12]$
9. Heap nach Extraktion 4: $[8, 6, 7, 5, 4, 3, 0]$ — Sortiert: $[9, 10, 11, 12]$
10. Heap nach Extraktion 5: $[7, 6, 3, 5, 4, 0]$ — Sortiert: $[8, 9, 10, 11, 12]$
11. Heap nach Extraktion 6: $[6, 5, 3, 0, 4]$ — Sortiert: $[7, 8, 9, 10, 11, 12]$
12. Heap nach Extraktion 7: $[5, 4, 3, 0]$ — Sortiert: $[6, 7, 8, 9, 10, 11, 12]$
13. Heap nach Extraktion 8: $[4, 0, 3]$ — Sortiert: $[5, 6, 7, 8, 9, 10, 11, 12]$
14. Heap nach Extraktion 9: $[3, 0]$ — Sortiert: $[4, 5, 6, 7, 8, 9, 10, 11, 12]$
15. Heap nach Extraktion 10: $[0]$ — Sortiert: $[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$

Das endgültige Array / Ausgabe: $[0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$.

Exercise 2:

a)

$$100n + 105 \in O(n^2)$$
$$205n^2 \geq 100n + 105$$
$$n_0 = 1 \quad \text{und} \quad c = 205$$

b)

$$0,1n^2 - 5 \in \Omega(n)$$
$$0,5n \leq 0,1n^2 - 5$$
$$n_0 = 10 \quad \text{und} \quad c = 0,5$$

c)

$$O(n^3):$$

$$18n^3 \geq 6n^3 + 6n^2 + 6$$
$$n_0 = 1 \quad \text{und} \quad c_0 = 18$$

$$\Omega(n^3):$$

$$1n^3 \leq 6n^3 + 6n^2 + 6$$
$$n_0 = 1 \quad \text{und} \quad c_1 = 1$$

Exercise 3: 2,5/3P.

a) 1/1P.

$Anfangsarray : [1, 5, 8, 5, 3]$

1. $[1, 5, 5, 3, 8]$
2. $[1, 5, 3, 5, 8]$
3. $[1, 3, 5, 5, 8]$
4. $[1, 3, 5, 5, 8]$

b) 0,5/1P.

Das Ergebnis des Algorithmus ist das Sortieren des Eingabearrays in aufsteigender Reihenfolge. Das endgültige Ergebnis nach Abschluss des Algorithmus ist ein sortiertes Array.

nicht-absteigender

c) 1P.

Um den Algorithmus stabil zu machen, muss man die Vergleichsbedingung so anpassen, dass Elemente nicht vertauscht werden, wenn sie gleich sind. Ein stabiler Algorithmus bewahrt die ursprüngliche Reihenfolge gleichwertiger Elemente.

Im bereitgestellten Algorithmus erfolgt der Tausch, wenn die Bedingung $A[j-1] \geq A[j]$ wahr ist. Das bedeutet, dass selbst wenn $A[j-1]$ gleich $A[j]$ ist, ein Tausch durchgeführt wird, was die ursprüngliche Reihenfolge gleichwertiger Elemente stören kann. Um ihn stabil zu machen, muss man die Bedingung in streng "größer als" ändern, also $A[j-1] > A[j]$. Auf diese Weise wird

der Algorithmus nur Elemente tauschen, wenn das linke Element streng größer als das rechte Element ist, und die Reihenfolge gleichwertiger Elemente wird beibehalten.

Neuer Algorithmus:

```
1: for i ← n, n-1, ..., 2 do
2:     for j ← 2, 3, ..., i do
3:         if A[j-1] > A[j] then
4:             swap value in A[j-1] and A[j]
5:             print(A)
6:         end if
7:     end for
8: end for
```