## Aufgabe 1: Verschiedenes

a) Verkettete Liste und Array

b) nicht sortiert: O(n)
sortiert: O(log n)

c)

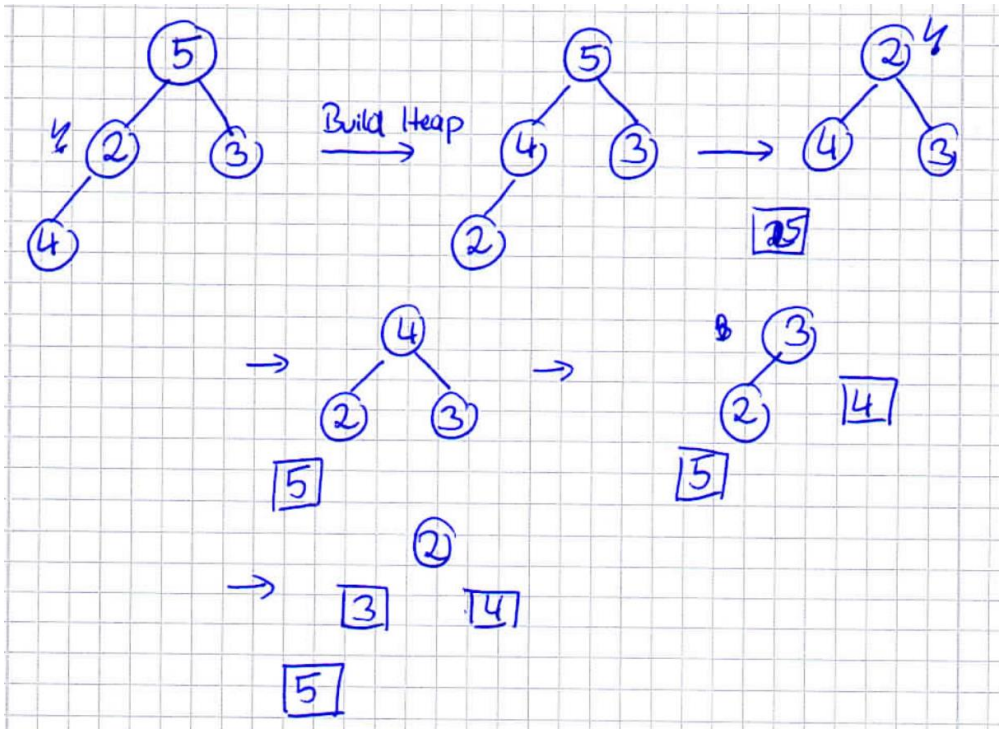
| | Unsortierte Liste | Sortierte Liste | MinHeap |
|---|---|---|---|
| `size` | O(1) | O(1) | O(1) |
| `insert` | O(1) | O(n) | O(log n) |
| `removeMin` | O(n) | O(1) | O(log n) |

d) Aussage ist richtig,z.B. Baumrepräsentation zeichnen.

## Aufgabe 2: Heapsort

a) Mergesort und Heapsort.

b)

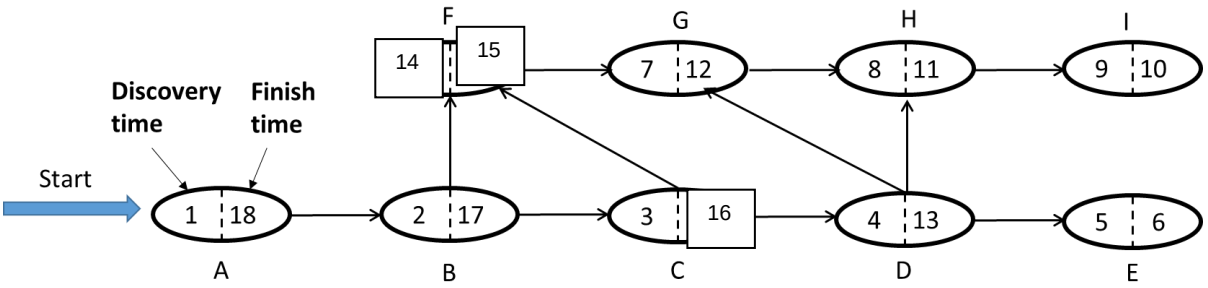



5 2 3 4 (Start)
5 4 3 2
2 4 3 5
4 2 3 5
3 2 4 5
2 3 4 5

c) Nein, er hat nicht Recht**.** Es wird immer zuerst ein Heap aufgebaut -> Beispiel

## Aufgabe 3: Graphen

a)




b) „Reihenfolge bei der Abhängigkeiten erfüllt sind“
Halbordnung zu Gesamtordnung.

c)

Beginn: Q={a, b, c, d, e, f, g}

Q ={b, c, d, e, f, g}

Q ={b, c, e, f, g}

Q = {b, e, f, g }

Q = {e, f, g}

Q = {e, g}

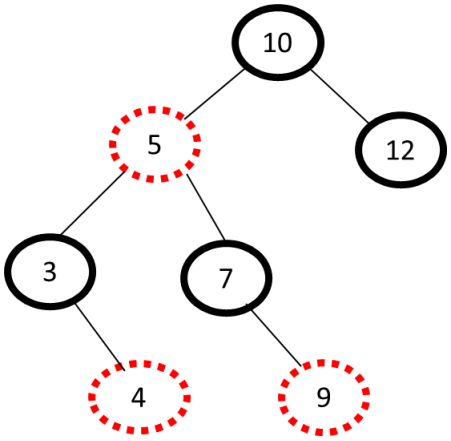
Q = {g}

## Aufgabe 4: Binäre Suchbäume und Rot-Schwarz-Bäume

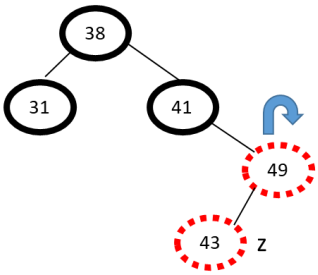
a)

(1) Es handelt sich um einen binären Suchbaum, da für ***jeden*** Knoten gilt: Die Schlüssel des linken Kindes sind kleiner als der Elternknoten und die Schlüssel des rechten Kindes sind größer als der Elternknoten.

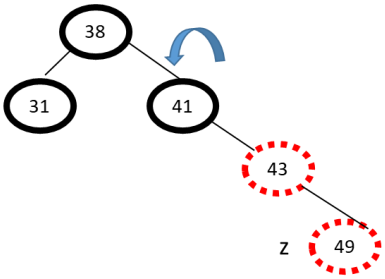
(2) ja .



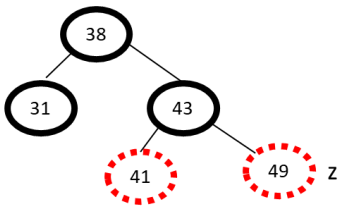

b)




Nach dem Einfügen, Knoten ist rot




Nach Rechtsrotation um 49, Z wechselt.




Nach Linksrotation um 41

c)

```
while (!q.isEmpty()) {
        BTNode n = (BTNode) q.dequeue(); // 1P (Typcast wird nicht gerechnet)
        System.out.print(n.item);
        if (n.left != null) {
            q.enqueue(n.left);
        }
        if (n.right != null) {
            q.enqueue(n.right);
        }
    }
```

## Aufgabe 5: Dynamische Programmierung

a)

```
    // using dynamic programming
private static long fibonacci(int n) {
    // TODO
    if (n < 2) {
        return 0;
    }
    if (n == 2) {
        return 1;
    }

    long prevprevprev = 0;
    long prevprev = 0;
    long prev = 1;
    for (int i = 3; i < n; i++) {
        long tempPrev = prev;
        prev = prev + prevprev + prevprevprev;
        prevprevprev = prevprev;
        prevprev = tempPrev;

    }
    return prev
}
```

b) Der maximale Erlös ist 27 bei einer Unterteilung von 2+2+1

c) 1. Änderung in Zeile 6: q mit p[j] initialisieren
2. Änderung in Zeile 4: Nicht auf –unendlich setzen
Eine alternative Lösung wäre bei Zeile 6 eine if-Abfrage: if (i==j)

```
BOTTOM-UP-CUT-ROD(p,n)
1       let r[0..n] be a new array
2       r[0] = 0
3       for j = 1 to n
4             q =p[j]
5             for i = 1 to j
6                   q = max(q, p[i] + r[j − i] − c)
7             r[j] = q
8       return r[n]
```