Aufgabe 04 - Bäume und Heaps

Heaps

In einem Heap werden Werte so eingefügt, dass das extrahieren des größten oder kleinsten Wertes ohne großen Aufwand möglich ist. Abhängig von der gewählten Datenstruktur (zum Beispiel *Binary-*, *Binomial-* oder *Fibonacci-Heap*) erfolgt die interne Organisation der Werte auf unterschiedliche Arten. Abbildung 1 zeigt einen Binary-Heap.

Erweitern Sie Ihre Implementierung eines binären Suchbaums aus *Übung 03* zu einer Heap-Datenstruktur. Es soll dabei die *min-Heap* Eigenschaft gelten und der kleinste Wert sehr effizient aus dem Heap extrahiert werden können. Implementieren Sie die Methode *insert* und *extractMin*. Testen Sie Ihre Implementierung mit unterschiedlichen Folgen von Zahlen und der Ausführung beider Operationen.

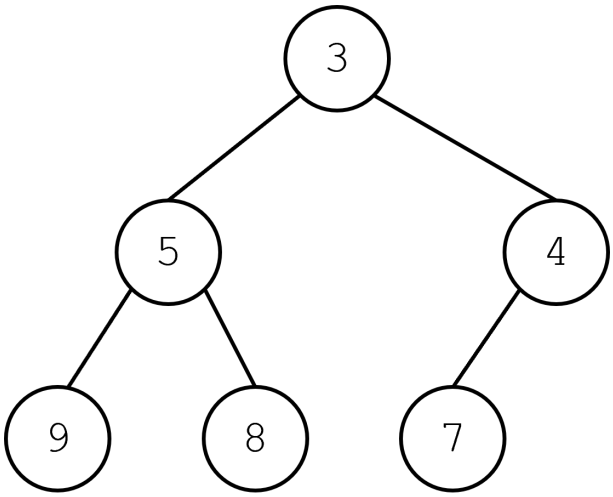

Abbildung 1: Beispiel für einen Binary-Heap

B-Bäume

In B-Bäumen werden Werte geordnet in einer Baumstruktur eingefügt. Durch einen höheren Verzweigungsgrad wächst der Baum schnell in die Breite und gleichzeitig die Höhe reduziert, was wiederum zu verbesserten Laufzeiten führen kann. Jeder B-Baum hat dabei einen maximalen Verzweigungsgrad *max(degree)*, der nicht überschritten werden darf.

Abbildung 2 zeigt einen bereits befüllten B-Baum. (Einführungsreihenfolge: 1, 5, 12, 15, 23, 24, 25, 10, 18, 34, 45, 64, 75, 80, 100, 125, 66, 90, 150)

Führen Sie auf dem in Abbildung 2 dargestellten B-Baum folgende Operationen (in der gegebenen Reihenfolge) durch:

1. Insert: 30, 35, 40, 85, 50, 20
2. Delete: 12

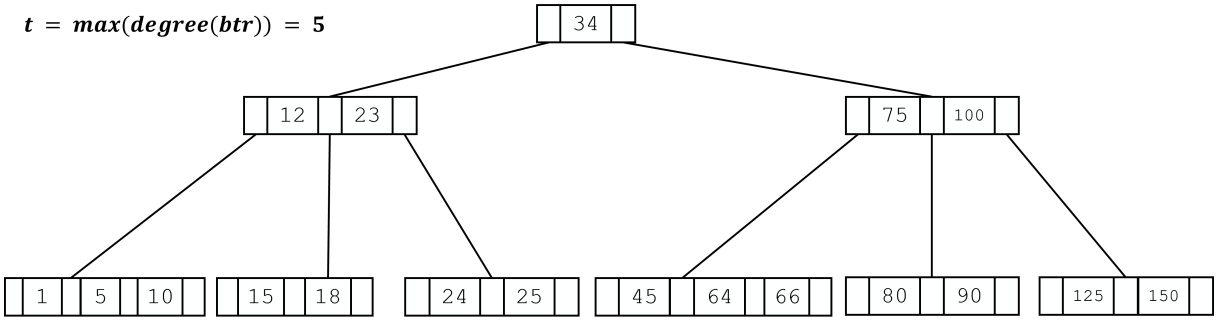

Abbildung 2: Bereits befüllter B-Baum mit max(degree)=5

3. Insert: 12, 77, 83
4. Delete: 30, 1
5. Insert: 105, 110, 115, 120, 130, 150
6. Delete: 34, 90
7. Insert: 90, 70

Recherchieren Sie wie das Löschen eines Keys in einem B-Tree durchgeführt wird. Überprüfen Sie Ihr Ergebnis mittels der Visualisierung auf https://www.cs.usfca.edu/~galles/visualization/BTree.html und geben Sie den finalen B-Baum an.