1. Übung Komplexe Softwaresysteme 2

Abgabe bis 4. März (vor der Übung) durch Hochladen in git. Gelöste Beispiele müssen hergezeigt und erklärt werden können. Aufgabe 10 bis 10. März. Aufgabe 4 kann in 2er-Teams gelöst werden, ansonst Einzelaufgaben.

1. Heap Pflicht

   (a) Zeichnen Sie den *(min)-heap*, der durch Einfügen von $5, 7, 1, 6$ entsteht.

   (b) Löschen Sie ein Element aus dem Heap (`pop()`) und zeichnen Sie den neuen Heap.

2. Wie funktioniert *heap sort*? Implementieren Sie *heap sort* mit Hilfe der bekannten Datenstrukturen. Argumentieren Sie warum der Algorithmus *worst-case* Komplexität $O(n \log n)$ besitzt.

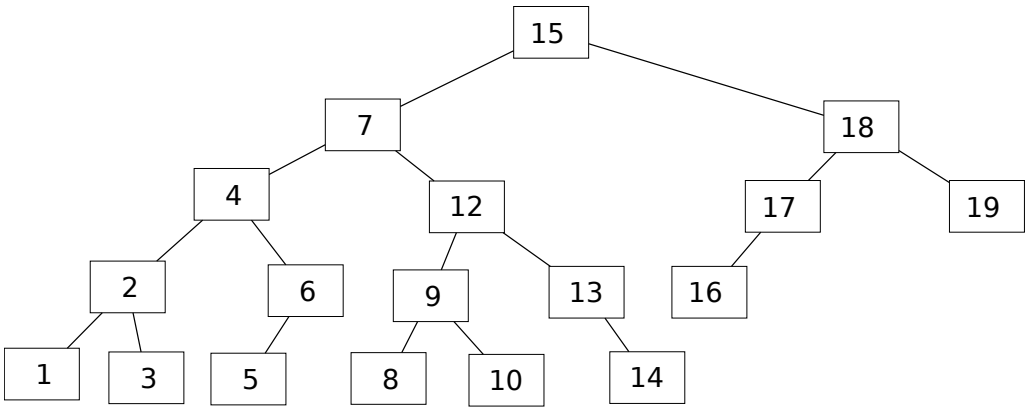
3. Wie kann man die Ausführungszeit eines Programms oder einer Funktion messen? Welche Alternativen gibt es? Was sind die Vor- und Nachteile von `time()`, `clock_gettime()`, `clock()`, `rdtsc`? Was ist processor time? Was ist wall clock time?

4. Schreiben Sie ein kleines Programm, mit welchem n Zufallszahlen in einen binären Suchbaum eingefügt werden können. Bestimmen Sie die Höhe des resultierenden Baumes (Grafik!). Wiederholen Sie diesen Test für $n = 10000, 20000, 30000, ...$ und vergleichen Sie die Höhe und die Laufzeit mit einem ausbalanzierten (AVL-)Baum (Grafik!).

5. Fügen Sie die Schlüssel $308, 52, 600, 987, 51, 4, 222, 412, 999, 577, 578$ in einen leeren AVL-Baum ein und zeichnen Sie den Baum nach jeder Einfügeoperation. Pflicht

6. Geben Sie eine Reihenfolge für das Einfügen der 10 Schüssel $0, 1, 2, ..., 8, 9$ in einen (zunächst leeren) AVL-Baum an, sodass keine Rotationen notwendig sind. (Es gibt mehrere Lösungen.)

7. Gegeben sei der folgende AVL-Baum.



   (a) Fügen Sie den Knoten 0 ein (ausgehend vom Baum oben) und balanzieren Sie den Baum.

   (b) Fügen Sie den Knoten 11 ein (ausgehend vom Baum oben) und balanzieren Sie den Baum.

8. Der `AvlTree` aus der Vorlesung hat ein Probleme. Was ist die Komplexität? Wie beheben Sie das Problem (Idee)?

9. AVL-Baum Pflicht

   (a) Welche wesentliche Eigenschaft besitzt ein *AVL-Baum*?

   (b) Zeichnen Sie den *AVL-Baum*, der sich durch Einfügen von $9, 13, 5, 7, 1, 8$ ergibt. Geben Sie die Gewichte aller Knoten vor und nach eventueller Korrekturoperationen an.

10. Studieren Sie die papers zu *Skiplists* und *Patricia* und beantworten Sie folgende Fragen: Pflicht: Personen mit Vornamen mit gerader Buchstabenanzahl machen (a), ansonst (b)

    (a) Skip list

        i. Zeichnen Sie die *skip list*, die durch Einfügen der Zahlen $11, 3, 7, 15, 13$ entsteht. Verwenden Sie in der Reihenfolge des Einfügens die (zufälligen) Level $2, 1, 3, 1, 3$. MaxLevel ist 3.

        ii. Welche und wie viele Vergleiche sind für die Suche des Wertes 10 in obiger *skip list* erforderlich? Illustrieren Sie die Suche!

        iii. Welchen Einfluss hat der Wert p auf den Algorithmus? In welchen Teil des Algorithmus findet der Wert p Eingang? Welche Werte für p schlägt der Author vor?

    (b) *Patricia*

        i. Wie kann die Suche in einem *Patricia*-Baum implementiert werden? Erklären Sie Ihren (Pseudo-)Code! Welche Daten werden für jeden Knoten gespeichert?

        ii. Welche Vor- und Nachteile hat der *Patricia*-Baum gegenüber einem *AVL*-Baum bzw. gegenüber *radix*-Suche?