Prof. Dr. Ir. Joost-Pieter Katoen

Jonathan Heinen, Sabrina von Styp

Hinweise:

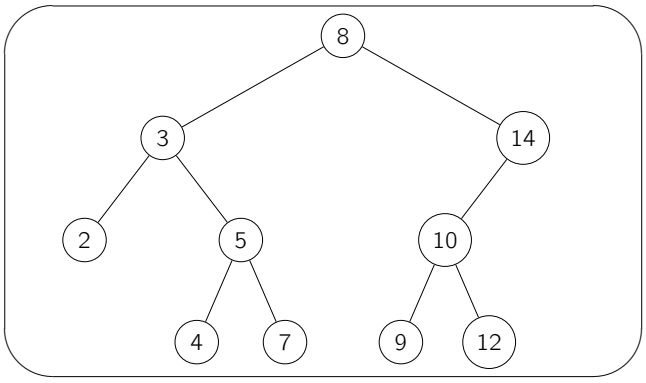
- Die Übungsblätter sollen in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung bearbeitet werden.
- Die Lösungen müssen bis Montag, den 7. Juni um 11:00 Uhr in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Namen und Matrikelnummern der Studenten sowie die Nummer der Übungsgruppe sind auf jedes Blatt der Abgabe zu schreiben. Heften bzw. tackern Sie die Blätter!
- Am Dienstag den 1. Juni findet, wegen dem RWTH SPORTS DAY, keine Vorlesung statt.

Aufgabe 1 (Binäsrer Suchbaum): (3 + 4 + 4 + 6 + 5 Punkte)

a) Geben Sie den binären Suchbaum an, der entsteht, wenn die folgenden Zahlen in *gegebener Reihenfolge* in einen *leeren Suchbaum* einfügt werden:

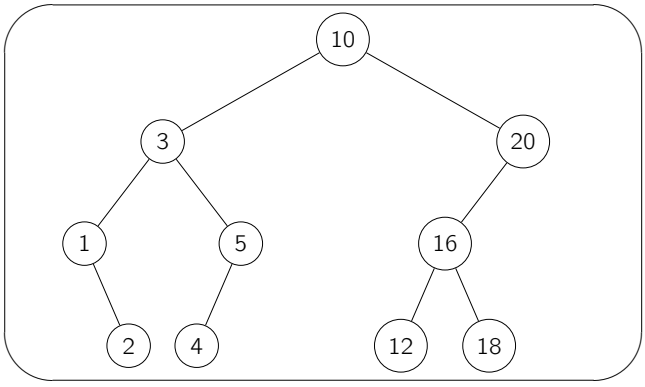
5 2 3 8 6 7 10 4

b) Gegeben sei der folgende binäre Suchbaum:

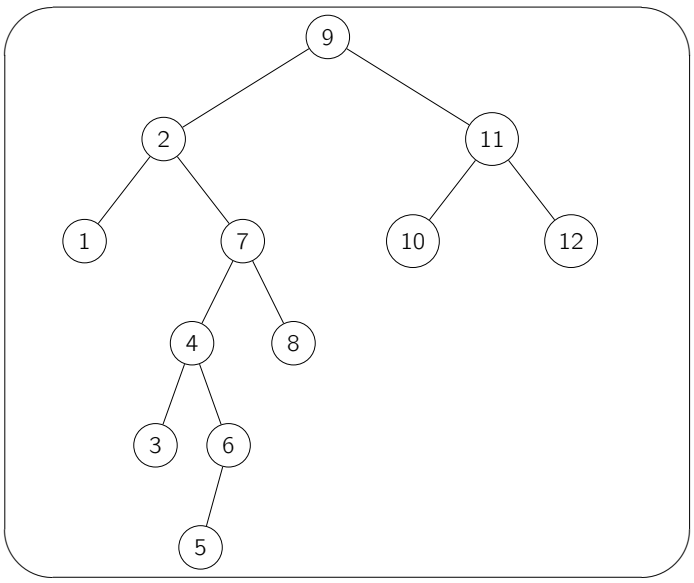


Geben Sie die binären Suchbäume an, die entstehen, wenn sie erst **14**, dann **3** und schließlich die **12**, entsprechend dem in der Vorlesung vorgestellten Verfahren, aus dem Baum heraus löschen.

c) Geben Sie für den folgenden binären Suchbaum die Binärbäume an, die entstehen, wenn eine **Rechtsrotation** auf den Knoten mit dem Wert **20** ausgeführt wird, dann eine **Linksrotation** auf den Knoten mit Wert **3** und zuletzt eine **Linksrotation auf die Wurzel** des entstandenen Baumes:



d) Geben Sie **maximal vier Rotationen** an, die den folgenden Baum der **Höhe fünf** in einen Baum der **Höhe drei** transformieren. Geben Sie auch den Zustand des Baumes nach jeder Rotation an.



e) In der Vorlesung wurde ein Verfahren zur *Bestimmung des Nachfolgers* eines Elementes im binären Suchbaum vorgestellt. Hierfür wird der Pfad zwischen den beiden Elementen zurückgelegt. Wie lang ist dieser Pfad maximal für einen Baum mit n Elementen? Beweisen Sie Ihre Aussage.

Aufgabe 2 (Quicksort): (3+3+4 Punkte)

a) Sortieren Sie das folgende Array mit Hilfe von *Quicksort*. Skizzieren Sie den Zustand des Arrays jeweils nach der Wahl eines neuen *Pivotelements* und geben Sie dieses an.

3	7	1	5	8	2	4	6

b) Zeigen Sie, anhand eines geeigneten Beispiels, dass die in der Vorlesung vorgestellte *Implementierung von Quicksort nicht stabil* ist.

Geben Sie eine geeignete Eingabe an, machen Sie gleiche Elemente durch geeignete Markierung unterscheidbar und führen Sie dann Quicksort auf der gewählten Eingabe aus, wobei Sie in jedem Schritt den Zustand der Folge sowie das Pivotelement angeben.

c) Modifizieren Sie die in der Vorlesung vorgestellte *Implementierung für Quicksort* so, dass sie *stabil* ist. Die *Komplexitätsklasse* der Laufzeit soll sich nicht verändern.

Hilfestellung: Möchte man, mit Hilfe eines *nicht stabilen* Sortieralgorithmus, *stabil* sortieren, so gibt es die Möglichkeit die ursprüngliche Position der Elemente als *sekundäres Sortierkriterium* hinzuziehen. Werden dann zwei Elemente mit gleichem *Primärschlüssel* verglichen wird die Sortierung anhand des *sekundären Sortierkriteriums* - also der ursprünglichen Position im Array - vorgenommen.

Aufgabe 3 (Höhergradige Heaps): (3+2+4+3 Punkte)

In Übung 5 haben Sie Heapsort, basierend auf *binären Heaps*, zum Sortieren genutzt. In binären Heaps hat jedes Element bis zu zwei Kinderelemente. Das Konzept der binären Heaps lässt sich auf *d-Heaps* erweitern. Ein *d*-Heap ist ein Baum mit Verzweigungsgrad *d* (d.h. jedes Element besitzt bis zu *d* Kinder), der die *max-Heapeigenschaft* (alle Kinderelemente haben *niedrigere* Werte) erfüllt.

a) Geben Sie die Formel an mit der, für die Arraydarstellung eines *d*-Heaps, die Position der Kinder des Elementes an Position *i* bestimmt werden kann.

In welchem Bereich des Arrays ist die Heapeigenschaft immer erfüllt?

b) Stellen Sie den im folgenden Array repräsentierten *3-Heap* als Baum dar.

40	32	24	36	28	30	5	12	21

c) Geben Sie eine modifizierte Version des in der Vorlesung vorgestellten Algorithmus `sink` an, der Elemente in einem *d*-Heap versickern lässt. Der Grad *d* wird hierbei als Parameter übergeben.

d) Sortieren Sie das in *b)* gegebene Array entsprechend der *Heapsortmethode für 3-Heaps*.

Aufgabe 4 (Optimales Sortierverfahren): (3 Punkte)

Bei der Kommunikation über Netzwerke kann nicht immer sichergestellt werden, dass die einzelnen Pakete beim Empfänger in der gleichen Reihenfolge ankommen wie sie versendet wurden. Um dadurch entstehende Probleme zu lösen, wird jedes Paket mit einer eindeutigen, aufsteigenden ID versehen. Eine Veränderung der Reihenfolge ist jedoch nicht der Regelfall, sondern eher die Ausnahme. Um die ursprüngliche Reihenfolge der Pakete wieder herzustellen werden diese, nach dem Empfangen aller Pakete, auf der Empfängerseite sortiert. Welcher Sortieralgorithmus ist für diese Aufgabe optimal geeignet? Begründen Sie Ihre Antwort.