TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

## 186.866 Algorithmen und Datenstrukturen VU

# Übungsblatt 3

PDF erstellt am: 14. April 2023

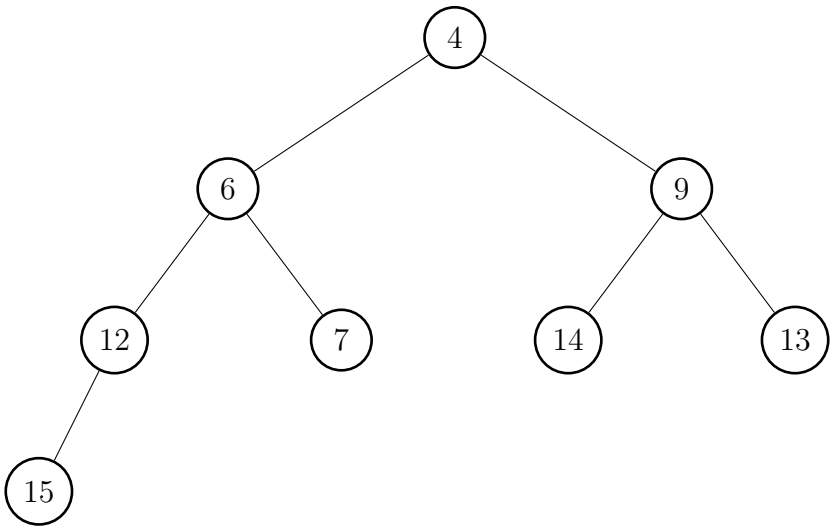
Deadline für dieses Übungsblatt ist **Montag, 01.05.2023, 20:00 Uhr**. Um Aufgaben für diese Übung anerkannt zu bekommen, gehen Sie folgendermaßen vor:

1. Öffnen Sie den TUWEL-Kurs der Lehrveranstaltung *186.866 Algorithmen und Datenstrukturen (VU 5.5)* und navigieren Sie zum Abschnitt *Übungsblätter*.
2. Teilen Sie uns mit, welche Aufgaben Sie gelöst haben **und** welche gelösten Aufgaben Sie gegebenenfalls in der Übungseinheit präsentieren können. Gehen Sie dabei folgendermaßen vor:
   - Laden Sie Ihre Lösungen in einem einzigen PDF-Dokument in TUWEL hoch.
     Link *Hochladen Lösungen Übungsblatt 3*
     Button *Abgabe hinzufügen*
     PDF-Datei mit Lösungen hochladen und *Änderungen sichern*.
   - Kreuzen Sie an, welche Aufgaben Sie gegebenenfalls in der Übung präsentieren können. Die Lösungen der angekreuzten Aufgaben müssen im hochgeladenen PDF enthalten sein.
     Link *Ankreuzen Übungsblatt 3*
     Button *Abgabe bearbeiten*
     Bearbeitete Aufgaben anhaken und *Änderungen speichern*.

Bitte beachten Sie:

- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft verändern. Nach der Deadline ist keine Veränderung mehr möglich. Es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen hochladen (beachten Sie die maximale Dateigröße).
- Beachten Sie die Richtlinien für das An- und Aberkennen von Aufgaben (Details finden Sie in den Folien der Vorbesprechung).

**Aufgabe 1.** Gegeben ist der folgende **Min-Heap**:




(a) Fügen Sie die Elemente 8 und 2 in dieser Reihenfolge in den Min-Heap ein. Stellen Sie nach jedem Einfügevorgang den resultierenden Heap als Baum dar. Erklären Sie dabei, wie Sie **Heapify-up** korrekt anwenden.

(b) Entnehmen Sie aus dem ursprünglich in Aufgabe 1 gegebenen **Min-Heap** drei Mal das kleinste Element. Stellen Sie nach jedem Löschvorgang den resultierenden Heap als Baum dar. Erklären Sie dabei, wie Sie **Heapify-down** korrekt anwenden.

**Anmerkung für Interessierte:** Mithilfe von Heaps können Arrays sortieren werden. Ein bekannter Sortieralgorithmus, der einen Heap als zentrale Datenstruktur verwendet, ist Heapsort.

---

**Aufgabe 2.**

(a) Gegeben ist das Array $[3, 6, 4, 2, 5, 7, 1]$. Führen Sie **Quicksort** auf dem oben gegebenen Array aus. Wählen Sie stets das erste Element als Pivot-Element aus und implementieren Sie den Schritt des Aufteilens so, dass die Elemente einer Subfolge immer in derselben Reihenfolge angeordnet werden wie in der Originalfolge. Geben Sie die Zwischenschritte wie im Foliensatz „Divide-and-Conquer“ an.

(b) Geben Sie eine Permutation dieses Arrays an, welches die Laufzeit des Algorithmus maximiert. Welche asymptotische Laufzeit hat der Algorithmus daher im Worst-Case in Abhängigkeit der Anzahl der zu sortierenden Elemente $n$?

---

**Aufgabe 3.** Ein Vertreter der linearen Sortieralgorithmen ist **Radixsort**. Die Grundidee basiert darauf, dass die zu sortierenden Wörter oder Zahlen mehrmals in Fächer (Buckets) verteilt werden, sodass nach der letzten Verteilung die Elemente in sortierter Reihenfolge entnommen werden können.

Finden Sie zunächst mithilfe einer Internet- und/oder Literaturrecherche heraus, wie Radixsort genau funktioniert.

(a) Sortieren Sie die folgenden Zahlen aufsteigend mithilfe von Radixsort bezüglich der Basis 10. Geben Sie Zwischenschritte mit an.

224, 206, 373, 234, 303, 326, 236

(b) Handelt es sich bei Radixsort um einen Divide-and-Conquer-Algorithmus? Begründen Sie Ihre Antwort.

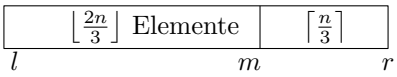
---

**Aufgabe 4.**

(a) Führen Sie **Mergesort** auf dem folgenden Array aus und geben Sie Zwischenschritte in Form eines Ablaufdiagrammes wie im Foliensatz „Divide-and-Conquer“ an.

| 54 | 21 | 93 | 39 | 89 | 71 | 19 |
|---|---|---|---|---|---|---|

(b) In der Vorlesung wurde die klassische **Mergesort** Variante vorgestellt, bei der ein Array immer in zwei gleich große Hälften geteilt wird (abgesehen davon, dass im Fall einer ungeraden Elementanzahl gerundet werden muss). In der Praxis kann es von Vorteil sein, manchmal ein Array nur in **ungefähr** gleich große Hälften zu teilen. Wir wollen argumentieren, dass dies die asymptotische Worst-Case Laufzeit nicht beeinflusst. Dafür nehmen wir beispielshalber an, dass die Aufteilung immer im Verhältnis $\frac{2}{3}$ zu $\frac{1}{3}$ geschieht.




Beweisen Sie die Laufzeit von $O(n \log n)$ für diese Aufteilung durch vollständige Induktion. Folgen Sie dabei dem Konzept aus der Vorlesung. Zeigen Sie, dass für die Anzahl der Vergleiche $C(n) \leq n \log_{\frac{3}{2}} n$ gilt.

**Hinweis:** Beachten Sie, dass für $n > 1$ gilt: $1 \leq \lceil \frac{n}{3} \rceil \leq \lfloor \frac{2n}{3} \rfloor \leq \frac{2n}{3}$. Dadurch vereinfacht sich die Rechnung aus der Vorlesung. Es ist sinnvoll mit $\log_{\frac{3}{2}}$ zu rechnen, da z.B. $\log_{\frac{3}{2}} \frac{2n}{3} = \log_{\frac{3}{2}} n - 1$ gilt.

---

**Aufgabe 5.** Gegeben ist die **preorder-Traversierung** eines Binärbaums

$$5, 3, 1, 2, 4, 8, 7, 6, 9.$$

Im Allgemeinen kann es mehrere Binärbäume geben, die diese preorder-Traversierung haben. Es gibt allerdings nur einen **binären Suchbaum** mit dieser preorder-Traversierung.

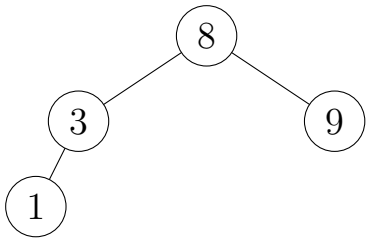
Rekonstruieren Sie mithilfe dieser Information den zugehörigen binären Suchbaum nach dem Divide-and-Conquer-Prinzip. Geben Sie dabei jeweils den aktuellen Zustand des Baumes nach jedem Teile-Schritt an.

**Optionale Zusatzaufgabe:** Geben Sie eine Einfügesequenz der Elemente $1, 2, 3, 4, 5, 6, 7, 8, 9$ an, die diesen binären Suchbaum erzeugt.

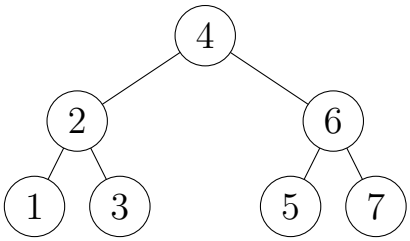
---

**Aufgabe 6.**

(a) Fügen Sie die Schlüssel 5 und 6 in dieser Reihenfolge in den folgenden AVL-Baum ein. Falls notwendig rebalancieren Sie den Baum nach jedem Einfügen mit einer geeigneten Operation (siehe Foliensatz „Suchbäume“), um wieder einen gültigen AVL-Baum zu erhalten. Zeichnen Sie den Baum auch in allen relevanten Zwischenschritten. Markieren Sie dabei den unbalancierten Knoten mit maximaler Tiefe.




(b) Löschen Sie aus dem folgenden AVL-Baum die Schlüssel $1, 2, 3$ in dieser Reihenfolge. Falls notwendig rebalancieren Sie wieder und markieren Sie dabei den unbalancierten Knoten mit maximaler Tiefe.




---