Notizen zur Klausurvorbereitung

Bastian - br0sinski @ github

March 3, 2025

Contents

1 Allgemeine Lernratschläge - Nutz nicht nur dieses Dokument!

- Nutze interaktive visuelle und didaktische Aufbereitungen.
- Wiederhole die Inhalte aus den ersten Vorlesungsfolien.
- Verwende das Moodle-Forum für Fragen und Diskussionen.

2 Definitionen

- **Fast vollständiger binärer Baum (fast complete binary search tree)**: Ein binärer Suchbaum, der die folgenden Eigenschaften erfüllt:
  - **Binärer Suchbaum (BST)**:
    - Jeder Knoten hat höchstens zwei Kinder (linkes und rechtes Kind).
    - Alle Werte im linken Teilbaum eines Knotens sind kleiner als der Wert des Knotens.
    - Alle Werte im rechten Teilbaum eines Knotens sind größer als der Wert des Knotens.
  - **Fast vollständige Baumstruktur**:
    - Alle Ebenen außer der letzten sind vollständig gefüllt.
    - In der letzten Ebene sind die Knoten so weit wie möglich nach links ausgerichtet.
    - Die Blätter befinden sich auf höchstens zwei aufeinanderfolgenden Ebenen.
- **Heap**: Ein Heap ist ein spezieller binärer Baum, der die folgende Bedingung erfüllt:
  - **Max-Heap**: Der Wert eines Knotens ist immer größer oder gleich den Werten seiner Kinder.
  - **Min-Heap**: Der Wert eines Knotens ist immer kleiner oder gleich den Werten seiner Kinder.

  In einem Heap sind die Knoten so angeordnet, dass der größte oder kleinste Wert immer an der Wurzel steht. Ein Heap ist auch fast vollständig, wobei nur die letzte Ebene von oben nach unten und von links nach rechts gefüllt ist.
- **AVL-Baum (Adelson-Velsky und Landis Baum)**: Ein AVL-Baum ist ein selbstbalancierender binärer Suchbaum, bei dem die Höhe der linken und rechten Teilbäume eines jeden Knotens sich um höchstens 1 unterscheidet. Das bedeutet:
  - Die Balancefaktor jedes Knotens (Differenz der Höhen des linken und rechten Teilbaums) liegt zwischen -1 und 1.

  Ein AVL-Baum garantiert, dass alle grundlegenden Operationen wie Suchen, Einfügen und Löschen in logarithmischer Zeit $O(\log n)$ ausgeführt werden.

3 Datenstrukturen und Algorithmen

3.1 Komplexitätsanalyse

- Übersicht der Laufzeitkomplexitäten verschiedener Algorithmen.
- Big-O-Notation für verschiedene Datenstrukturen.

3.2 Sortieralgorithmen

- Heapsort: Zeitkomplexität und Eigenschaften.
- Quicksort: Schrittweise Anwendung mit Beispiel.

4 Beispielaufgaben

4.1 Funktionsweise von Algorithmen

- Sortieren eines Arrays mit Heapsort.
- Aufbau eines Heaps mit insert().

4.2 Java-Implementierungen

- Beispielcode für die PriorityQueue mit AVL-Baum.
- Algorithmus zur effizienten Suche nach gemeinsamen Elementen in zwei Arrays.

5 Zusätzliche Anmerkungen

- Links zu relevanten Moodle-Diskussionen und Ressourcen.
- Checkliste für die Klausurvorbereitung.