Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme

Sommersemester 2021
06. 06. 2021

Prof. Dr. Bernhard Seeger
Nikolaus Glombiewski, M.Sc.
Amir El-Shaikh, M.Sc.

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Abgabe: 13. 06. 2021,
bis **spätestens** 23:59 Uhr
über die ILIAS Plattform

Übung 8

Aufgabe 8.1: Heaps (1+1+1+2+1) (6 Punkte)

Im ILIAS finden Sie die Implementierung eines generischen Heaps (Klasse `Heap`).

a) Ein Heap kann aus einem unsortierten n-elementigen Array in linearer Zeit aufgebaut werden. Dies wurde in der Vorlesung besprochen und eine detaillierte Beschreibung dieser Methode finden Sie unter https://de.wikipedia.org/wiki/Binärer_Heap#Algorithmen. Vervollständigen Sie den Konstruktor `Heap(T[] values, Comparator<T> comp)`, so dass der Heap in linearer Zeit aus dem übergebenen Array aufgebaut wird.
Tipp: Die `heapify`-Methode des Artikels entspricht der `downheap`-Methode der Klasse `Heap`.

b) Vervollständigen Sie die Methode `findKSmallestNaive(Integer[] elems, int k)`, die das k-kleinste Element des übergebenen Arrays findet, indem sie alle Elemente nacheinander in einen anfangs leeren `Heap` einfügt.

c) Vervollständigen Sie die Methode `findKSmallestHeapify(Integer[] elems, int k)`, die das k-kleinste Element des übergebenen Arrays findet, indem sie den in Teilaufgabe a) implementierten Konstruktor zum Aufbau des Heaps verwendet.

d) Vervollständigen Sie die Methode `findKSmallestMaxHeap(Integer[] elems, int k)`. Diese verwendet einen Max(!)-Heap zum Auffinden des k-kleinsten Elements. Im ersten Schritt werden die ersten k Elemente des Arrays hinzugefügt. Für alle weiteren Elemente wird nach dem Hinzufügen das größte Element wieder aus dem Heap entfernt. Wurde das gesamte Array durchlaufen, befindet sich das k-kleinste Element an der Spitze des Heaps.

e) Geben Sie die Laufzeiten der drei Methoden aus b), c) und d) in $\mathcal{O}$-Notation an. Führen Sie anschließend die `main`-Methode aus und erstellen Sie einen Graphen, der die Laufzeiten der drei Methoden in Abhängigkeit von k zeigt. Diskutieren Sie den Zusammenhang zwischen den gemessenen und theoretischen Laufzeiten.

Aufgabe 8.2: Binäre Suchbäume (2+2+2+2+1) (9 Punkte)

a) Erstellen Sie eine Unterklasse namens der gegebenen Klasse `BinaryTree<K>`, welche den Namen `BinarySearchTree<K>` hat und die einen binären Suchbaum implementiert. Im Gegensatz zum `BinaryTree` soll der `BinarySearchTree` nur Elemente aufnehmen, die das `Comparable`-Interface implementieren. Überschreiben Sie die beiden Methoden `getLeft` und `getRight`, so dass sie den linken beziehungsweise rechten Teilbaum als Instanz der Klasse `BinarySearchTree` zurückgeben.

b) Ergänzen Sie die Klasse `BinarySearchTree` um die Methoden `add(key)`, `search(key)` und `remove(key)`. Diese Methoden sollen entsprechend einen neuen Knoten hinzufügen, einen bestehenden suchen (und zurückgeben) und einen bestehenden Knoten aus dem Baum löschen. Realisieren Sie den Löschvorgang mittels einer separaten Helfer-Methode namens `removeSymmetricPredecessor`, die analog zur Vorlesung den symmetrischen **Vorgänger** zum Löschen verwendet.

c) Implementieren Sie zwei Methoden `getMinKey` und `getMaxKey`, die den kleinsten bzw. größten Schlüssel des Baums zurückgeben.

d) Implementieren Sie zwei Methoden `getSuccessor(node)` und `getPredecessor(node)`, die für einen gegebenen Knoten im aktuellen Baum den Nachfolger- bzw. Vorgänger-Knoten bezüglich der Sortierreihenfolge findet und ausgibt.

e) Implementieren Sie eine Methode `getMaxKey(x)`, die mit Aufwand $\mathcal{O}(h)$ den größten im Suchbaum vorhandenen Schlüssel y mit $y \leq x$ zurückgibt. Dabei ist h die Höhe des Suchbaumes.

Aufgabe 8.3: Algorithmen Entwurf* (2+3) **(5 Punkte)**

Gegeben sei eine Sammlung von n Computer-Spielen. Ein Computer-Spiel ist durch das Paar (Name, Erscheinungsjahr) gegeben. Gesucht ist ein Algorithmus, der alle Computer-Spiele aus der Sammlung nach Erscheinungsjahr unterteilt und zurückgibt.

a) Beschreiben Sie einen Algorithmus, der das Problem in $\Theta(n \log n)$ löst.

b) Kann das beschriebene Problem mit erwarteter Laufzeit von $\Theta(n)$ gelöst werden? Begründen Sie Ihre Antwort.

*** Aufgabe 8.3 ist für Lehramtsstudierende optional.**