Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme

Sommersemester 2023
08. 05. 2023

Prof. Dr. Bernhard Seeger
Jana Holznigenkemper M.Sc.
Marius Kuhrt M.Sc.

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Abgabe: 14. 05. 2023,
bis **spätestens** 23:59 Uhr
über die ILIAS Plattform

Übung 4

Aufgabe 4.1: Verwendung von Listen (5+3) (8 Punkte)

Eine Menge (englisch: set) ist eine Datenstruktur, in der jedes Element nur einmal vorkommen darf (vgl. Menge in der Mathematik).

a) Vervollständigen Sie in der im Ilias zur Verfügung gestellten Klasse `ListSet` die Methoden `add`, `remove`, `union` (Vereinigung), `intersect` (Schnitt) und `subtract` (Differenz).

b) Modifizieren Sie Ihre Implementierungen aus a) so, dass `union` bei einer Vereinigung von einer Menge mit n Elementen mit einer Menge mit m Elementen im schlechtesten Fall eine Laufzeit von $\mathcal{O}(n+m)$ besitzt.

Aufgabe 4.2: Stacks und Queues (4+2) (6 Punkte)

a) Gegeben sei ein Array der Größe n. Beschreiben Sie ein Verfahren, um 2 verschiedene Stacks innerhalb von diesem einem Array zu verwalten, solange die Summe der Größe der beiden Stacks kleiner gleich n ist. Geben Sie Pseudocode für die `push`- und `pop`-Operationen an, welche jeweils eine Laufzeitkomplexität von $\mathcal{O}(1)$ aufweisen. Fangen Sie den Fall, dass die Summe der beiden Stackgrößen n überschreitet, in Ihrem Pseudeocode an entsprechender Stelle ab.

b) Eine *Queue* ist eine Warteschlange nach dem FIFO-Prinzip (First In – First Out). Dabei fügt die Operation `add` ein Element in die Queue ein und die Operation `poll` entnimmt das Element, welches sich am längstem in der Queue befindet.

Führen Sie die folgenden Operationen manuell auf einer Queue aus. Welche Zahl wird durch die letzte Anweisung ausgegeben? Wie sieht die Queue nach der Ausgabe aus? Der Rechenweg soll ersichtlich sein.

```
add(5);
add(2);
poll();
add(13);
add(poll()*poll());
add(13);
add(3);
add(poll()+poll());
add(9)
add(poll()/poll());
System.out.println(poll());
```

Aufgabe 4.3: Skip-Listen* **(6 Punkte)**

Stellen Sie für das Interface `RandomSkipList` eine *generische* Implementierung einer randomisierten Skip-Liste bereit.

*** Aufgabe 4.3 ist für Lehramtsstudierende optional.**