Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 09

Algorithmen I – Sommersemester 2022

Abgabe im ILIAS bis 29.06.2022, 14:00 Uhr

Bitte beschrifte Deine Abgabe gut sichtbar mit Deinem Namen und Deiner Matrikelnummer. Achte insbesondere bei handschriftlichen Abgaben auf Lesbarkeit und genügend Platz für Korrektur-Anmerkungen. Die Abgabe erfolgt über das Übungsmodul in der Gruppe Deines Tutoriums im ILIAS. Gib Deine Ausarbeitungen in *einer* PDF-Datei ab. Achte darauf, effiziente Algorithmen zu formulieren, also solche mit möglichst geringer asymptotischer Laufzeit!
Wenn du die Korrektheit eines Algorithmus begründen oder dessen Laufzeit analysieren sollst, tue dies getrennt von der Beschreibung des Algorithmus.
Wenn nicht anders spezifiziert oder aus dem Kontext ersichtlich, bezeichnen wir mit Graph einen einfachen ungerichteten Graphen.

Aufgabe 1 - Nicht ganz Corona-konform (6 Punkte)

Dr. Meta schmuggelt sich undercover in die Mitarbeiterversammlung, um herauszufinden, wie stark die Miterarbeiterschaft nach der Ausdünnung noch vernetzt ist. Dazu zählt er, wie viele Mitarbeiter sich gegenseitig die Hand geben und macht eine überraschende Feststellung: Die Anzahl an Mitarbeitern, die einer ungeraden Anzahl ihrer Kollegen die Hand geben, scheint stets gerade zu sein. Dr. Meta braucht nun deine Hilfe, um diese Aussage zu beweisen.

1. Beschreibe, wie das Handgeben auf der Mitarbeiterversammlung als Graph modelliert werden kann. (1 Punkt)
   Hinweis: Was sind die Knoten? Zwischen welchen Knoten gibt es eine Kante? Ist der Graph gerichtet oder nicht? Brauchst du Kantengewichte?

2. Übersetze Dr. Metas Hypothese in eine Aussage über den Graph aus der ersten Teilaufgabe. (2 Punkte)

Nachdem wir das Problem als Graphproblem formuliert haben, geht es nun an den Beweis. Dieser soll sich aus folgenden Schritten zusammensetzen:

3. Sei $G = (V, E)$ ein Graph. Wir betrachten einen Algorithmus, der die Summe der Knotengrade aller Knoten berechnet. Dabei iteriert er alle Knoten und macht für jeden Knoten $v \in V$ folgendes. Alle zu v inzidenten Kanten werden iteriert und für jede gesehene Kante wird die Summe um 1 erhöht.
   Sei $e \in E$ eine beliebige Kante. Wie oft wird e vom Algorithmus gesehen? (1 Punkt)

4. Stelle eine Gleichung auf, die die Summe der Knotengrade in einem beliebigen Graphen G in Beziehung zur Anzahl der Kanten in G darstellt. (1 Punkt)

5. Benutze die Erkenntnisse aus den vorherigen Teilaufgaben, um die Aussage aus Teilaufgabe 2 zu beweisen. (1 Punkt)
   Hinweis: Überlege, wie man von der Summe aller Knotengrade zur Summe über Knoten mit ungeradem Grad kommt.

Aufgabe 2 - Déjà vu (6 Punkte)

In dieser Aufgabe möchten wir eine Datenstruktur entwickeln, mit der wir auf einer dynamischen Menge von geordneten Elementen für gegebene k effizient das k-kleinste Element finden können. Analog zu balancierten Suchbäumen soll es möglich sein in $O(\log n)$ Zeit Elemente einzufügen, zu finden und zu löschen.
Um dies Umzusetzen möchten wir (a, b)-Bäume erweitern. Gegeben sei dazu ein (a, b)-Baum, der die Elemente aus einer Menge $S \subset \mathbb{N}$ der Größe n hält.

1. Beschreibe, welche zusätzlichen Informationen in den Knoten des (a, b)-Baumes gespeichert werden können, um in $O(\log(n))$ das k-kleinste enthaltene Element in S zu finden. (1 Punkt)

2. Begründe wieso das asymptotische Laufzeitverhalten von INSERT, FIND und REMOVE trotz der Ergänzungen aus Teilaufgabe 1 gleich bleibt. (3 Punkte)
   Hinweis: Überlege, wie sich die zusätzlichen Informationen in den Knoten durch Operationen auf dem (a, b)-Baum ändern können und wie sie aktuell gehalten werden können.

3. Beschreibe einen Algorithmus, der auf einem (a, b)-Baum mit deinen Erweiterungen aus Teilaufgabe 1 das k-kleinste Element in Zeit $\Theta(\log n)$ findet. Begründe die Korrektheit deines Algorithmus und dass er das geforderte Laufzeitverhalten hat. (2 Punkte)

Aufgabe 3 - Baumschule (5 Punkte)

In dieser Aufgabe wollen wir uns mit der Konstruktion von $(2, 3)$-Bäumen beschäftigen.

1. Beschreibe, wie in Linearzeit aus einer gegebenen sortierten Folge ein $(2, 3)$-Baum konstruiert werden kann. Begründe, warum dein Algorithmus das geforderte Laufzeitverhalten hat. (3 Punkte)

2. Beschreibe, wie in Linearzeit zwei gegebene $(2, 3)$-Bäume zu einem $(2, 3)$-Baum zusammengefügt werden können. Begründe, warum dein Algorithmus das geforderte Laufzeitverhalten hat. (2 Punkte)

Aufgabe 4 - Geburtstag (3 Punkte)

Um immer sicherzugehen, dass du nie mehr vergisst, einem Freund zum Geburtstag zu gratulieren, hast du beschlossen, dich zu Beginn jedes Monats an alle anstehenden Geburtstage erinnern zu lassen. Da du dich im Rahmen der letzten Vorlesungen ausgiebig mit (a, b)-Bäumen befasst hast, beschließt du, sie zu diesem Zwecke zu verwenden. Die Geburtstage deiner n Freunde hast du bereits in einer sortierten Folge gesammelt, wobei die Elemente Tupel der Form (Geburtstag, Name) sind, welche nach der ersten Komponente (Geburtstag) und im Falle von gleichen Geburtstagen nach der zweiten Komponente (Name) sortiert sind. Aufgrund der hohen Verwechslungsgefahr können wir annehmen, dass die Folge keine Duplikate enthält. Die Tage des Jahres sind in aufsteigender Reihenfolge durchnummeriert, d.h. jeder Geburtstag wird durch eine natürliche Zahl im Intervall $[0, 365]$ repräsentiert.
Unter der Annahme, dass im angefragten Monat k Geburtstage liegen, beschreibe, wie du diese mit Hilfe eines (a, b)-Baumes, in $O(k + \log(n))$ bestimmen kannst. Begründe, warum dein Algorithmus das geforderte Laufzeitverhalten hat.