Karlsruher Institut für Technologie

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

Übungsblatt 08

Algorithmen I – Sommersemester 2022

Abgabe im ILIAS bis 22.06.2022, 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 - Kleiner Haufen ganz groß (4 Punkte)

Ein binary Min-Heap kann mithilfe eines Arrays umgesetzt werden, um Elemente effizient so zu verwalten, dass man schnell das Minimum extrahieren kann. Natürlich lässt sich auch leicht eine Variante definieren, die einem schnellen Zugriff auf das Maximum gibt.

1. Zeichne den Max-Heap der entsteht, wenn man die folgende Reihe von Zahlen einfügt:

   $$(5, 3, 23, 17, 10, 4, 30, 32, 31)$$

   Zeichne außerdem den Max-Heap, nachdem ein Mal `popMax`() aufgerufen wurde.
   (1 Punkt)

2. Beschreibe einen Algorithmus, der als Eingabe ein Array $A\colon [\mathbb{N}; n]$ erhält und in Zeit $O(n)$ entscheidet, ob A einen Max-Heap repräsentiert. Begründe die Korrektheit deines Algorithmus und dass er das geforderte Laufzeitverhalten hat.
   (3 Punkte)

Aufgabe 2 - Déjà vu (3+4 Punkte)

Schon wieder haben wir ein unsortiertes Array $A\colon [\mathbb{N}; n]$ und wollen das k-kleinste Element in A finden.

1. Beschreibe einen Algorithmus, der unter Benutzung eines Heaps das k-kleinste Element in A mit einem Zeitbedarf in $O(n \log(k))$ ausgibt. Begründe die Korrektheit deines Algorithmus und dass er das geforderte Laufzeitverhalten hat. (2 Punkte)

*. Beschreibe einen Algorithmus, der unter Benutzung von Heaps das k-kleinste Element in A mit einem Zeitbedarf in $O(n + k \log(k))$ ausgibt. Begründe die Korrektheit deines Algorithmus und dass er das geforderte Laufzeitverhalten hat. (4 Punkte)

2. Wir wollen nun die Laufzeit des Algorithmus aus 2.* mit der des Algorithmus aus der Bonusaufgabe von Blatt 04 vergleichen. Beschreibe unter welchen Umständen man welchen Algorithmus eher verwenden sollte. (1 Punkt)

Aufgabe 3 - Dairy Min-Heap (7 Punkte)

Das Prinzip von binary Min-Heaps lässt sich leicht auf Min-Heaps höheren Grades erweitern. Während ein Knoten im binary Min-Heap maximal zwei Kinder hat, sind es beim d-ary Min-Heap bis zu d Kinder.

1. Beschreibe einen Algorithmus der die Operationen `push` in einem d-ary Min-Heap umsetzt und analysiere dessen Laufzeit in Abhängigkeit von n und d. (2 Punkte)

2. Beschreibe einen Algorithmus der die Operationen `decPrio` in einem d-ary Min-Heap umsetzt und analysiere dessen Laufzeit in Abhängigkeit von n und d. (2 Punkte)

3. Beschreibe einen Algorithmus der die Operationen `popMin` in einem d-ary Min-Heap umsetzt und analysiere dessen Laufzeit in Abhängigkeit von n und d. (2 Punkte)

4. Beschreibe unter welchen Umständen man einen d-ary Min-Heap für $d > 2$ einem binary Min-Heap vorziehen sollte. (1 Punkt)

Aufgabe 4 - Kontaktbeschränkungen (4 Punkte)

Bevor Dr. Meta ihn zur Rechenschaft hat ziehen können, ist der Verräter getürmt. Doch das beruhigt Dr. Meta nicht, im Gegenteil: er befürchtet, dass sich noch weitere seiner Handlanger haben korrumpieren lassen. Deswegen greift er zu drastischeren Maßnahmen. Er hat einen Graph erstellt, der abbildet welche Mitarbeiter miteinander in Kontakt stehen. Um zu verhindern, dass die Verräter in Zukunft noch viele weitere Mitarbeiter

gegen ihn aufhetzen, soll dieser Graph nun ausgedünnt werden. Der Plan ist, nach und nach den Mitarbeiter mit den meisten Kontakten zu „entfernen“, bis die Anzahl der verbleibenden Kontakte ausreichend klein ist. Dabei sollen jedoch nicht unnötig viele Arbeitskräfte verloren gehen.

Im folgenden soll nun ein Algorithmus entworfen werden, der die zu entfernenden Mitarbeiter identifiziert.

1. Wir betrachten einen Graph G und sortieren seine n Knoten nach Knotengrad in absteigender Reihenfolge. Nun soll iterativ ein höchstgradiger Knoten inklusive seiner inzidenten Kanten entfernt und die verbleibenden Knoten erneut sortiert werden. Zeige, dass sich die Reihenfolge der niedriggradigeren Knoten dadurch verändern kann. (1 Punkt)

2. Beschreibe einen Algorithmus, der als Eingabe einen Graph G und eine Zahl k erhält und iterativ einen Knoten mit dem größten Grad (inklusive der inzidenten Kanten) aus dem Graph entfernt, bis die Anzahl der verbleibenden Kanten höchstens k aber möglichst groß ist. (2 Punkte)

3. Im folgenden darfst du annehmen, dass ein Knoten v mit Grad $\deg(v)$ inklusive seiner inzidenten Kanten aus in Zeit $O(\deg(v))$ aus einem Graphen entfernen kannst. Zeige, dass dein Algorithmus eine Laufzeit von $O(n + m \log(n))$ hat. (1 Punkt)