Karlsruher Institut für Technologie

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

Übungsblatt 08

Algorithmen I – Sommersemester 2023

Abgabe im ILIAS bis 23.06.2023, 18:00 Uhr

Die Abgabe erfolgt als *eine* PDF-Datei über das Übungsmodul in der Gruppe deines Tutoriums im ILIAS.
Beachte bitte die Hinweise zum Bearbeiten auf der Webseite.

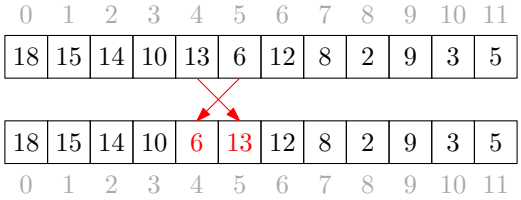
Aufgabe 1 - Über den Haufen geworfen (6 Punkte)

Binary Min-Heaps sind nützlich, 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. Im Verlauf der Aufgabe wollen wir uns ausschließlich mit solchen *Max-Heaps* beschäftigen.

a) Zeichne den *Max-Heap* der entsteht, wenn man die folgenden Zahlen nacheinander einfügt: $(15, 22, 29, 14, 38, 11, 43, 49, 37)$.

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

Wir haben in der Vorlesung gelernt, dass man so einen Heap gut mithilfe eines Arrays darstellen kann. Im folgenden bezeichnen wir ein Array als *gültiges Heap-Array*, wenn es einen Max-Heap repräsentiert. Nun können sich bei Datenübertragungen ab und an Fehler einschleichen. (Ursachen können Random Bit-Flips, kosmische Strahlen oder Biber sein.) Im folgenden gehen wir davon aus, dass sich der Fehler durch einen *Swap* äußert, bei dem ein Element im Array mit einem benachbarten Element getauscht wurde. Ein gültiges Heap-Array, das aufgrund eines Swaps keinen Heap mehr repräsentiert, bezeichnen wir als *ungültig*.



Ein gültiges Heap-Array, bevor der Fehler aufgetreten ist, und darunter das ungültige Heap-Array.

b) Gib ein ungültiges Heap-Array A' an, bei dem es ein Element gibt, das mit beiden Nachbarn geswappt werden kann, um jeweils ein gültiges Heap-Array zu erhalten. Zeichne den ungültigen Heap der durch A' repräsentiert wird, sowie die beiden gültigen Heaps, die durch die Swaps entstehen. (2 Punkte)

c) Beschreibe einen Algorithmus, der als Eingabe ein ungültiges Heap-Array mit n Elementen bekommt und alle Indexpaare von benachbarten Elementen ausgibt, deren Swap zu einem gültigen Heap-Array führt. Begründe, dass dein Algorithmus eine Laufzeit in $O(n)$ hat.
Hinweis: Ein ungültiges Heap-Array kann mit nur einem Swap gefixt werden. (3 Punkte)

Aufgabe 2 - Kleinigkeiten (6 Punkte)

Während ein Min-Heap geeignet ist, um schnell auf das kleinste Element einer Menge zugreifen zu können, sind wir im folgenden an mehr als nur dem kleinsten Element interessiert. Das i-kleinste Element ist dabei das Element, das in einem sortierten Array der Eintrag an Index $i-1$ ist.

a) Beschreibe einen Algorithmus, der als Eingabe ein unsortiertes Array mit n Elementen erhält und das i-kleinste Element in Zeit $O(n\log(i)+n)$ ausgibt. Begründe die Korrektheit des Algorithmus und warum er das gewünschte Laufzeitverhalten hat. (3 Punkte)

Mit diesem Ansatz ist der Zugriff auf das i-kleinste Element ziemlich teuer. Im folgenden soll deshalb eine Datenstruktur namens k-Heap entworfen werden, welches die folgenden Operationen unterstützt:

- `insert`: Fügt ein Element in die Datenstruktur ein.
- `get(`i`)`, mit $i \in \{1,\dots,k\}$: Gibt das i-kleinste Element aus (falls es existiert).
- `pop(`i`)`, mit $i \in \{1,\dots,k\}$: Entfernt das i-kleinste Element aus der Datenstruktur (falls es existiert).

b) Beschreibe eine Datenstruktur, mit der die Operationen `insert` und `pop(`i`)` in Zeit $O(\log(n)+k)$ und `get(`i`)` in Zeit $O(1)$ ausgeführt werden können, wobei n die aktuelle Anzahl der Elemente in der Datenstruktur ist. Beschreibe dazu auch, wie die Operationen mithilfe deiner Datenstruktur umgesetzt werden und begründe jeweils die Laufzeit. (3 Punkte)

Aufgabe 3 - Dairy Min-Heap (4 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.

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

b) 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)

Aufgabe 4 - Eindämmung (5 Punkte)

Diese Holzköpfe! Allen Bibern wurden genaue Anweisungen gegeben, welche Züge sie nehmen müssen, um zu den Baustellen zu gelangen. Ausgestattet mit dem Orientierungssinn eines Baumstumpfes irren die Biber nun aber durchs Land, weil sie in die falschen Züge einsteigen. Dr. Meta muss wieder mal eingreifen. Diesmal wird zu drastischeren Mitteln gegriffen. Eine gesondert ausgebildete Division von Bibern, die *NSA* (Nager-Spezial-Abteilung), soll eingesetzt werden, um das Zugnetzwerk durch buchstäbliche *Eindämmung* zu vereinfachen. In absteigender Größe (gemessen an der Anzahl anliegender Schienen) werden Bahnhöfe dabei aus dem Verkehr gezogen, indem die Schienen mit Dämmen blockiert werden.

a) Wir repräsentieren das Zugnetzwerk mithilfe eines Graphen G, bei dem die n Bahnhöfe die Knoten sind und die Schienen als Kanten zwischen ihnen dargestellt werden. Angenommen, wir haben die Knoten nach Knotengrad in absteigender Reihenfolge sortiert und wollen nun iterativ einen höchstgradigen Knoten inklusive seiner inzidenten Kanten entfernen und die verbleibenden Knoten erneut sortieren. Zeige, dass sich die Reihenfolge der niedriggradigeren Knoten dadurch verändern kann. (1 Punkt)

b) Beschreibe einen Algorithmus, der als Eingabe einen Graph G und eine Zahl $k \geq 1$ erhält und iterativ einen Knoten mit dem größten Grad (inklusive der inzidenten Kanten) aus dem Graph entfernt, bis der durchschnittliche Knotengrad höchstens k aber möglichst groß ist.

   Zeige, dass dein Algorithmus eine Laufzeit von $O(n + m \log(n))$ hat.
   Hinweis: Du darfst annehmen, dass ein Knoten v mit Grad $\deg(v)$ inklusive seiner inzidenten Kanten in Zeit $O(\deg(v))$ aus einem Graphen entfernt werden kann. (4 Punkte)