technische universität dortmund

Fakultät für Informatik
Lehrstuhl 11

Mart Hagedoorn
Guangping Li

Sommersemester 2024
26. April

DAP2 Praktikum – Blatt 4

Abgabe: KW 19

Studienleistung

- Zum Bestehen des Praktikums muss jeder Teilnehmer*innen die folgenden Leistungen erbringen:
  - Es müssen mindestens 50 Prozent der Punkte in den Kurzaufgaben erreicht werden.
  - Es müssen mindestens 50 Prozent der Punkte in den Langaufgaben erreicht werden.
- Im Krankheitsfall kann ein Testat bei Vorlage eines Attests in der folgenden Woche nachgeholt werden.
- Wenn ein Praktikumstermin auf einen Feiertag fällt, müssen Sie sich an einem beliebigen anderen Praktikumstermin in der gleichen Woche testieren lassen.
- **Hinweis:** Notieren Sie sich Ihre Punkte nach jedem Testat! Dies dient der eigenen Kontrolle. (Ihr Punktestand kann Ihnen während des Semesters nicht genannt werden.)

Wichtige Information (im Moodle verfügbar)

- Beachten Sie die Erklärung des **Ablaufs (Blatt A)**.
- Beachten Sie die **Regeln und Hinweise (Blatt R)** in der aktuellsten Version!
- Beachten Sie die **Hilfestellungen (Blatt H)** in der aktuellsten Version!

Langaufgabe 4.1: Min-Heaps und Heap-Select (4 Punkte)

In der Vorlesung haben sie Max-Heaps kennengelernt, welche Sie als Priority-Queue oder zum deterministischen Sortieren in $\mathcal{O}(n \log n)$ Zeit nutzen können. Natürlich können Sie auch ein gängiges Lehrbuch (z.B. "Introduction to Algorithms" von Cormen et al.) verwenden. In dieser Aufgabe sollen sie *Min*-Heaps implementieren, die analog zu Max-Heaps funktionieren, aber immer das kleinste Element in der Wurzel speichern.

- Implementieren Sie die Methode `minHeapify(data,i,n)`, welche $\mathcal{O}(\log n)$ Zeit braucht. Das Argument n ist die Anzahl der Elemente im Heap, welche nicht unbedingt mit der Länge des Arrays übereinstimmt. Das Argument i ist der Knoten, für welchen Sie die Heap-Eigenschaft herstellen wollen.

  ```
  public static void minHeapify(int [] data, int i, int n)
  ```

- Implementieren Sie die Methode `buildMinHeap(data)`, welche mittels `minHeapify` einen Min-Heap erzeugt. Dies funktioniert in $\mathcal{O}(\texttt{data.length} \cdot \log(\texttt{data.length}))$ Zeit.

  ```
  public static void buildMinHeap(int [] data)
  ```

- Implementieren Sie die Methode `extractMin(data,n)`, welche in $\mathcal{O}(\log n)$ Zeit das kleinste Element aus dem Heap entfernt und ausgibt. Danach befinden sich die verbleibenden Heap-Elemente in $\texttt{data}[0, n-2]$, und das entfernte Minimum steht an Position $\texttt{data}[n-1]$.

  ```
  public static int extractMin(int [] data, int n)
  ```

- Mit hilfe eines Min-Heaps können Sie leicht das *Auswahlproblem* lösen. Dieses fragt nach dem k-kleinsten Element eines Arrays der Länge n (ein klassischer Anwendungsfall des Auswahlproblems ist die Suche nach dem Median, also dem $\left\lfloor \frac{n}{2} \right\rfloor$-kleinsten Element). Implementieren Sie eine Methode `heapSelect(data, k)`, die mithilfe der vorherigen drei Methoden das k-kleinste Element in `data` findet und ausgibt. Die Methode darf keine Hilfsarrays verwenden. Der Inhalt von `data` darf beliebig permutiert werden.

  ```
  public static int heapSelect(int [] data, int k)
  ```

Abschließend sollen Sie eine geeignete `main`-Methode schreiben, sodass Ihr Porgramm das eine Ganzzahl n und eine Liste von Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In (Nach jedem Wert bestätigen Sie mit der Enter-Taste), sowie ein Argument k erhält, und dann den k-kleinsten Wert der Liste ausgibt. Wenn es in der Liste Duplikate gibt, dann brauchen Sie diese nicht besonders zu behandeln:

```
java B4A1.java 3 <Enter>
5 1 2 2 3 4 <Enter>
Input Array: [1, 2, 2, 3, 4]
The 3-smallest element is 2
```

Wie immer können Sie davon ausgehen, dass die Eingabe korrekt ist und müssen den Code nicht um diese Eingabe Robust gestalten.

Langaufgabe 4.2: Quick-Select (4 Punkte)

In Aufgabe 1 haben sie das Auswahlproblem gelöst. Mithilfe eines Heaps finden Sie das k-kleinste Element eines Arrays der Länge n in $\mathcal{O}(n \log n)$ Zeit. Asymptotisch hätten Sie also genau so viel Zeit gebraucht, wenn Sie das Array einfach vollständig sortiert hätten. In der Vorlesung werden sie einen besseren Algorithmus für das Auswahlproblem kennenlernen, welcher es deterministisch und in $\mathcal{O}(n)$ Zeit löst.

Auf diesem Blatt sollen Sie stattdessen den randomisierten Algorithmus *Quick-Select* implementieren. Für ein Arrayinterval $A[\ell, r]$ funktioniert dies so:

- Sie wollen das k-kleinste Element in $A[\ell, r]$ finden. Dazu wählen Sie eine zufällige Position $p \in \{\ell, \ldots, r\}$. Nun verwenden Sie $q = A[p]$ als Pivot-Element und partitionieren Sie das Arrayinterval wie folgendes:

  Beim Partitionieren stellen Sie fest, dass m Elemente kleiner als q sind, sodass anschließend $\forall i \in \{\ell, \ldots, \ell + m - 1\} : A[i] < q$ und $\forall i \in \{\ell + m, \ldots, r\} : A[i] \geq q$

- Wenn $k \leq m$, dann bestimmen Sie rekursiv das k-kleinste Element in $A[\ell, \ell + m - 1]$. Ansonsten bestimmen Sie rekursiv das $(k - m)$-kleinste Element in $A[\ell + m, r]$.

Die Implementierung sieht wie folgt aus:

- Die Methode `partition` bekommt die Pivot-Position p übergeben und führt die Partitionierung aus. Es gilt also $\ell \leq p \leq r$.

  ```
  public static int partition(int [] data, int l, int p, int r)
  ```

- Die Methode `quickSelect` findet rekursiv das k-kleinste Element im Interval $[\ell, r]$, und wählt immer eine zufällige Pivot-Position.

  ```
  public static int quickSelect(int [] data, int l, int r, int k)
  ```

  Eine Zufallszahl können Sie mit den Java-Bibliotheken `ThreadLocalRandom` oder `Random` generieren. Mehr Informationen finden Sie in der Java-Documentation dieser Bibliotheken.

- Ihr Programm bekommt wie gewohnt eine Ganzzahl n und eine Liste von Ganzzahlen $a_0, a_1, \ldots, a_{n-1}$ via Standard-In, sowie eine positive Ganzzahl k als Argument. Wie immer können Sie davon ausgehen, dass die Eingabe korrekt ist und müssen den Code nicht um diese Eingabe Robust gestalten. Wenn es in der Liste Duplikate gibt, dann brauchen Sie diese nicht besonders zu behandeln:

  ```
  java B4A2.java 3 <Enter>
  5 1 2 2 3 4 <Enter>
  Input Array: [1, 2, 2, 3, 4]
  The 3-smallest element is 2
  ```