Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Komplexität

186.866 Algorithmen und Datenstrukturen VU

Programmieraufgabe E3

---

1 Übersicht

Ihre Aufgabe ist es einen Algorithmus zu implementieren, der in einem unsortierten Array das k-te Element effizient findet, wobei k ab 0 gezählt wird. Außerdem sollen sie die Laufzeit ihres effizienten Algorithmus mit der der naiven Herangehensweise vergleichen. Beachten Sie, dass bei doppelten Zahlen **jedes Vorkommen einer Zahl gezählt wird**. Das k-te Element ist also äquivalent mit dem Element an der Stelle k im sortierten Array (Siehe Beispiel B).

Beispiel A

Aufgabe: Finden sie das k-te Element
Input: Ein Array $[3, 8, 1, 6, 5, 7, 9]$ und die Zahl $k = 2$
Lösung: 5

Beispiel B

Aufgabe: Finden sie das k-te Element
Input: Ein Array $[1, 3, 3, 1, 4]$ und die Zahl $k = 3$
Lösung: (Die zweite) 3

Das benötigte Framework ("AlgoDat_E3.zip") steht im TUWEL zu Verfügung. Implementieren Sie in der Datei "src/exercise/E3.java" folgende Methoden:

- `sortSelect(Integer[] numbers, int k)`
- `select(Integer[] numbers, int k)`

2 Algorithmus

Beginnen Sie mit der Methode *sortSelect(Integer[] numbers, int k)*, wobei **numbers** ein, im Allgemeinen, unsortiertes Integer-Array ist. Es soll dann der k-te Eintrag gefunden werden (siehe Beispiel). Dabei wird hier, wie in Java üblich, der Arrayindex **bei 0 zu zählen begonnen**. In dieser Methode soll das Problem naiv gelöst werden, d.h. dass Array soll sortiert und anschließend das Element an der k-ten Stelle zurückgegeben werden. Sie dürfen das Sortieren mit einer, von Java zur Verfügung gestellten, Funktion durchführen (zum Beispiel "Arrays.sort()").

Für den effizienten Algorithmus implementieren Sie die Methode *select(Integer[] numbers, int k)*. Hier soll das Problem gelöst werden, indem Sie den aus der Vorlesung bekannten **QuickSort-Algorithmus** so anpassen, dass unnötige Rekursionsaufrufe vermieden werden. Überlegen Sie sich, welcher Abschnitt des Arrays bei einem neuen Rekursionsaufruf noch durchsucht werden muss und welcher vielleicht nicht.

Hinweis: Wenn Sie Ihren effizienten Algorithmus korrekt implementiert haben, sollte es kein Problem sein den naiven Ansatz zu schlagen. Auch wenn dieser eine hoch optimierte Sortierfunktion verwendet!

3 Testen

In der Klasse *E3.java* befinden sich zwei boolean Variablen, die kennzeichnen welche Methoden Sie schon implementiert haben. Wenn Sie diese Variablen auf true setzen wird ihre Implementierung dieser Methode getestet, sonst nicht.

- `SELECT_IS_IMPLEMENTED = true` kennzeichnet, dass die Methode *select* implementiert ist.
- `SORT_SELECT_IS_IMPLEMENTED = true` kennzeichnet, dass die Methode *sortSelect* implementiert ist.

Sie müssen **beide Methoden implementieren um die Aufgabe vollständig zu lösen**. Achten Sie darauf, dass in Ihrer finalen Abgabe beide Variablen auf *true* gesetzt sind.

Wenn sie die Datei "AlgoDat_E3.java" ausführen und eine der vier Instanzen auswählen erhalten sie eine der beiden folgenden Nachrichten:

- Success! Suche ohne Fehler beendet!

  Diese Nachricht bedeutet, dass die ausgewählte Instanz korrekt von allen implementierten Methoden gelöst und eine dementsprechende *.csv* im *solutions* Ordner erzeugt wurde.

- Error! Instanz X nicht korrekt gelöst! Ihre Lösung: NAME: Y

  Diese Nachricht bedeutet sie haben einen Fehler in Instanz Nummer X, ihre Lösung Y war nicht korrekt. Direkt vor ihrem Ergebnis sehen Sie den Namen (*select*/*sortSelect*) der Methode, die das falschen Ergebnis geliefert hat.

4 Auswertung

Wenn Sie die Datei "AlgoDat_E3.java" ausführen, können Sie, wie im Tutorial beschrieben, in der Konsole eine Testinstanz auswählen. Es wird eine *.csv-Datei* im *solutions* Ordner erstellt, welche Sie mit der Datei "E3/Evaluation/plot.html" auswerten können. Sie bekommen dann einen Vergleich der Laufzeiten relativ zur Instanzgröße für die beiden von Ihnen implementierten Varianten präsentiert. **Beachten Sie wie immer, dass die Laufzeiten der sehr kleinen Instanzen nicht aussagekräftig sind**. Vergleichen Sie die Laufzeiten der beiden Varianten und beantworten Sie folgende Fragen:

- Wie haben Sie den QuickSort-Algorithmus angepasst? Beschreiben Sie Ihre Implementierung der *select* Methode.
- Welchen Unterschied sehen Sie zwischen den beiden Implementierungen bezüglich ihrer Laufzeiten? Begründen Sie Ihre Antwort?

Hinweis: Bevor Sie beginnen, lesen Sie sich unbedingt das Tutorial zu den Programmieraufgaben durch („Tutorial.pdf"). Dieses finden Sie im ZIP-Archiv des Programmier-Frameworks im TUWEL. Diese bietet eine kurze Einführung in das verwendete Framework und erklärt Ihnen detailliert was Sie zu tun haben.