

**Algorithmen und Datenstrukturen**  
 SS 2018

**Übungsblatt 5: Sortieren**

Tutorien: 15.05-21.05.2018

**Aufgabe 5-1**     *Quicksort*

Gegeben sei folgendes Array:

37	16	9	42	51	17	71	56	89	39	6	3
----	----	---	----	----	----	----	----	----	----	---	---

- (a) Wenn sie ein Element des unsortierten Arrays finden wollen, wie lange müssen Sie maximal suchen? Begründen Sie ihre Lösung.
- (b) Sortieren Sie das Array mittels Quicksort, nach dem Algorithmus der Vorlesung. Verwenden Sie dabei das letzte Element als Pivot-Element.  
*Machen Sie die einzelnen Schritte bestmöglich nachvollziehbar. Verwenden Sie für jeden Rekursions-schritt eine neue Zeile. Kennzeichnen Sie das jeweils aktuelle Pivot-Element und die bereits an die richtige Position getauschten Pivot-Elemente.*
- (c) Nun da das Array sortiert ist, wie lange brauchen sie im schlechtesten Fall um ein Element zu finden. Was für Strategien gibt es? Zahlt es sich laufzeittechnisch aus zu sortieren, welche Voraussetzungen braucht es, damit sich dies auszahlt?
- (d) Ist dieser Sortieralgorithmus stabil? Begründen Sie.
- (e) Der Algorithmus sei nun an seine obere Laufzeit gestoßen ( $O(n^2)$ ). Unter welchen Umständen kommt es zu diesem Worst Case?
- (f) Was ist die optimale Laufzeit des Sortieralgorithmus und wie sieht das Array aus, wenn diese auftritt?
- (g) Was ist im Allgemeinen eine gute Wahl für das Pivot-Elements?

**Lösungsvorschlag:**

a) Suchzeiten: Bei einem unsortierten Array muss im Worst Case jedes Element betrachtet werden daher suchzeit der Größe n.

b)

37	16	9	42	51	17	71	56	89	39	6	③
③	37	16	9	42	51	17	71	56	89	39	⑥
③	⑥	37	16	9	42	51	17	71	56	89	③⑨
③	⑥	37	16	9	①⑦	③⑨	42	51	71	56	⑧⑨
③	⑥	16	⑨	①⑦	③⑦	③⑨	42	51	71	⑤⑥	⑧⑨
③	⑥	9	①⑥	①⑦	③⑦	③⑨	42	⑤①	⑤⑥	⑦①	⑧⑨
③	⑥	9	16	17	37	39	42	51	56	71	89

### Lösungsvorschlag:

- c) Nun da das Array sortiert ist, wie lange brauchen sie im schlechtesten Fall um ein Element zu finden. Was für Strategien gibt es?

Da das Array sortiert ist, kann man zum Beispiel in der Mitte anfangen und abhängig vom Größenverhältnis nach links (kleiner) oder nach rechts (größer) in die jeweilige Mitte gehen. Dadurch ergibt sich eine Laufzeit von  $\log(n)$ .

Zahlt es sich laufzeittechnisch aus zu sortieren, welche Voraussetzungen braucht es, damit sich dies auszahlt?

Es zahlt sich aus zu sortieren, sobald ein mehrfacher Zugriff auf die Daten stattfindet. Für einen einmaligen Zugriff zahlt sich das Sortieren meist nicht aus. Das Entscheidungskriterium hierfür ist die Differenz zwischen der Laufzeit des Sortierens und der Ersparnis durch eventuelles Sortieren.

- d) Der Sortieralgorithmus ist nicht stabil.

Beispiel:

11<sub>1</sub>, 11<sub>2</sub>, 13, 3, 2, ⑤

2, 11<sub>2</sub>, 13, 3, 11<sub>1</sub>, ⑤

2, 3, 13, 11<sub>2</sub>, 11<sub>1</sub>, ⑤

2, 3, 5, 11<sub>2</sub>, 11<sub>1</sub>, 13

- e) Der Algorithmus stößt an seine obere Laufzeit, wenn das gewählte Pivot-Element immer dem größten oder kleinsten Schlüssel des Datensegments entspricht.

Zum Beispiel wenn eine bereits sortierte Liste vorliegt und als Pivot-Element immer das erste gewählt wird.

- f) Die optimale Laufzeit liegt in  $O(n \log(n))$ . Dann wird das Datensegment immer in zwei gleich große Teile unterteilt.

$$T(n) = \begin{cases} 2 * T(\frac{n-1}{2}) + n - 1, & \text{wenn } n > 1, \\ 0, & \text{wenn } n = 1. \end{cases}$$

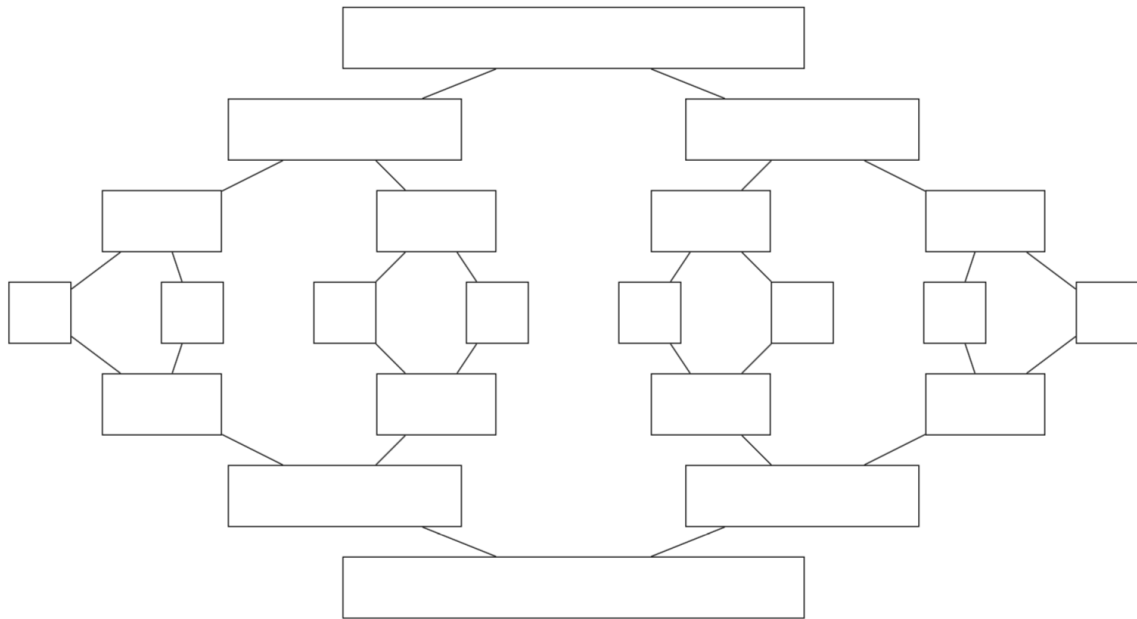
Lösung mit dem Mastertheorem ergibt dann  $O(n \log(n))$ .

- g) Die naiven Ansätze wie das Wählen des ersten, mittleren oder letzten Elements als Pivot-Element sind relativ ineffizient. Bei (teilweise) vorsortierten Listen kann es leicht zum Worst-Case kommen. Eine gute Wahl wäre ein zufälliges Element (sehr geringe Chance, dass der Worst-Case eintritt) oder gar der Median (kein Worst-Case). Es ist relativ aufwendig, den Median eines Datensatzes zu berechnen (dafür muss der Datensatz sortiert sein), deswegen werden häufig drei (am besten zufällige) Elemente zur Bildung dessen verwendet.

### Aufgabe 5-2 MergeSort

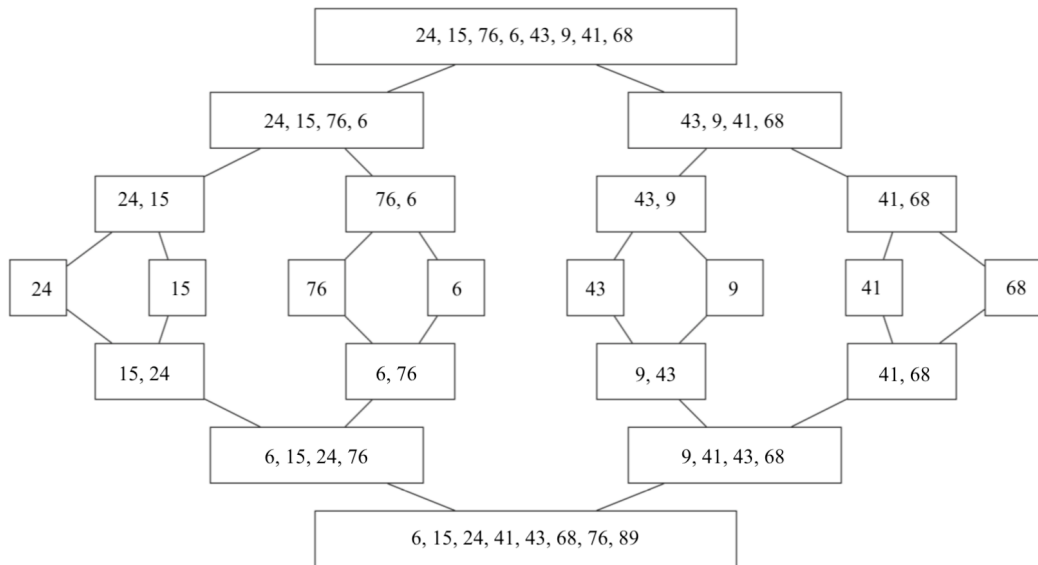
Gegeben sei folgende Liste von Zahlen: 24, 15, 76, 6, 43, 9, 41, 68

- (a) Sortieren Sie die Zahlenfolge mit dem MergeSort-Algorithmus, und geben Sie dabei alle Zwischenschritte an. Verwenden sie dazu die unten gegebene Vorlage.
- (b) Wie viele Tauschoperationen wurden beim Sortieren der Liste ausgeführt?



**Lösungsvorschlag:**

(a) Sortierte Zeichenfolge:



(b) Beim MergeSort werden keine Einträge vertauscht. In jedem "Zwischenschritt" beim Mergen werden die Einträge in der richtigen Reihenfolge in ein neues Array eingetragen.