Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Dennis Hofheinz
Lukas Barth, Lisa Kohl

6. Übungsblatt zu Algorithmen I im SoSe 2016

https://crypto.iti.kit.edu/index.php?id=algo-sose16
{lisa.kohl,lukas.barth}@kit.edu

Aufgabe 1 (*Quicksort,* 6 *Punkte*)

Sortieren Sie die Ziffern Ihrer Matrikelnummer mittels der in den Teilaufgaben angegebenen Varianten von Quicksort. (Falls Sie zu zweit abgeben, genügt eine von beiden.) Wählen Sie als Pivot immer jeweils das letzte Element des Arrays.

a) Verwenden Sie Quicksort mit dem Partitionierungs-Algorithmus *partition* aus der Vorlesung (23.05., Folien 16 und 17). Verwenden Sie das Schema aus der Vorlesung (23.05., Folie 18), markieren Sie insbesondere immer den Wert von i und j.

b) Verwenden Sie Quicksort mit dem Partitionierungs-Algorithmus *dpartition* aus der Übung (25.05., Folie 5). Markieren Sie jeweils den Wert von i und j.

c) Verwenden Sie Quicksort mit Drei-Wege-Partitionierung aus der Vorlesung (23.05., Folien 25 und 26). Markieren Sie jeweils den Wert von i, j und k.

Aufgabe 2 (*Sortieren kleiner Mengen,* 4 *Punkte*)

Geben Sie einen Algorithmus an, der 8 Elemente fast optimal mit maximal $\lceil \log_2 8! \rceil + 1 = 17$ Vergleichen sortiert. Schafft das ein Algorithmus aus der Vorlesung bereits? Geben Sie dafür die minimale und maximale Anzahl Vergleiche von Mergesort, Quicksort und Insertion Sort an.
Hinweis: Als Vergleich zählt nur ein Element-Vergleich, ein Index-Vergleich soll nicht berücksichtigt werden. Es gibt keinen Rekursionsabbruch und nur Zwei-Wege-Vergleiche.

Aufgabe 3 (*Algorithmenentwurf,* 4 *Punkte*)

Gegeben seien n ganze Zahlen im Intervall von 1 bis k. Geben Sie einen Algorithmus an, der für beliebige ganze Zahlen a, b mit $1 \leq a < b \leq k$ in konstanter Zeit berechnen kann, wieviele der n Zahlen im Intervall $[a, b]$ liegen. Der Algorithmus darf $\mathcal{O}(n + k)$ Vorberechnungszeit benötigen. Für Algorithmen mit schlechteren Laufzeiten werden gegebenenfalls noch Teilpunkte vergeben.

Aufgabe 4 (*Mehrfach-Selektion,* $3 + 1$ *Punkte*)

Gegeben sei eine (unsortierte) Menge M von n Elementen sowie eine totale Ordnungsrelation $<$ auf M (die Ordnungsrelation sei in konstanter Zeit auswertbar).

a) Skizzieren Sie einen vergleichsbasierten Algorithmus, der die Elemente mit den Rängen[1] 1, 3, 9, 27, …, $3^{\lfloor \log_3 n \rfloor}$ berechnet und dafür *insgesamt* (also nicht pro Element!) im schlimmsten Fall $\mathcal{O}(n)$ Zeit benötigt.

   Hinweis: Sie dürfen eine Funktion select(M, k) verwenden, die das Element mit Rang k einer Menge M zurückgibt, und zwar im schlechtesten Fall in $O(|M|)$ Zeit.

b) Begründen Sie kurz, wieso Ihr Algorithmus das gewünschte Laufzeitverhalten aufweist.

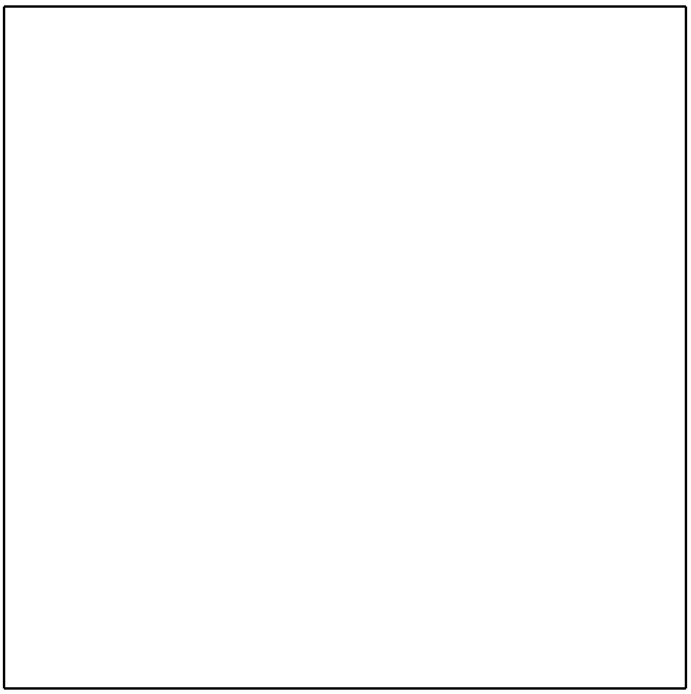
Ausgabe: Mittwoch, 25.5.2016
Abgabe: Freitag, 03.06.2016, 12:45 im Briefkasten im Untergeschoss von Gebäude 50.34

[1] Für eine endliche totalgeordnete Menge $(M, <)$ hat ein Element $x \in M$ genau dann den *Rang* $i \in \mathbb{N}$, wenn $m_1 < m_2 < \cdots < x = m_i < \cdots < m_n$ gilt für $M = \{m_1, \ldots, m_n\}$.

Deckblatt Übungsblatt 6
Algorithmen I

A1	**A2**	**A3**	**A4**	$\sum$

Tutoriumsnummer:



Name	**Matrikelnummer**	**Unterschrift**

Mit unseren Unterschriften bestätigen wir, dass die Aufgaben von den Unterzeichnern eigenständig gelöst worden sind.

Hinweis: Das Übungsblatt darf in Gruppen von bis zu zwei Personen bearbeitet werden. Beide Personen müssen demselben Tutorium zugeteilt sein. Möchte jemand seine Abgaben-Gruppe innerhalb des Semesters wechseln, so ist dies im Voraus mit dem Tutor abzusprechen. **Bitte tragen Sie in das obere Quadrat *groß* die Nummer Ihres Tutoriums ein.** Die Lösung des Übungsblattes ist in jedem Fall mit diesem Deckblatt abzugeben.