Universität Klagenfurt
Informatik – Systemsicherheit
M. Morak · R. Wigoutschnigg

UE Algorithmen und Datenstrukturen
SS 2019
Übungstermine: siehe ZEUS

Übungsblatt 3

Aufgabe 3.1: Elementare Sortierverfahren

a) Betrachten Sie die elementaren Sortierverfahren *Selection Sort*, *Insertion Sort* und *Bubble Sort*: Welches Verfahren läuft am schnellsten für ein Array, in dem alle Schlüssel identisch sind? Um wie viel ist dieses Verfahren schneller als das langsamste der drei genannten Verfahren?
b) Beweisen oder widerlegen Sie: Um ein beliebiges Array der Länge 10 zu sortieren sind 15 Vergleiche ausreichend.

Aufgabe 3.2: Analyse von Quicksort

Im Mittel benötigt Quicksort $O(n \cdot log(n))$ Vergleichsoperationen, um ein Array der Länge n zu sortieren.
a) Bestimmen Sie die Komplexitätsordnung für den schlechtesten Fall. Wann tritt dieser Fall ein?
b) Wie verhält sich der Algorithmus, wenn alle Elemente denselben Schlüssel besitzen? Wie, wenn das Array umgekehrt sortiert ist?
c) Bei einer Variante von Quicksort werden drei Elemente eines Intervalls gewählt, wobei das Mittlere als Pivotelement genutzt wird (median-of-three). Diskutieren Sie diese Variante.

Aufgabe 3.3: Heapaufbau

Gegeben sei das ganzzahlige Array $a = (10, 14, 8, 11, 7, 4, 9, 1, 7, 3, 15, 2, 6, 12)$.
a) Stellen Sie das Array a als Binärbaum dar.
b) Welche Vertauschungen sind im Binärbaum von a) nötig, um die Heapeigenschaft $H(3, 14)$ zu erzeugen? Geben Sie den resultierenden Binärbaum an.
c) Führen Sie den Heapaufbau zu Ende und stellen Sie den resultierenden Binärbaum mit der Eigenschaft $H(1, 14)$ dar.
d) Wenden Sie den Heapsort-Algorithmus auf das angegebene Array an.
e) Wie ist der Heapsort-Algorithmus abzuändern, damit das Array a absteigend sortiert wird?

Aufgabe 3.4: Heapeigenschaften

Gegeben ist ein Array a der Länge n.
a) Zeige: $(i \leq i' \leq j' \leq j, H(i, j)) \Rightarrow H(i', j')$
b) Zeige: $H(1, j) \Rightarrow a[1] \geq a[k]$ für $1 \leq k \leq j$
c) Für welche Werte n ist $H(\lceil n/2 \rceil, n)$ ein Heap?
d) Gibt es eine Belegung für a, sodass die Heap-, als auch die Min-Heap-Eigenschaft gilt.

Bitte wenden!

Aufgabe 3.5: Heapsort-Analyse

Heapsort baut aus dem Array a der Länge n zuerst einen Heap $H(1,n)$ auf. Bezeichnet $t = \lfloor log_2(n) \rfloor$ die Tiefe des Heaps in Baumdarstellung, so „versickern" nacheinander alle Elemente der Tiefe k für $k = t-1, t-2, \ldots, 0$ im Heap. Die Elemente der Tiefe k sinken dabei maximal bis in die Tiefe t. Zeigen Sie, dass

a) der Heapaufbau bei Heapsort(a, n) nur $O(n)$ Vergleiche benötigt.
   Hinweis:
$$\sum_{k=0}^{\infty} k \cdot x^k = \frac{x}{(1-x)^2} \qquad f\ddot{u}r\ |x| < 1$$

b) die Prozedur BuildHeap(a, i, j) nur $O(log(j) - log(i))$ Vergleiche benötigt.

c) Heapsort(a, n) in jedem Fall $O(n \cdot log(n))$ Vergleiche benötigt.