Prof. Dr. Ir. Joost-Pieter Katoen

Jonathan Heinen, Sabrina von Styp

Hinweise:

- Die Übungsblätter sollen in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung bearbeitet werden.
- Die Lösungen müssen bis Montag, den 31. Mai um 11:00 Uhr in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Namen und Matrikelnummern der Studenten sowie die Nummer der Übungsgruppe sind auf jedes Blatt der Abgabe zu schreiben. Heften bzw. tackern Sie die Blätter!

Aufgabe 1 (Min- und Max-Heaps): (4+4 Punkte)

a) Bestimmen Sie, ob folgende Arrays Heaps (Max-Heaps) sind. Falls nicht, geben Sie an wo die Heapeigenschaft verletzt ist.

1.)

56	47	56	10	20	50	51	1	2	3	18

2.)

56	56	47	10	20	50	51	1	2	3	18

3.)

56	47	56	10	51	20	50	1	2	3	18

4.)

56	47	56	10	5	20	50	1	2	3	18

b) In der Vorlesung wurden so genannte Max-Heaps vorgestellt. D.h jedes Element ist größer/gleich seiner Kinder. Ein *Min-Heap* ist ein Heap bei dem jedes Element kleiner/gleich seiner Kinderelemente ist. Bestimmen sie, ob die folgenden Arrays Min-Heaps sind, und falls nicht, geben sie an, wo die Heapeigenschaft verletzt ist

1.)

1	5	1	8	10	4	15	11	12	14	11

2.)

0	1	5	8	10	4	15	11	12	14	11

3.)

1	5	1	11	8	4	15	11	12	14	10

4.)

1	5	1	11	15	4	8	11	12	14	10

Aufgabe 2 (Insertionsort): (5 Punkte)

Nutzen Sie Insertionsort, um die Schlüsselwerte des folgenden Arrays aufsteigend zu sortieren. Geben Sie den Zustand des Arrays nach jeder Einfügeoperation an, sowie die Anzahl der Schlüsselwert-Vergleiche und Array-Zuweisungen, die für den Sortierschritt benötigt wurden.

1	8	5	9	0	2	3	4	7	6

Aufgabe 3 (Heapsort): (5 + 8 Punkte)

Das folgende Array soll mit Hilfe des Heapsort-Algorithmus sortiert werden:

5	3	4	7	0	2	3	6	1	8

a) Stellen Sie das gegeben Array als Baum dar und skizzieren Sie (durch nummerierte Pfeile) die nötige Versickerungen. Geben Sie den resultierenden Heap sowohl als Baum als auch als Array an.

b) Nutzen Sie Heapsort um, ausgehend von dem Heap aus Aufgabenteil **a)**, das gegebene Array zu sortieren. Geben Sie für jeden Sortierschritt den neuentstandenen Heap als Array sowie als Baum an und skizzieren Sie nötige Versickerungen in der Baumdarstellung. Geben Sie auch die Reihenfolge der Versickerungen an.

Aufgabe 4 (Ein weiterer Sortieralgorithmus): (2+2+5+5 Punkte)

Gegeben sei der folgende Sortieralgorithmus:

```
void sort(int E[]) {
  int i,j,m;
  for (i = 0; i < E.length; i++) {
    m = i;
    for (j = i + 1; j < E.length; j++) {
      if(E[j] <= E[m]){
          m = j;
      }
    }
    int v = E[i];
    E[i] = E[m];
    E[m] = v;
  }
}
```

a) Geben Sie in wenigen Worten wieder, wie der gegebene Algorithmus funktioniert. In der Vorlesung wurden mehrere Sortierverfahren genannt (siehe Folien). Welcher Name passt zu diesem Algorithmus?

b) Ist der Sortieralgorithmus stabil, oder kann er so angepasst werden, dass er stabil wird?

c) Nutzen Sie den gegebene Algorithmus, um das folgende Array zu sortieren. Geben Sie den Zustand des Arrays nach jedem Durchlauf der äußeren Schleife an.

1	3	2	7	0	4	8	5	7	6

d) Welche Average-Case Laufzeit besitzt der gegebene Sortieralgorithmus für eine Eingabe der Länge n? Geben Sie die Komplexitätsklasse $\Theta(T_{sort(E)})$ für Array E mit Länge `n = E.length` an und begründen Sie Ihre Antwort.