Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Jonathan Heinen, Thomas Ströder, Sabrina von Styp

Hinweise:

- Die Übungsblätter sind in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung zu bearbeiten.
- Die Lösungen müssen bis Montag, den 21. 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 (Sortieralgorithmus): (5 + 4 + 1 + 3 Punkte)

In dieser Aufgabe werden sie einen neuen Sortieralgorithmus implementieren. Dieser verfolgt, ebenso wie Quicksort, den Divide-and-Conquer Ansatz. Die Aufteilung in Teilprobleme erfolgt in diesem Fall durch einer Abwandlung des Dutch-National-Flag Algorithmus aus Vorlesung sieben.
Der Sortieralgorithmus ist in Java zu implementieren. Im L2P finden Sie ein Klassengerüst, das zu benutzen ist. Bitte kommentieren Sie Ihre Lösung und schicken Sie den Java Code an Ihren Tutor.

a) Implementieren Sie, basierend auf dem Dutch-National-Flag Algorithmus, einen Partitionierung Algorithmus `IntTuple partion(int[] E, int left, int right, int red, int blue)`, der für zwei gegebene Pivotelemente `red`, `blue` mit `red` $\leq$ `blue`, das gegebene Array `E` in dem Bereich zwischen `left` und `right` in drei Teile partitioniert. Die Partionierung soll so realisiert werden, dass der linke Teil die Elementen enthält, die kleiner sind als die beiden Pivotelemente, sowie der rechte Teil die Elementen die größer sind als die beiden Pivotelemente. Der mittlere Teil bleibt den restlichen Elementen vorbehalten. Die Grenzen der einzelnen Bereiche werden als `IntTuple` zurückgegeben.

b) Implementieren Sie nun die Methode `void sort(int[] E)`, die das gegebenes Array `E` sortiert. Wählen Sie hierzu zwei Pivotelemente und teilen Sie das Array, entsprechend dem Algorithmus aus **a)** in drei Partionen. Sortieren Sie dann rekursiv jede der drei Partionen. Achten Sie darauf, dass der Algorithmus stets terminiert.

c) Ist der von Ihnen implementierte Algorithmus stabil? Begründen Sie Ihre Antwort.

d) Geben Sie die Worst-Case sowie Best-Case Laufzeit Ihres Algorithmus an. Begründen Sie Ihre Antwort.

Aufgabe 2 (Höhergradige Heaps): (3+2+4+3 Punkte)

In Übung 5 haben Sie Heapsort, basierend auf *binären Heaps*, zum Sortieren genutzt. In binären Heaps hat jedes Element bis zu zwei Kinderelemente. Das Konzept der binären Heaps lässt sich auf *d-Heaps* erweitern. Ein *d*-Heap ist ein Baum mit Verzweigungsgrad *d* (d.h. jedes Element besitzt bis zu *d* Kinder), der die *max-Heapeigenschaft* (alle Kinderelemente haben *niedrigere* Werte) erfüllt.

a) Geben Sie die Formel an mit der, für die Arraydarstellung eines *d*-Heaps, die Position der Kinder des Elementes an Position *i* bestimmt werden kann.

In welchem Bereich des Arrays ist die Heapeigenschaft immer erfüllt?

b) Stellen Sie den im folgenden Array repräsentierten *3-Heap* als Baum dar.

40	32	24	36	28	30	5	12	21

c) Geben Sie eine modifizierte Version des in der Vorlesung vorgestellten Algorithmus `sink` an, der Elemente in einem *d*-Heap versickern lässt. Der Grad *d* wird hierbei als Parameter übergeben.

d) Sortieren Sie das in **b)** gegebene Array entsprechend der *Heapsortmethode für 3-Heaps*.

Aufgabe 3 (Komplexitätsanalyse): (3 + 3 Punkte)

a) Bestimmen Sie für den folgenden Code die asymptotische Laufzeit Θ für den Aufruf `berechne(n,m)` in Abhängigkeit von den Parametern n und m (für $n, m \geq 0$). Begründen Sie Ihre Antwort und geben Sie Zwischenschritte an.

```
int berechne(int n, int m){
        int k = m;
        while(n > 0){
            k = k * m;
            n--;
        }
        for(int i = k; i > n; i--){
            k = k - m;
            n++;
        }

        return k;
}
```

b) Bestimmen Sie für den folgenden Code die asymptotische Laufzeit Θ für den Aufruf `berechne(n, k)` in Abhängigkeit von den nicht negativen Parametern n und k (für $n, k \geq 0$). Begründen Sie Ihre Antwort und geben Sie Zwischenschritte an.

```
int berechne(int n, int k){
    if(n == 0)
        return k;

    int value = berechne(n/3, k);

    if(n < 42){

        for(int i = 0; i < k*k; i++)
            value += i * n;

        value += berechne(n/3, k) + 2;

    }else{

        for(int i = 0; i < n; i++)
            value += i * n;

        value += 3 * berechne(n/3, k) + 3;
    }

    return value;

}
```