Prof. Dr. Klaus Volbert

Regensburg, den 30. Oktober 2019

Votieren und abgeben bis zum 05. November 2019, 23:00 Uhr

Übungen zur Vorlesung

Algorithmen und Datenstrukturen

WiSe 2019/20

Blatt 5

Wichtige Hinweise:

- > Falls Sie bei der Bearbeitung einer Aufgabe größere Schwierigkeiten hatten und deswegen die Bearbeitung abgebrochen haben, so versuchen Sie bitte Ihre Schwierigkeiten in Form von Fragen festzuhalten. Bringen Sie Ihre Fragen einfach zur Vorlesung oder zur Übung mit!
- > Kursraum: https://elearning.uni-regensburg.de/course/view.php?id=9228

Aufgabe 1:
Demonstrieren Sie die Funktionsweise von Merge Sort und Heap Sort anhand des Feldes $a[] = \{-5, 13, -32, 7, -3, 17, 23, 12, -35, 19\}$. Führen Sie die Demonstration von Merge Sort anhand eines Rekursionsbaums durch, indem Sie den Baum in der Phase „Teilen“ von oben nach unten aufbauen und anschließend in der Phase „Mischen“ von unten nach oben durchlaufen. Nennen Sie jeweils die entstandenen Teilfolgen. Führen Sie bei der Demonstration von Heap Sort sowohl entstehende Bäume als auch entstehende Felder mit. Überprüfen Sie Ihre Ausführungen mit Hilfe eines C, C++, C# oder Java-Programms, das die beiden Sortierverfahren implementiert und die wesentlichen Informationen ausgibt.

Aufgabe 2:
Wandeln Sie den iterativen InsertionSort Algorithmus aus der Vorlesung in einen rekursiven und den rekursiven MergeSort Algorithmus aus der Vorlesung in einen iterativen um. Analysieren Sie jeweils die Laufzeit Ihrer Algorithmen.

Aufgabe 3:
Zeigen Sie folgende Aussagen:

1. Ein Heap mit n Elementen hat die Höhe $\lfloor \log n \rfloor$
2. Ein Heap mit n Elementen hat höchstens $\lceil \frac{n}{2^{h+1}} \rceil$ viele Knoten der Höhe h
3. Für alle x mit $|x| < 1$: $\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$

   (Tipp: Differenzieren Sie beide Seiten der geometrischen Reihe und multiplizieren Sie mit x)

Kann die Reihenfolge der Heapify-Aufrufe in der Operation BuildHeap ohne weitere Modifikation vertauscht werden? Begründen Sie Ihre Antwort!

Aufgabe 4:
Entwickeln Sie einen Algorithmus mit Laufzeit $\Theta(n \log n)$, der folgende Spezifikation erfüllt:

- Eingabe: $a[] = \{a_0, \ldots, a_{n-1}\}, s$ mit $a_i \in \mathbb{Z}, s \in \mathbb{Z}, n \in \mathbb{N}$ ($n+1$ ganze Zahlen)
- Ausgabe: `true`, falls es zwei Elemente a_i, a_j gibt mit $s = a_i + a_j, i \neq j$, `false` sonst

Implementieren und testen Sie Ihren Algorithmus in C, C++, Java oder C#.