Prof. Dr. Klaus Volbert

Regensburg, den 31. Oktober 2018

Votieren und abgeben bis zum 06. November 2018, 23:00 Uhr

Übungen zur Vorlesung
Algorithmen und Datenstrukturen
WiSe 2018/19
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:
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#.

Aufgabe 2:
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 3:
Stellen Sie sich vor, Sie sollen zwei quadratische Matrizen $M, N \in \mathbb{R}^{n \times n}$ miteinander multiplizieren. Sei $n = 2^i$ für ein $i \in \mathbb{N}$, dann kann man M, N und $O = M \cdot N$ wie folgt zerlegen mit $M_{ij}, N_{ij}, O_{ij} \in \mathbb{R}^{n/2 \times n/2}$ für $i, j \in \{1, 2\}$:

$$M := \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, N := \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{pmatrix}, O := \begin{pmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{pmatrix}$$

Zeigen Sie, dass die beiden folgenden Varianten die Produktmatrix O korrekt berechnen:

- Variante 1:

$$\begin{aligned} O_{11} &:= M_{11} \cdot N_{11} + M_{12} \cdot N_{21} \\ O_{12} &:= M_{11} \cdot N_{12} + M_{12} \cdot N_{22} \\ O_{21} &:= M_{21} \cdot N_{11} + M_{22} \cdot N_{21} \\ O_{22} &:= M_{21} \cdot N_{12} + M_{22} \cdot N_{22} \end{aligned}$$

- Variante 2:

$$\begin{aligned} H_1 &:= (M_{11} + M_{22}) \cdot (N_{11} + N_{22}) \\ H_2 &:= (M_{21} + M_{22}) \cdot N_{11} \\ H_3 &:= M_{11} \cdot (N_{12} - N_{22}) \\ H_4 &:= M_{22} \cdot (N_{21} - N_{11}) \\ H_5 &:= (M_{11} + M_{12}) \cdot N_{22} \\ H_6 &:= (M_{21} - M_{11}) \cdot (N_{11} + N_{12}) \\ H_7 &:= (M_{12} - M_{22}) \cdot (N_{21} + N_{22}) \end{aligned}$$

$$\begin{aligned} O_{11} &:= H_1 + H_4 - H_5 + H_7 \\ O_{12} &:= H_3 + H_5 \\ O_{21} &:= H_2 + H_4 \\ O_{22} &:= H_1 - H_2 + H_3 + H_6 \end{aligned}$$

Bestimmen Sie die asymptotische Laufzeitkomplexität beider Varianten und vergleichen Sie diese mit der Komplexität der Standardmethode zur Multiplikation zweier Matrizen.

Aufgabe 4:
Implementieren Sie eine verbesserte Variante für Count Sort mit Laufzeit $T(n) = 2k + 3n$, wenn n ganze Zahlen aus dem Wertebereich $\{0, \ldots, k\}$ eingegeben werden können. Vergleichen Sie Heap Sort, Count Sort und Map Sort anhand von zufällig, gleichverteilt erzeugten Feldern unterschiedlicher Größe unter der Annahme, dass die Zufallszahlen im Wertebereich $\{1000, \ldots, 10000\}$ liegen. Stellen Sie dar, bei welchen Größen welcher Algorithmus die beste Laufzeit liefert. Betrachten Sie sowohl die Anzahl der ausgeführten Operationen und Vergleiche, als auch die real verbrauchte Rechenzeit.