Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Anna Beer, Florian Richter

München, 22.05.2018

Algorithmen und Datenstrukturen

SS 2018

Übungsblatt 6: Sortieren

Tutorien: 23.05-25.05.2018

Aufgabe 6-1 *Heapsort*

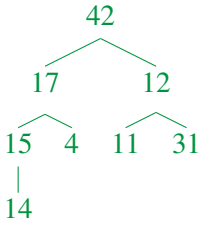
Sei A = [42, 17, 12, 15, 4, 11, 31, 14]

a) Welche Elemente verletzten die Heap-Eigenschaft? Geben sie Alternativwerte an, sodass die Heap-Eigenschaft erfüllt ist.

b) Sortieren sie das unveränderte Array A mittels Heapsort. Geben sie die Zwischenschritte als Binärbäume an.

Lösungsvorschlag:

a) Die 31 ist falsch, sie muss kleiner 12 sein. Z.B. 10.



b) Start mit A = [42, 17, 12, 15, 4, 11, 31, 14]

Sortieren (42 und 14 vertauschen) ⇒ A = [14, 17, 31, 15, 4, 11, 13, 42]

Sortieren (12 und 31 vertauschen) ⇒ A = [12, 17, 14, 25, 4, 11, 12, 31, 42]

Sortieren (11 udn 17 vertauschen) ⇒ A = [11, 15, 14, 12, 4, 17, 31, 42]

Sortieren (15 und 4 vertauschen) ⇒ A = [4, 12, 14, 11, 15, 17, 31, 42]

Sortieren (14 und 11 vertauschen) ⇒ A = [11, 12, 4, 14, 15, 17, 31, 42]

Sortieren (12 und 4 vertauschen) ⇒ A = [4, 11, 12, 14, 15, 17, 31, 42]

Sortieren (11 und 4 vertauschen) ⇒ A = [4, 11, 12, 14, 15, 17, 31, 42]

Aufgabe 6-2 *Vergleich zwischen Sortieralgorithmen*

Sortieren Sie das Array $A = [42, 17, 12, 15, 4, 11, 31, 14]$ und vergleichen Sie die angewendeten Sortierverfah-

ren bzgl. der Anzahl der Vergleiche.

(a) Benutzen Sie QuickSort mit folgenden zwei Pivotstrategien: Letztes Arrayelement bzw. Median aus den Elementen an Positionen 1, $\lfloor \frac{n}{2} \rfloor$ und n (Position 1, falls $n \leq 3$). Markieren Sie alle Pivot-Elemente. Ein-elementige Teillisten enthalten kein Pivot-Element.

(b) Sortieren Sie außerdem mit SelectionSort und InsertionSort.

Lösungsvorschlag:

- QuickSort, Pivot = letztes Element
  12,4,11,*14*,42,17,15,31
  4,*11*,12,*14*,17,15,*31*, 42
  4,*11*,12,*14*,*15*,17,*31*, 42

- QuickSort, Pivot = Median-of-three
  Wir nehmen den Median der Elemente an den Positionen 1, $\lfloor \frac{n}{2} \rfloor$ und n, das heißt den Median von 42, 15 und 14. Dieser Median ist 15. Damit ist 15 im ersten Schritt das Pivotelement.
  Das Teilarray, das auf der rechten Seite entsteht, ist von der Länge 3, das heißt das Pivotelement ist das erste Element des Teilarrays (die 42).
  12,4,11,14,[15],*42*,17,31 (+7)
  4,11,[12,14,15],*17*,31,[42] (+3+2)
  4,11,12,14,15,17,31,42 (+1+1)

- SelectionSort
  42,17,12,15,4,11,31,14 (+7)
  4,17,12,15,42,11,31,14 (+6)
  4,11,12,15,42,17,31,14 (+5)
  4,11,12,15,42,17,31,14 (+4)
  4,11,12,14,42,17,31,15 (+3)
  4,11,12,14,15,17,31,42 (+2)
  4,11,12,14,15,17,31,42 (+1)
  4,11,12,14,15,17,31,42

- InsertionSort
  42,17,12,15,4,11,31,14 (+0)
  42,17,12,15,4,11,31,14 (+1)
  17,42,12,15,4,11,31,14 (+2)
  12,17,42,15,4,11,31,14 (+3)
  12,15,17,42,4,11,31,14 (+4)
  4,12,15,17,42,11,31,14 (+5)
  4,11,12,15,17,42,31,14 (+2)
  4,11,12,15,17,31,42,14 (+5)
  4,11,12,14,15,17,31,42

QuickSort, Pivot=last: 13
QuickSort, Pivot=median-of-three: 14
Selection: 28
InsertionSort: 22

Aufgabe 6-3 *Hybride Sortierverfahren*

Überlegen sie in allen Aufgabenteilen qualitativ. Sie müssen keine streng mathematisch korrekten Herleitungen formulieren. Verwenden sie außerdem die aus der Vorlesung bekannten Laufzeiten

(a) Oracles Java Implementierung implementiert die Methode $Array.sort(int[])$ als sogenannten Dual-Pivot Quicksort Algorithmus (zumindest in Version 7 und 8). Wie zuvor wird bei kleinen Arrays Insertion-Sort benutzt, ansonsten wird mit zwei Pivot-Elementen das Array dreigeteilt und anschließend rekursiv fortgefahren. Geben sie die Laufzeit für den Best- und Worst-Case an.

(b) Timsort wird in der Python Implementierung CPython benutzt. Dieser Algorithmus setzt sich aus Mergesort und Insertionsort zusammen. Timsort identifiziert in einem ersten Durchlauf Folgen von aufsteigenden, bzw. strikt absteigenden Elementen genannt *runs*. Kleine *runs* werden dann mit Insertion-Sort gemerged, große mit Merge-Sort.

i) Wie bewerten sie die Entscheidung zunächst die *runs* zu identifizieren?

ii) Geben sie die Laufzeit für den Best- und Worst-Case an.

Lösungsvorschlag:

a) Rekursionsgleichung für den Best-Case $T(n) = 3 \cdot T(\frac{n-2}{3}) + f_b(n)$. Rekursionsgleichung für den Worst-Case (nur die Pivot Elemente werden in jedem Schritt korrekt einsortiert): $T(n) = T(n-2) + f_w(n)$. Es reicht zu erkennen, dass $f_b(n), f_w(n) \in \mathcal{O}(n)$ um das richtige Ergebniss zu erhalten. Es folgt nach dem Mastertheorem für den Worst Case $\mathcal{O}(n^2)$ und den Best Case $\mathcal{O}(n \cdot log(n))$.
Etwas ausführlicher: Für den Best Case hat man 1 Vergleich zwischen den beiden Pivot-Elementen, dann wird jedes der verbleibenden Elemente mit beiden Pivots verglichen, wobei der zweite Vergleich nicht immer ausgeführt werden muss. Im Idealfall wird gedrittelt. 1/3 der verbleibenden $n-2$ Elemente muss nur mit einem Pivot-Element Verglichen werden (kleiner dem kleineren), 1/3 auf jeden Fall mit beiden (zwischen den beiden), und das letzte 1/3 könnte zwar theoretisch nur mit dem größeren Verglichen werden, aber man muss sich bereits in der Implementierung festlegen, ob man zuerst mit dem kleineren oder erst mit dem größeren Vergleicht. Hier wurde zuerst der Vergleich mit dem kleineren Pivot-Element gewählt, d.h. alle Elemente größer dem größeren Pivot werden auch mit beiden Pivot-Elementen Verglichen. Macht also in Summe $1 + 1/3 \cdot (n-2) + 2 \cdot 1/3(n-2) + 2 \cdot 1/3(n-2) = 1 + 5/3 \cdot (n-2) = f_b(n)$ Vergleiche. Für den Worst Case folgen dann $f_w(n) = 2 \cdot (n-2) + 1$. +1 wegen des Vergleichs der beiden Pivotelemente, und $2 \cdot (n-2)$ Vergleiche der verbleibenden Elemente mit beiden Pivot-Elementen.
b)
i) Dies erscheint sinnvoll, da in der Praxis komplett ungeordnete Daten selten vorkommen. Auch kann das Merging beschleunigt werden, wenn bekannt ist, das ein *run* absteigend ist.
ii) Best-Case: Bereits sortiert: Dies wird im ersten Durchlauf zur Identifizierung der *runs* erkannt $\Rightarrow \mathcal{O}(N)$. Worst Case: Wie Mergesort $\mathcal{O}(N \cdot log(N))$. Insertion-Sort kann vernachlässigt werden, da er stets für eine konstant lange Teilliste aufgerufen wird und dann in O(1) läuft.