Datenstrukturen und Algorithmen, SS 2024
Prof. Dr. P. Rossmanith
D. Cloerkes, M. Gehnen, D. Mock, N. Speetzen

Theoretische Informatik | RWTH AACHEN UNIVERSITY

Abgabe: 16.05.2024

Übungsblatt 05

Aufgabe T15

Gegeben ist folgendes Array: $[8, 6, 3, 7, 4, 2, 20, -45]$.

(a) Wie viele Inversionen hat es?

(b) Wie viele Läufe hat es?

(c) Was macht Quicksort in der ersten Partitionierungsphase daraus?

(d) Was macht Mergesort in der letzten Mischphase?

(e) Wie sieht das Array nach der Konstruktion des Max-Heaps aus, wenn Heapsort angewandt wird? Wie siehte es nach der ersten Bubble-Operation aus?

Aufgabe T16

Stellen Sie sich vor, Sie befinden sich in folgender Situation: Sie wollen ein Terabyte (1024 Gigabyte) an Strings aufsteigend ordnen. Alle Strings sind genau 16 Byte groß. Dafür steht Ihnen ein Computer mit fünf Festplatten, die jeweils ein Terabyte groß sind, zur Verfügung. Eine der Festplatten enthält die Strings und ist *read-only*. Die weiteren vier sind frei benutzbar. Man kann von verschiedenen Festplatten parallel schreiben/lesen. Nehmen Sie an, der Computer hat 16 Gigabyte an Arbeitsspeicher, den Sie frei benutzen können.

(a) Wie sortiert man die Strings am besten, sodass am Ende eine der Festplatten eine Liste aller Strings in aufsteigender Reihenfolge enthält?

(b) Gehen Sie davon aus, dass t_1 die Zeit ist, die man braucht, um 16GB an Strings zu sortieren, t_2 um ein Terabyte von einer Festplatte zu lesen, und t_3 die Zeit, um ein Terabyte auf eine Festplatte zu schreiben. Geben Sie eine möglichst gute Abschätzung der Laufzeit Ihres Verfahrens mit Hilfe dieser Werte.

(c) Was denken Sie sind realistische Werte für t_1, t_2 und t_3?

Aufgabe T17

Beweisen Sie, dass es keinen vergleichsbasierten Sortieralgorithmus gibt, welcher ein beliebiges Array der Größe fünf mit nur sechs Vergleichen sortieren kann.

Aufgabe T18

Wir untersuchen in dieser Aufgabe, wie man binäre Suchbäume zum Sortieren verwenden kann.

(a) Entwerfen Sie einen Algorithmus, der in $O(n)$ Schritten die n Schlüssel aus einem binären Suchbaum in aufsteigender Reihenfolge ausgeben kann.

(b) Beschreiben Sie, wie man mit binären Suchbäumen möglichst effizient sortieren kann.

(c) Wie schnell ist Ihr Verfahren aus (b) im worst-case mit gewöhnlichen binären Suchbäumen, mit AVL-Bäumen und mit Splay-Bäumen?

(d) Beantworten Sie Frage (c) auch für den Spezialfall, dass die Eingabe bereits sortiert bzw. umgekehrt sortiert ist.

Aufgabe H16 (8+8+8 Punkte)

(a) Sortieren Sie das folgende Array mithilfe von Quicksort aus der Vorlesung. Geben Sie dazu das Array nach jeder Partition-Operation an und markieren Sie das Pivot-Element und den Bereich, auf dem die Partition-Operation angewandt wurde.

$$[6, 8, 3, 9, 2, 1, 4, 5, 7]$$

(b) Sortieren Sie das folgende Array mithilfe von Mergesort aus der Vorlesung. Geben Sie dazu das Array nach jeder Merge-Operation an und markieren Sie den Bereich, auf dem die Merge-Operation angewandt wurde.

$$[3, 5, 2, 6, 1, 7, 4]$$

(c) Sortieren Sie das folgende Array mithilfe von Heapsort aus der Vorlesung. Geben Sie dazu das Array nach der jeder Bubble-Operation an. Sie können zusätzlich den Heap (der übrigen Elemente) grafisch angeben.

$$[3, 5, 2, 6, 1, 7, 4]$$

Aufgabe H17 (10 Punkte)
Armin und Christoph unterhalten sich bei einem Glas Absinth über das Sortieren von Strings.[1] Voller Stolz und Freude teilt Armin seine Idee mit Christoph:

> Ein Algorithmus linearer Zeit,
> Strings der Länge n in Gesamtheit.
> In einen Trie sie passen,
> Sortiert sie ausspucken lassen,
> Zum MIT: nun eine echte Möglichkeit!

Doch Christoph scheint nicht überzeugt:

> Theorie deutlich,
> $n \log n$, unüberwindbar,
> Algorithmus fort.

Können Sie aufklären, wer, Armin oder Christoph, recht hat?
Gegeben seien Strings beliebiger Länge (über dem Alphabet $\{0, 1\}$), getrennt durch ein Trennzeichen (z.B. \$), sodass die Gesamtlänge n ist. Kann man diese Strings in Zeit $O(n)$ sortieren, indem man sie in einen anfangs leeren Trie hinzufügt und im Anschluss passend ausgibt?
Wenn ja, begründen Sie die Laufzeit von $O(n)$, geben Sie an, wie aus dem Trie die sortierte Folge in linear Zeit ausgegeben werden kann und argumentieren Sie, wieso diese Laufzeit nicht der unteren Schranke aus der Vorlesung fürs Sortieren widerspricht. Wenn nein, zeigen Sie, wo dieses Verfahren mehr als $O(n)$ Zeit braucht oder fehlerhaft ist.

Aufgabe H18 (6 Punkte)
Stellen Sie sich vor, dass ein sortiertes Array der Länge n durch $\log \log n$ Vertauschungen benachbarter Elemente verändert wird.

(a) Wie sortieren Sie das entstandene Array möglichst schnell? Können Sie eine Laufzeit von $\Omega(n \log n)$ vermeiden?

(b) Wie viele Vertauschung verkraftet ihre Methode, bevor sie nicht mehr schneller als der allgemeine Fall von $\Theta(n \log n)$ Zeit ist?

[1] Dialog unterstützt durch ChatGPT.