Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2025-11-14
Version: 2025-11-18 21:21

Übungsblatt 5 zur Vorlesung Effiziente Algorithmen (Winter 2025/26)

Abgabe: *Bis 2025-11-21 18:00, on ILIAS.*

1. Aufgabe

20 + 20 + 20 Punkte

Gegeben sei folgende Beschreibung des *Partial Sorting* Problems:

> Gegeben sei ein Array $A[0..n)$ mit n (paarweise verschiedenen) Elementen und einer Zahl $k \in \{0, \dots, n-1\}$. Ordnen Sie die Elemente im Array so an, dass die ersten k Positionen im Array die k-kleinsten Elemente in sortierter Reihenfolgen enthalten.

Nach Lösung des Problems muss im Array gelten:

$$A[0] \le A[1] \le \cdots \le A[k-2] \le A[k-1] \quad \text{and} \quad \forall i \in \{k, \dots, n-1\} : A[k-1] \le A[i].$$

Elemente in Sortierproblemen könne große oder komplexe Objekte sein. In dieser Aufgabe nehmen wir lediglich an, dass eine totale Ordnung existiert und mithilfe der Vergleichsoperation $<$ gefunden werden kann.

a) Entwickeln Sie einen Algorithmus für das Partial Sorting Problem. Ihre Lösung sollte eine (erwartete) Laufzeit von $O(n + k \log n)$ besitzen.

b) Versuchen Sie Lösungen für folgende erweiterte Fragestellungen zu entwickeln:

   Frage 1: Können Sie einen Algorithmus mit Laufzeit $O(n + k \log k)$ angeben?

   Frage 2: Können Sie eine Lösung mit lediglich $O(1)$ zusätzlichen Speicherplatz entwickeln?

   Frage 3: Können Sie einen Algorithmus angeben, welcher die Laufzeit im Worst Case erreicht?

c) Beweisen Sie, dass jeder vergleichs-basierte Algorithmus für das Partial-Sorting-Problem eine Laufzeit von $\Omega(n + k \log k)$ hat.

   Beachten Sie, dass dies bedeutet, dass ein Algorithmus für die erweiterte Frage 1 aus Aufgabenteil b) eine *optimale* Komplexität hat, d. h. bis auf konstante Faktoren eine asymptotisch optimale Laufzeit hat.

2. Aufgabe

50 Punkte

Wenden Sie, wo möglich, das Master-Theorem (Theorem 5.1) an, um die Lösungen der folgenden Rekursionsgleichungen zu bestimmen. Wenn das Master-Theorem nicht anwendbar ist, begründen Sie dies. (Es ist jeweils $T(1) = 1$ gegeben).

a) $T(n) = 8T(\frac{n}{2}) + n^3$

b) $T(n) = 4T(\frac{n}{3}) + n \log n$

c) $T(n) = 4T(\frac{n}{2}) + n^2\sqrt{n}$

d) $T(n) = 27T(n/3) + n^3/\log n$

e) $T(n) = 3T(\frac{n}{2}) + n$

3. Aufgabe

10 + 10 + 20 Punkte

Im k-way Mergesort wird die zu sortierende Menge in k Teile aufgeteilt, diese werden rekursiv sortiert, und anschließend mit einem k-way Merge verschmolzen. Das Standard-Mergesort-Verfahren ist somit ein Spezialfall vom k-way Mergesort für $k = 2$.

Im Folgenden sollen Sie zwei Varianten der Analyse für k-way Mergesort ausführen. In der ersten Variante wird der Aufwand in der Anzahl der Vergleiche angegeben. In der zweiten Variante wird der Aufwand in der Anzahl der zugegriffenen Elemente angegeben. Ein Elementzugriff erfolgt hierbei beim Einlesen des Inputs und beim Rausschreiben in den Output.

Treffen Sie eine geeignete Annahme für die Kosten im Merge-Schritt. Um im k-way Merge das Minimum aus den k sortieren Eingaben zu wählen, bietet sich eine Priority Queue an.

Tipp: Sie können vereinfachend davon ausgehen, dass es ein $i \in \mathbb{N}_0$ gibt, sodass $n = k^i$.

a) Stellen Sie die zwei Rekurrenzgleichungen auf. Geben Sie jeweils eine möglichst genaue Funktion für den “Conquer” Aufwand an.

b) Verwenden Sie, wenn möglich, das Master-Theorem, um die Lösungen der beiden Rekurrenzen zu bestimmen. Wenn das Master-Theorem nicht anwendbar ist, begründen Sie dies.

c) Behandeln k nun als (nicht-konstanten) Parameter und geben Sie eine möglichst kleine obere Schranke für die Rekurrenz in Abhängigkeit von n und k an. Begründen Sie Ihre Lösung (z.B. durch iteratives Einsetzen). Diskutieren Sie die Lösungen der Aufgabenteile b) und c). Gehen Sie dabei insbesondere auf den Vergleich zum Standard Mergesort Verfahren ($k = 2$) ein.

4. Aufgabe

40 Punkte

In einem Turnier sollen n Teams gegeneinander antreten, wobei jedes Team gegen jedes andere spielen soll. n sei hierbei eine Zweierpotenz, also $n = 2^k$.

Entwerfen Sie einen effizienten Divide & Conquer Algorithmus, der einen entsprechenden Spielplan in Form einer $n \times (n-1)$ Tabelle aufstellt, wobei jede Zeile für ein Team und jede Spalte für eine Runde steht. Der Eintrag an der Position (i, j) bestimmt das Gegnerteam von Team i in Runde j. Beispiele sind in Tabelle 1 zu sehen.

$n = 2$

$i \backslash j$	1
1	2
2	1

$n = 4$

$i \backslash j$	1	2	3
1	2	4	3
2	1	3	4
3	4	2	1
4	3	1	2

Tabelle 1: Beispiele für mögliche Turnierpläne für $n = 2$ und $n = 4$.

Begründen Sie die Korrektheit Ihres Algorithmus und bestimmen Sie seine Laufzeit (Θ-Klasse).