Algorithmen und Datenstrukturen 1

ALGO1 · SoSe-2021 · tcs.uni-frankfurt.de/algo1

Wochenplan: Suchen und Sortieren

Revision c0c1956 (2021-04-21)

Vorbereitung

Lies CLRS Kapitel 2 und schau dir die Videos der Woche an.

Dienstag

Aufgabe 1 (Von Hand laufen lassen und Eigenschaften).

a) (einfach) Beschreibe den Ablauf von insertion sort, wenn die Eingabe ein Feld $A = [31, 41, 59, 26, 41, 58]$ ist.

b) (einfach) Verändere den Pseudocode für insertion sort, um das Eingabefeld monoton fallend anstatt monoton wachsend zu ordnen.

c) Analysiere die Laufzeit von linearer Suche *im mittleren Fall*: Wenn das gesuchte Element x zufällig und gleichwahrscheinlich irgendeines der Elemente von A ist, wie viele Einträge von A prüft lineare Suche dann, um x zu finden?

d) (einfach) Beschreibe den Ablauf von merge sort auf dem Feld $A = [3, 41, 52, 26, 38, 57, 9, 49]$.

e) Überzeuge dich, dass insertion sort auch rekursiv wie folgt formuliert werden kann: um $A[0..n-1]$ zu sortieren, sortieren wir $A[0..n-2]$ rekursiv und fügen dann $A[n-1]$ in das sortierte Feld $A[0..n-2]$ ein. Stelle die Rekursionsgleichung für die Laufzeit dieses Algorithmus auf und löse sie.

f) Ein Freund schlägt vor, dass du binäre Suche nutzen solltest, um den Einfügeschritt von insertion sort schneller zu machen. Funktioniert das, und welchen Effekt hat es auf die Laufzeit des Algorithmus?

Aufgabe 2 (Duplikate und nahe Nachbarn). Sei $A[0..n-1]$ ein Feld von ganzen Zahlen.

a) (einfach) Ein *Duplikat* in A ist ein Paar (i, j) von unterschiedlichen Indizes mit $A[i] = A[j]$. Entwirf einen Algorithmus an, der in Zeit $O(n^2)$ findet, ob A ein Duplikat hat.

b) Entwirf einen Algorithmus an, der in Zeit $O(n \log n)$ findet, ob A ein Duplikat hat. *Hinweis:* Nutze merge sort.

c) Ein *nächstes Paar* in A ist ein Paar (i, j) mit $i \neq j$, sodass der Abstand $|A[i] - A[j]|$ minimal ist unter allen Paaren von Einträgen. Entwerfe einen Algorithmus, der den Abstand eines nächsten Paars in Zeit $O(n \log n)$ findet.

Aufgabe 3 (Korrektheit von Merge Sort). Beweise, dass Merge Sort alle gegebenen Felder korrekt sortiert. Hierbei darfst du annehmen, dass die Verflechtungsoperation MERGE auf sortierten Feldern korrekt arbeitet. *Hinweis: benutze vollständige Induktion.*

Donnerstag

Aufgabe 4 (2-Summe und 3-Summe). Sei $A[0..n-1]$ ein Feld von ganzen Zahlen (positive und negative Zahlen sind erlaubt). Das Feld A hat eine *2-Summe*, wenn es zwei Einträge i und j gibt mit $A[i] + A[j] = 0$. Analog hat A eine *3-Summe*, wenn es drei Einträge i, j, und k gibt, sodass $A[i] + A[j] + A[k] = 0$.

a) (einfach) Gib einen einfachen Algorithmus an, der in Zeit $O(n^2)$ feststellt, ob A eine 2-Summe hat.

b) Gib einen Algorithmus an, der in Zeit $O(n \log n)$ feststellt, ob A eine 2-Summe hat. *Hinweis: binäre Suche.*

c) (einfach) Gib einen Algorithmus an, der in Zeit $O(n^3)$ feststellt, ob A eine 3-Summe hat.

d) Gib einen Algorithmus an, der in Zeit $O(n^2 \log n)$ feststellt, ob A eine 3-Summe hat. *Hinweis: binäre Suche.*

e) (sehr schwer) Gib einen Algorithmus an, der in Zeit $O(n^2)$ feststellt, ob A eine 3-Summe hat.

Aufgabe 5 (Auswahl, Partition, und Quick Sort). Sei $A[0..n-1]$ ein Feld von unterschiedlichen ganzen Zahlen. Der Eintrag von *Rang* k in A ist die kt kleinste Zahl unter den Zahlen in A. Der *Median* von A ist die Zahl von Rang $\lceil (n/2) \rceil$.

a) Gib einen Algorithmus mit Laufzeit $O(n \log n)$ an, der A und k als Eingabe nimmt und die Zahl in A ausgibt, die Rang k hat.

Eine *Partition* von A ist eine Aufteilung in Felder A_{low} und A_{high}, sodass A_{low} alle Zahlen von A enthält, die kleiner oder gleich dem Median von A sind, und A_{high} enthält alle Zahlen von A, die größer sind als der Median von A. Es gibt einen Linearzeitalgorithmus, der den Median eines Feldes findet; diesen Algorithmus darfst du im Folgenden einfach annehmen.

b) Gib einen Algorithmus an, der eine Partition von A in Zeit $O(n)$ ausrechnet.

c) (schwer) Gib einen Algorithmus an, der A in Zeit $O(n \log n)$ sortiert, indem er rekursiv partitioniert.

d) (sehr schwer) Gib einen Algorithmus mit Laufzeit $O(n)$ an, der A und k als Eingabe nimmt und die Zahl in A ausgibt, die Rang k hat.

Aufgabe 6 (k-Merge Sort). Professorin M. Erge stellt ihren neuen Algorithmus, 3-Merge Sort, vor. 3-Merge Sort funktioniert genau wie das uns bekannte Merge Sort, nur wird rekursiv in drei Teile anstatt zwei aufgeteilt, die daraufhin sortiert und wieder verflochten werden. Löse die folgenden Teilaufgaben.

a) Zeige, dass sich drei sortierte Felder in asymptotischer Linearzeit verflechten lassen.

b) Führe eine Laufzeitanalyse für 3-Merge Sort durch. Stelle hierzu die Rekursionsgleichung für die Laufzeit $T(n)$ auf und löse diese.

c) (schwer) Verallgemeinere den Algorithmus und die Analyse von 3-Merge Sort zu k-Merge Sort für $k > 3$.
Hat Prof. M. Erge den Durchbruch geschafft? Stellt k-Merge Sort eine Verbesserung gegenüber dem klassischen 2-Merge Sort dar?