Algorithmen und Datenstrukturen
Sommersemester 2022
Woche 10

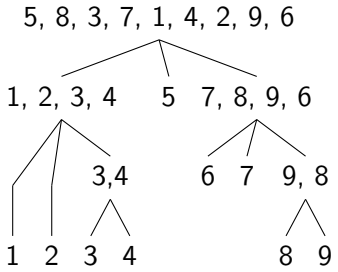
Kevin Angele, Tobias Dick, Oskar Neuhuber,
Andrea Portscher, Monika Steidl, Laurin Wischounig

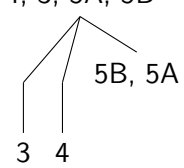
Abgabe bis 07.06.2022 23:59
Besprechung im PS am 09.06.2022

Aufgabe 1 (3 Punkte): Quicksort

- Sortieren Sie das Array $[5, 8, 3, 7, 1, 4, 2, 9, 6]$ manuell mit Quicksort. Wählen Sie das Pivot-Element mit dem Median des ersten, mittleren und letzten Elements. Wenn es zwei mittlere Elemente gibt, wählen Sie das linke.
- Beweisen Sie per Gegenbeispiel, dass Quicksort nicht stabil ist.

Lösung:

- 

- Wir sortieren die Sequenz $[4, 3, 5A, 5B]$ und benuten das erste Element als Pivot. A und B dienen hierbei nur zur Markierung des Elementes und haben keinen Einfluss auf die Sortierung.


Aufgabe 2 (2 Punkte): Parallelisierung von Sortieralgorithmen

In der Vorlesung haben Sie nun einige verschiedene Sortieralgorithmen kennengelernt.

- Selection Sort
- Insertion Sort

- Heap Sort
- Quicksort
- Mergesort

Damit ein Algorithmus parallelisierbar ist, müssen Unteraufgaben unabhängig von einander bearbeitbar sein. Bei welchen der o.g. Sortieralgorithmen ist dies der Fall? Beschreiben Sie jeweils kurz, wieso der Algorithmus parallelisierbar ist bzw. warum er das nicht ist.

Lösung:

- Selection Sort: Nicht parallelisierbar.
- Insertion Sort: Nicht parallelisierbar.
- Heap Sort: Die Botton-Up Heap Construction kann zwar parallelisiert werden, der Rest allerdings nicht. Daher ist Heap Sort nicht einfach parallelisierbar.
- Quicksort: Parallelisierbar, da Divide & Conquer Algrithmus. Unterarrays können parallel bearbeitet werden.
- Mergesort: Parallelisierbar, da Divide & Conquer Algrithmus. Unterarrays können parallel bearbeitet werden.

Aufgabe 3 (5 Punkte): Dynamische Programmierung

Sie sind freiberuflicher Programmierer und müssen Ihre Projekte für den kommenden Zeitraum von m Monaten planen. Dafür müssen Sie aus den zahlreichen Anfragen, die Sie erreicht haben, auswählen. Gehen Sie für diesen Fall davon aus, dass Sie für jedes angebotene Leistungspaket unendlich viele Anfragen haben, und dass ein Projekt entweder ganz oder gar nicht bearbeitet werden kann. Sie bieten n verschiedene Leistungspakete an, die jeweils m_n (ganze) Monate dauern und Ihnen Einnahmen von e_n Euro bescheren. Jedes Leistungspaket kann beliebig oft gewählt werden. Welche Leistungen sollten Sie jeweils wie oft auswählen, um am Ende des Zeitraums so viel Gewinn wie möglich gemacht zu haben?

Die folgenden Arrays zeigen jeweils die Einnahmen E und die Dauer M in Monaten für jedes der n Leistungspakete an:

$$E = [e_1, \ldots, e_n]$$
$$M = [m_1, \ldots, m_n]$$

Die folgende Funktion berechnet den maximalen Gewinn, wobei m die Anzahl der verfügbaren Monate und n die Anzahl der von Ihnen angebotenen Leistungspakete bezeichnet:

$$g[m,n] = \begin{cases} 0 & \text{für } m = 0 \vee n = 0 \\ g[m, n-1] & \text{für } M[n-1] > m \\ \max(g[m, n-1], g[m - M[n-1], n] + E[n-1]) & \text{andernfalls} \end{cases}$$

1. Beschreiben Sie die Funktion $g[m,n]$ zum besseren Verständnis kurz in natürlicher Sprache.

2. Berechnen Sie den maximalen Gewinn, der mit $E = [4500, 6500, 13000]$ und $M = [2, 3, 5]$ gemacht werden kann, wenn sie 12 Monate zur Verfügung haben. Welche Leistungspakete müssen Sie hierfür auswählen? Inkludieren Sie die Tabelle, die von der Funktion $g[m, n]$ ausgefüllt wird, in Ihre Antwort.

3. Wie könnten Sie den Algorithmus, dessen Grundlage die obige Funktion darstellt, modifizieren, damit nicht nur der maximale Gewinn ausgegeben wird, sondern auch die gewählten Leistungspakete?

Lösung:

1. Wenn null Monate oder keine Leistungspakete verfügbar sind, kann kein Gewinn gemacht werden. Wenn nicht genug Zeit ist, um das Modell $M[n]$ zu drucken, können wir es nicht drucken, und berücksichtigen daher nur die anderen Modelle. Wenn wir genug Zeit haben, um Modell $M[n]$ zu drucken, dann gibt es zwei Möglichkeiten für den maximalen Gewinn: Entweder, wir wählen dieses Leistungspaket nicht aus, und berücksichtigen nur die anderen, oder wir wählen dieses Leistungspaket aus und bekommen nun die Summe der Einnahmen dieses Leistungspaketes und der Einnahmen, die wir bereits hatten, bevor wir dieses Leistungspaket ausgewählt hatten.

2. Wir müssen die Tabelle von links nach rechts und von oben nach unten auffüllen:

	n .. Leistungspakete			
m	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	4500	4500	4500
3	0	4500	6500	6500
4	0	9000	9000	9000
5	0	9000	11000	13000
6	0	13500	13500	13500
7	0	13500	15500	17500
8	0	18000	18000	19500
9	0	18000	19500	22000
10	0	22500	22500	26000
11	0	22500	24500	26500
12	0	27000	27000	30500

3. Eine Möglichkeit wäre, immer wenn im dritten Fall der Funktion $g[m - M[n-1], n] + E[n-1]$ größer ist als $g[m, n-1]$ das jeweilige n in einer Liste zu speichern. Diese Liste gibt dann am Ende die jeweiligen Leistungspakete aus.