Technische Universität München
Fakultät für Informatik, I-14
Prof. Harald Räcke
Robin Münk

SoSe 2025
Übungsblatt 8
13. Juni 2025

Algorithmen und Datenstrukturen

Aufgabe 1 Komplexität rekursiver Algorithmen

Gegeben sind folgende Rekursionsgleichungen. Bestimmen Sie einen geschlossenen Ausdruck für die Laufzeit nach der Methode „Raten+Induktion". (Zum Teil Wiederholungen aus der Vorlesung)

a) $T(n) = T(n-1) + 2, T(0) = 3$ (z.B. lineare Suche in der Linked List)

b) $T(n) = T(\frac{n}{2}) + 2, T(1) = 3$ (z.B. binäre Suche in einem Array)

c) $T(n) = T(\sqrt{n}) + 1, T(2) = 1$
(z.B. Berechnung des Maximums auf einem parallelen Rechner)
Hinweis: Sehen Sie sich alle Zahlen $n = 2^{2^k}$ für steigende k's an, zählen Sie, wie oft Sie die Rekursion ausführen müssen, um den Basisfall zu erreichen und finden Sie die Gemeinsamkeit

Aufgabe 2 Master-Theorem

Rufen Sie sich zunächst das Master-Theorem in Erinnerung. Bestimmen Sie anschließend damit die asymptotische Laufzeit der folgenden Funktionen:

a) $A(n) = 2A(\frac{n}{2}) + \log(n^3)$

b) $B(n) = B(\frac{n}{2}) + 42$

c) $C(n) = 4C(\frac{n}{2}) + n\log n$

d) $D(n) = 6D(\frac{n}{3}) + n^{\log_3(2)} \cdot n\log_7^2(4n)$

Aufgabe 3 Divide and Conquer - Maximale Teilsequenz

Gegeben ist ein Array a der Länge n. Es soll die maximale Teilsequenz $\max_{i\leq j}\sum_{k=i}^{j} a[k]$ berechnet werden. Sie haben bereits einen Algorithmus mit der Laufzeit $O(n^2)$ für dieses Problem kennengelernt. Wir entwickeln einen verbesserten Algorithmus, der nach *Divide-and-Conquer-Ansatz* arbeitet.

Die zentrale Beobachtung dabei ist die folgende:

Sei $m \in \{0,\dots,n-1\}$. Dann gibt es genau drei Möglichkeiten: 1. die maximale Teilsequenz ist im Abschnitt $a[0],\dots,a[m-1]$ 2. die maximale Teilsequenz ist im Abschnitt $a[m+1],\dots,a[n-1]$ 3. die maximale Teilsequenz enthält $a[m]$

a) Überlegen Sie, wie für einen gegebenen Index m die maximale Teilsequenz, die $a[m]$ enthält, berechnet werden kann.

b) Entwickeln Sie auf Basis der bisherigen Beobachtungen einen effizienten, rekursiven Algorithmus, um die maximale Teilsequenz eines Arrays zu bestimmen.

c) Analysieren Sie die Laufzeit ihres Algorithmus.

d) Implementieren Sie Ihren Algorithmus in C/C++. Sie können dafür das Testing Framework verwenden.

Aufgabe 4 (P) Linked List - Polynome II

Wir betrachten erneut Polynome der Form

$$p(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0, \quad n \geq 0$$

welche als einfach verkettete Liste umgesetzt sind. Ein Listenelement enthält dabei den Koeffizienten c_i sowie den Exponenten i. Die einzelnen Listenelemente sind nach absteigenden Exponenten geordnet. Listenelemente mit dem Koeffizienten 0 kommen nicht vor.

Sie können für diese Aufgabe das Testing Framework verwenden.

Implementieren Sie folgende Methoden für die Klasse `Polynomial`:

a) `void flip()`: Die Methode spiegelt das Polynom entlang der x-Achse.

b) `void moveUp(float c)`: Die Methode bekommt als Parameter einen `float c` und passt die Koeffizienten so an, dass der Graph des Polynoms um c Einheiten nach oben verschoben wird.

c) `void add(Polynomial& other)`: Die Methode bekommt als Eingabe ein zweites Polynom `other` und addiert dieses zu dem aktuellen Polynom. Listenelemente mit dem Koeffizienten 0 sollen nicht vorkommen.

Hinweise:

- Vielleicht helfen Ihnen sinvolle rekursive Hilfsfunktionen. Arbeiten Sie systematisch die möglichen Fälle ab (beide Polynome zu Ende, nur eins zu Ende, keins zu Ende, ...).
- In der Klasse `Polynomial` gibt es eine print Funktion zum debuggen.
- Als Hilfestellung können Sie sich an den Hinweisen in der Datei `blatt8hint.cpp` orientieren.