Algorithmen und Datenstrukturen
Aufgabenblatt 3

Abgabe: 03.05.2023 17:45 Uhr
Abnahme: 04.05.2023
Gesamtpunkte: 23

1. O-Kalkül **3 Pkt.**

Entwickeln Sie ein Stück Programmcode, das in O(n*log(n)) läuft.

2. Geldwechsler **6 Pkt.**

Implementieren Sie den in der Vorlesung vorgestellten Algorithmus zur Rückgabe von Wechselgeld mit dem Greedy-Verfahren (in Java oder Python). Hierbei soll Ihnen jedoch nicht unendlich viel Kleingeld zur Verfügung stehen, sondern Sie haben eine Kasse mit einer bestimmten Anzahl Münzen von jeder Sorte.

Eingabe: Die zu zahlende Summe $S \epsilon \mathbb{N}$,
der gegebene Betrag $B \epsilon \mathbb{N}$,
die verfügbaren Münzgrößen $m_i, 1 \leq i \leq n$,
die vorhanden Anzahlen $a_i, 1 \leq i \leq n$, jeder Münzart
Ausgabe: das Wechselgeld in Form (m_j, w_j),
m_j = Münzgröße,
w_j = Anzahl der von der Münzgröße m_j zurückgegebenen Münzen
"Geldwechseln nicht möglich", falls nicht genug Wechselgeld da ist.

Begründen Sie Ihre Wahl der eingesetzten Datenstrukturen.

Ziel: Implementierung eines Greedy-Algorithmus.

3. Laufzeitbestimmung von rekursiven Funktionen **9 Pkt.**

Hier sollen Sie Regeln entwickeln zur Aufwandsabschätzung von Funktionen und rekursiven Funktionen. Dazu schätzen Sie die Komplexität des in der Vorlesung vorgestellten rekursiven Algorithmus für die Berechnung der maximalen Teilsumme mit Hilfe der Strategie **Teile-und-herrsche** ab. Begründen Sie Ihre Antwort.

Beantworten Sie dazu zunächst folgende Fragen, die Ihnen helfen sollen, die Regeln zu finden:

Teilaufgabe 3.1: (1 Pkt)

Welche Komplexität haben die Funktionen `rechtesRandmax` und `linkesRandmax`?

Teilaufgabe 3.2: (1 Pkt)

Welche Komplexität hat die Funktion `maxTeilsummerek`, wenn man die Zeile mit den rekursiven Aufrufen nicht berücksichtigt?

Teilaufgabe 3.3: (1 Pkt)

Welche Komplexität hat die Funktion `maxTeilsummerek`, wenn man annimmt, dass der rekursive Aufruf f(n)-mal stattfindet?

Teilaufgabe 3.4: (1 Pkt)

Wie oft kann man eine Folge der Länge 2, der Länge 4, der Länge 8, der Länge 16, ... in zwei gleiche Hälften teilen, d. h. wie oft kann man diese Zahlen halbieren, bis man (in einem Ast) bei 1 angelangt ist?

Teilaufgabe 3.5: (1 Pkt)

Wie oft kann man eine Folge der Länge 27, der Länge 173 oder der Länge 291 in zwei möglichst gleichlange Teilfolgen aufteilen?

Teilaufgabe 3.6: (1 Pkt)

Wie oft kann man eine Folge der Länge n in zwei möglichst gleichlange Folgen aufteilen?

Teilaufgabe 3.7: (1 Pkt)

Wie oft wird $maxTeilsummerek$ bei einer Folgenlänge von n aufgerufen?

Teilaufgabe 3.8: (2 Pkt)

Welche Komplexität hat $maxTeilsummerek$? Begründen Sie Ihre Antwort mit Hilfe der Antworten auf die vorangehenden Fragen. Wie berechnet man die Laufzeit im O-Kalkül für rekursive Funktionen?

Ziel: Komplexitätsabschätzung von rekursiven Funktionen.

4. Rekursive maximale Teilsumme **5 Pkt.**

Implementieren Sie den rekursiven Algorithmus zur Berechnung der maximalen Teilsumme in Python und in Java. Messen Sie dabei wieder die Zeiten (in msec) für die Bearbeitung von 30, 300 und 3000 Folgenelementen. Vergleichen Sie Ihre Ergebnisse mit denen aus dem ersten Aufgabenblatt.