Datenstrukturen und Algorithmen
Übung 10, Frühling 2018

3. Mai 2018

Abgabe: Diese Übung muss zu Beginn der Vorlesungsstunde bis spätestens um 14 Uhr 15 am 9. Mai abgegeben werden. Die Abgabe der DA Übungen erfolgt immer in schriftlicher Form auf Papier. Programme müssen zusammen mit der von Ihnen erzeugten Ausgabe abgegeben werden. Drucken Sie wenn möglich platzsparend 2 Seiten auf eine A4-Seite aus. Falls Sie mehrere Blätter abgeben heften Sie diese bitte zusammen (Büroklammer, Bostitch, Mäppchen). *Der gesammte Sourcecode muss ausserdem elektronisch über Ilias abgegeben werden.*

Die Übung sollte vorzugsweise in Zweiergruppen bearbeitet werden, kann aber auch einzeln abgegeben werden. Vergessen Sie nicht, Ihren Namen und Ihre Matrikelnummer auf Ihrer Abgabe zu vermerken. Jede Übungsserie gibt 10 Punkte. Im Durchschnitt müssen Sie 7 von 10 Punkten erreichen, um die Testatbedingungen zu erfüllen.

Theoretische Aufgaben

1. Nicht jede Greedy-Methode für das Aktivitäten-Auswahl-Problem erzeugt eine maximale Menge paarweise zueinander kompatibler Aktivitäten. Geben Sie Beispiele für die folgenden Strategien, die zeigen, dass die Strategien nicht zu optimalen Lösungen führen. Skizzieren Sie Ihre Beispiele grafisch.

   - Man wählt immer die Aktivität mit der geringsten Dauer, die zu den vorher ausgewählten Aktivitäten kompatibel ist.
   - Man wählt immer die kompatible Aktivität, die sich mit den wenigsten anderen noch verbliebenen Aktivitäten überlappt.
   - Man wählt immer die Aktivität mit der frühesten Startzeit, die zu den vorher ausgewählten Aktivitäten kompatibel ist.

   1 Punkt

2. Gegeben sei folgende Zeichenkette bestehend aus den Zeichen a, b, d, o, p, q, r:

   $$opabqprqpdbqpdopq$$

   Konstruieren Sie einen Huffman-Code für diese Zeichenkette. Stellen Sie den binären Codierungsbaum dar und geben Sie für jedes Zeichen den Binärcode an. Wie lang ist die Huffman-codierte Zeichenkette? Sie müssen nicht den gesamten Code für die Zeichenkette aufschreiben.

   1 Punkt

3. In dieser Aufgabe betrachten wir das Problem, Wechselgeld für n Rappen zusammenzustellen, indem wir so wenige Münzen wie möglich verwenden.

   a) Entwerfen Sie einen Greedy-Algorithmus, der das Wechselgeld aus Münzen mit Nennwerten aus der Menge $N = \{25, 10, 5, 1\}$ Rappen zusammenstellt und beweisen Sie, dass Ihr Algorithmus zu einer optimalen Lösung führt. Der Beweis soll die folgenden Schritte beinhalten:

   – Zeigen Sie, dass das Wechselgeld-Problem eine *optimale Teilstruktur* hat.
   – Zeigen Sie, dass Ihr Algorithmus die *Gierige-Auswahl-Eigenschaft* hat. Das bedeutet, dass es immer eine optimale Lösung gibt die durch Ihre *gierige* Auswahlvorschrift gefunden werden kann (die lokale optimale Wahl führt zu einer globalen optimalen Lösung).

   1 Punkt

   b) Geben Sie eine Menge N von Münzenwerten an, für die Ihr Greedy-Algorithmus nicht zur optimalen Lösung führt. Ihre Menge sollte 1-Rappen Münzen enthalten, damit es für jeden Wert von n eine Lösung gibt. Nehmen Sie an, dass der Nennwert jeder Münze eine ganze Zahl ist. Geben Sie ein Beispiel an, wo der Greedy-Algorithmus versagt. **1 Punkt**

   c) Geben Sie einen auf dynamischer Programmierung basierenden Algorithmus an, der das Wechselgeld für eine beliebige Menge N von k verschiedenen Münzwerten und einen beliebigen Betrag von n Rappen erstellt, wobei in N immer 1-Rappen Münzen enthalten sind. Ihr Algorithmus sollte in Zeit $O(nk)$ ablaufen. *Tipp: Bauen sie eine Tabelle $c[j]$ auf, welche für jeden Betrag j die optimale Anzahl Münzen enthält. Bauen Sie parallel dazu eine Tabelle $denom[j]$ auf, welche den Wert einer beliebigen Münze angibt, die in der optimalen Lösung für j Rappen vorkommt. Im letzten Schritt soll der Algorithmus mittels der Lösungstabelle $denom[j]$ eine optimale Lösung konstruieren.* **1 Punkt**

Praktische Aufgaben

Implementieren Sie eine Java Klasse *HuffmannCode*, welche folgende Funktionalität bietet:

1. Eine Methode `void prefixCode(String s)`, welche für eine gegebene Zeichenkette einen optimalen Präfix-Code generiert. Der Präfix-Code soll als Binärbaum repräsentiert werden. Schreiben Sie dazu eine Klasse `Node` die einen Knoten im Baum darstellt. Verwenden Sie die Klasse `java.util.PriorityQueue` als Prioritätswarteschlange für die Konstruktion des Codes. **2 Punkte**

2. Eine Methode `void printCode(String s)`, welche die codierte Darstellung einer Zeichenkette als Folge von Nullen und Einsen ausgibt. **2 Punkte**

3. Demonstrieren Sie Ihr Programm anhand der folgenden Eingaben und geben Sie die durchschnittliche Anzahl Bits pro Zeichen an:

   - *Jobs launched into a sermon about how the Macintosh and its software would be so easy to use that there would be no manuals.*

- *An academic career in which a person is forced to produce scientific writings in great amounts creates a danger of intellectual superficiality, Einstein said.*

1 Punkt

Vergessen Sie nicht Ihren Sourcecode innerhalb der Deadline über die Ilias Aufgabenseite einzureichen.