Fakultät IV - Elektrotechnik und Informatik
Algorithmen und Datenstrukturen SoSe 2017
Brock / Mabrouk / Sieverling / Stahl

Aufgabenblatt 9bis

letzte Aktualisierung: 29. Juni, 14:23 Uhr

Ausgabe: 29.06.2017
Abgabe: 11.07.2017 23:59

Thema: *Greedy Algorithmen, Strategien für Algorithmendesign*

Abgabe

Die folgenden Dateien müssen für eine erfolgreiche Abgabe im svn Ordner
`Tutorien/txx/Studierende/deinname@TU-BERLIN.DE/Abgaben/`
eingecheckt sein:

Geforderte Dateien:	
`Blatt09bis/src/KnapsackSolver.java`	Aufgabe 1.1, 1.2, 1.3

Wichtige Ankündigungen

- Unit tests dürfen geteilt werden. Lösungen dürfen auf keinen Fall geteilt werden! Wir testen auf Plagiate.
- Verändert die Methodenköpfe vorgegebener Klassen nicht. Verändert keine Klassen die nicht Teil der Abgabe sind und fügt keine neuen hinzu.

1. Aufgabe: 0-1 Knapsack (Rucksack Problem)

In dieser Aufgabe sollt ihr das Rucksack Problem mit verschiedenen Algorithmen lösen. In den Rucksack Problem ist die Aufgabe, einen Rucksack mit einer Auswahl von n Gegenständen von unterschiedlichen Gewichten w_i und Werten v_i zu füllen.
Die Frage ist: welche Gegenstände müssen mitgenommen werden, damit das Gesamtgewicht ein gegebenes Maximalgewicht W nicht überschreitet, aber der Gesamtwert der mitgenommenen Gegenstände möglichst hoch ist.

1.1. Optimale Lösung (60 Punkte) Implementiert die Methode public Knapsack solveKnapsackOptimally(Knapsack k, LinkedList<Item> items), die die optimale Lösung des Knapsack Problems findet. Die Methode nimmt als Parameter ein Objekt des Typs Knapsack (der einen leeren Knapsack mit einem Maximalgewicht definiert) und eine Liste von Gegenständen mit verschiedenen Gewichten und Werten. Als Ergebnis gibt die Methode den gefüllte Rucksack mit dem höchstmöglichen Gesamtwert zurück, ohne dass das Gesamtgewicht überschritten wird.

Eure Implementierung soll alle mögliche Kombinationen von Objekte probieren und die Kombination mit höchstem Gesamtwert benutzen.

1.2. Greedy (Gierige) Lösung I (20 Punkte) Implementiert die Methode public Knapsack solveKnapsackGreedyStupid(Knapsack k, LinkedList<Item> items), die einen greedy Algorithmus implementiert. In dieser Implementierung sollen die Gegenständen in der Reihenfoge des höchsten Wertes hinzugefügt werden, d.h. man probiert erstmal den wertvollsten Gegenstand hinzufügen, dann den nächst wertvollsten etc..

1.3. Greedy (Gierige) Lösung II (20 Punkte) Implementiert die Methode public Knapsack solveKnapsackGreedySmart(Knapsack k, LinkedList<Item> items), die eine greedy Lösung des Problems implementiert. In dieser Implementierung sollen die Gegenstände in der Reihenfolge des Wertes per Gewicht hinzugefügt werden, d.h. man probiert erstmal den wertvollsten Gegenstand per Gewicht hinzufügen, dann den nächst wertvollsten etc..

Hinweis: Dieser Algorithmus liefert für das Fractional-Knapsack Problem die Optimallösung, aber nicht für das hier bearbeitete $(0, 1)$-Rucksackproblem.