Übungen zu Objektorientierte Programmierung 2

frieder.loch@ost.ch

Woche 4: Algorithmenparadigmen

Aufgabe 1: Rückgeld

Die meisten Bezahlautomaten, zum Beispiel Automaten im Parkhaus oder Selecta-Automatien, können Wechselgeld ausgeben. Die einfachste Lösung bei der Ausgabe des Wechselgeldes wäre, alles in 5-Rappen-Stücken auszugeben. Jedoch wird der Kunde kaum Freude daran haben.

In der Vorlesung haben wir eine Lösungen kennengelernt, bei der der Betrag mit einer möglichst geringen Münz- oder Notenzahl bestimmt wird. Dieser Greedy-Algorithmus hat die allgemeine Form:

1. Wählen Sie die grösstmögliche Münze/Note aus, bei der der Zielwert nicht überschritten wird.
2. Ziehen Sie diesen Wert vom Zielwert ab.
3. Zurück zu Schritt 1 bis der Zielwert 0 ist.

a) Implementierung des Algorithmus Implementieren Sie den beschriebenen Algorithmus. Warum handelt es sich bei diesem Beispiel um einen Greedy-Algorithmus?

Erstellen Sie bei der Implementierung zunächst ein Enum, das alle möglichen Noten und Münzen enthält. Der auszugebende Betrag soll auf der Konsole eingegeben werden. Das Ergebnis – also die Noten und Münzen, die vom Automaten zurückgegeben werden – soll auf der Konsole ausgegeben werden.

Der Algorithmus muss nur Ganzzahlen-Werte unterstützen.

b) Grenzen des Algorithmus Der Algorithmus nimmt an, dass im Automaten alle Noten und Münzen jederzeit in ausreichender Menge vorhanden sind. In der Praxis ist das aber nicht immer der Fall. Beschreiben Sie eine Situation, in der der Greedy-Algorithmus versagt, obwohl es theoretisch möglich wäre den Betrag exakt zurückzugeben.

Aufgabe 2: Wegsuche im Labyrinth

Die Wegsuche im Labyrinth kann gut mit Backtracking gelöst werden. In dieser Aufgabe versucht eine Ratte durch ein Labyrinth zu finden.

1. Der Startpunkt wird festgelegt.
2. Die Ratte läuft solange es geht in eine Richtung. Bis entweder das Ziel erreicht wurde, eine Wand vor der Ratte ist oder man schon auf dem Weg war.
3. Jeder Schritt soll im Array markiert werden, auf dem man schon war (`State.WALKED`).
4. Die Ratte versucht in jede Richtung zu gehen. Wenn Sie auf eine Wand stösst, dann geht die Ratte zurück bis zur letzten Kreuzung (Backtracking) bei der noch nicht alle Möglichkeiten getestet wurden. Dabei kann man alle Felder als besucht markieren (`STATE.BACKTRACKED`).

Implementieren Sie die Methode `step` in der Klasse `Rat`. Diese Funktion soll die Lösung des Labyrinths simulieren.

Aufgabe 3: Binärsuche

In der Vorlesung wurde eine rekursive Implementierung der Binärsuche gezeigt. Diese Implementierung konnte nur Integer-Arrays durchsuchen.

Erweitern Sie diese Implementierung, so dass die Lösung nun generisch und iterativ ist. Die Methode erhält eine Liste und das gesuchte Element als Übergabeparameter.