Ada-Kompaktkurs (Wirtschaftsinformatiker), Übungsblatt 8
Claus/N.Weicker, WS 02/03

Aufgabe 1: Fakultät (sehr leicht) 2 Punkte

Die *Fakultät* einer natürlichen Zahl $n \geq 0$ ist bekanntermaßen definiert als $\prod_{i=1}^{n} i$. Alternativ kann man sie auch rekursiv festlegen durch

$$0! = 1, \qquad n! = n \cdot (n-1)! \text{ für } n \geq 1.$$

Schreiben Sie ein Programm mit zwei Funktionen, die diesen Definitionen entsprechen. Die eine Funktion soll die Fakultät mit einer Schleife berechnen, die andere rekursiv. Beide Funktionen sollen ein Argument `n` vom Typ `Natural` als Eingabeparameter haben und die Fakultät von `n` zurückliefern. Im Hauptprogramm soll man eine Zahl eingeben können, von der dann auf beide Weisen die Fakultät berechnet wird.

Aufgabe 2: Politiker II (mittel) 4 Punkte

Diese Aufgabe ist eine Fortsetzung der Aufgabe 6.2. Auf der Webseite des Programmierkurses finden Sie erneut eine Datei, die Datensätze von Politikern mit ihren Amtszeiten und Ämtern enthält. Diese ist so wie in Aufgabe 6.2 aufgebaut mit dem einzigen Unterschied, dass die erste Zeile, in der die Anzahl der Datensätze stand, diesmal fehlt. Da sie die Anzahl also nicht von vornherein kennen, sollen Sie die Daten dieses Mal in verkettete Listen einlesen.
Legen Sie für jede in der Datei vorkommende Amtsbezeichnung eine eigene Liste an, die die jeweiligen Amtsinhaber in chronologischer Reihenfolge enthält. (Die verschiedenen Listen müssten Sie wiederum in einer verketteten 'Liste von Listen' anordnen.) Nach dem Einlesen der Datei soll der Benutzer eine Amtsbezeichnung angeben, woraufhin alle Amtsinhaber zeitlich sortiert ausgegeben werden sollen.

Aufgabe 3: Flaggenproblem (mittel) 5 Punkte

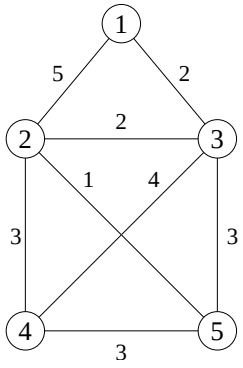
Als *niederländisches Flaggenproblem* bezeichnet man die folgende Aufgabenstellung: Gegeben sei eine Folge von n Mosaiksteinen, die entweder rot, weiß oder blau sind. Die Steine sollen so sortiert werden, dass zuerst alle roten Steine kommen, dann alle weißen und am Schluß die blauen (also die Farbreihenfolge der niederländischen Nationalflagge).
Schreiben Sie ein Programm, welches zunächst eine zufällige Sequenz von n Steinchen erzeugt und diese dann sortiert (wobei n benutzerseitig vorgegeben wird). Die Sequenz soll als doppelt verkettete Liste verwaltet werden. Das Sortieren soll so geschehen, dass Elemente der Liste solange vertauscht werden, bis die Reihenfolge stimmt. Das Vertauschen soll dabei ausschließlich durch Umhängen von Zeigern erfolgen.

Aufgabe 4: Kürzeste Wege (mittel bis schwer) 4+3+2 Punkte

Gegeben sei eine Menge von Punkten, die wir mit 1 bis n bezeichnen. Zwei Punkte i und j können durch einen direkten Weg der Länge d_{ij} verbunden sein, wobei $d_{ij} = d_{ji}$ eine nicht-negative ganze Zahl sei. Existiert kein direkter Weg zwischen i und j, so sei $d_{ij} = \infty$. Man kann dies als Matrix oder als ungerichteten Graph veranschaulichen:
Von der Matrix ist der Symmetrie wegen nur eine Hälfte zu sehen, im Graphen sind nur Kanten endlicher Länge eingezeichnet. Gesucht ist nun die Länge der kürzesten Wege von Knoten 1 zu allen anderen Knoten, wobei wir diese Länge für i mit w_i bezeichnen wollen. Im gezeigten Beispiel können Sie sich leicht davon überzeugen, dass $w_2 = 4$, $w_3 = 2$, $w_4 = 6$ und $w_5 = 5$ ist. Diese Längen lassen sich mit dem Algorithmus von *Dijkstra* berechnen, der sich wie folgt formulieren lässt:

$s_1 := 0$; $s_j := \infty$ für $2 \leq j \leq n$;
$P := \{1, \ldots, n\}$;

$$\begin{pmatrix} 0 & 5 & 2 & \infty & \infty \\ & 0 & 2 & 3 & 1 \\ & & 0 & 4 & 3 \\ & & & 0 & 3 \\ & & & & 0 \end{pmatrix}$$



solange $P \neq \emptyset$:
    wähle i so, dass $s_i = \min_{j \in P} s_j$;
    $P := P \setminus \{i\}$;
    $s_j := \min\{s_j, s_i + d_{ij}\}$ für $1 \leq j \leq n$;

Implementieren Sie diesen Algorithmus. Die Daten sollen aus einer Datei ausgelesen werden; dazu finden Sie auf der Kurs-Webseite einige Beispieldateien. Diese enthalten in der ersten Zeile die Anzahl der Punkte; alle anderen Zeilen sind von der Form i j d_{ij}. Zwischen solchen Paaren von Punkten, für die die Datei keine Längenangaben enthält, existiert kein direkter Weg. Die obige recht abstrakte Formulierung des Algorithmus läßt offen, wie die Distanzen d_{ij} und die Menge P im Einzelnen verwaltet werden, und wie die Auswahl von i erfolgt. Sie können sich für eine der unten aufgeführten Varianten entscheiden, diese werden mit unterschiedlichen Punktzahlen bewertet.
Im einfachsten Fall verwalten Sie die Distanzen in einem zweidimensionalen Feld (d.h. einer Matrix), P als ein n-elementiges Feld von Booleans und erhalten i durch lineares Durchsuchen aller augenblicklichen Distanzen innerhalb von P. (4 Punkte)
Sie können diese Lösung durch zwei Maßnahmen verbessern. Bei großen Graphen ist es oft sinnvoll, nur die Matrixeinträge zu speichern, die einen endlichen Wert enthalten. Bei dieser Variante verwalten Sie also kein zweidimensionales Feld, sondern für jeden Punkt eine Liste von direkten Wegen, die von dem Punkt ausgehen. (3 Zusatzpunkte)
Die zweite Maßnahme besteht darin, P ebenfalls als verkettete Liste zu verwalten und zwar so, dass die Elemente in P der augenblicklich bekannten Distanz zu 1 nach sortiert sind, wodurch sich die Auswahl von i auf das Herausgreifen des ersten Elements der Liste reduziert. Halten Sie in der Liste nur die Punkte vor, für die bereits ein Weg endlicher Länge zu 1 bekannt ist. (2 Zusatzpunkte)

Hinweise

- Pro Aufgabenblatt werden maximal 20 Punkte auf den Übungsschein angerechnet.
- Falls Sie Fragen irgendwelcher Art haben, wenden Sie sich bitte an Ihren Tutor oder an die Übungsleitung: `Nicole.Weicker@informatik.uni-stuttgart.de`, Raum 1.101, oder Tel. 7816-412