Raff

Der berühmte, geldgierige Dagobert beobachtet den Geldmarkt in der Stadt Valuta im Viel-Länder-Eck immer sehr genau. Jeden Donnerstag kann man dort an vielen einzelnen Ständen Geld von einer Währung in eine andere tauschen. Dagobert hat festgestellt, dass er bei den unterschiedlichen Tageskursen oft durch mehrere geschickte Tauschvorgänge bei verschiedenen Händlern sein Geld vermehren kann. Dagobert geht über den Markt und notiert sich Ankaufkurse in einer Tabelle. Heute hat er folgende Tabelle zusammengestellt:

Währung	Kurs
1 Bärentaler	2,70 Mausmark
1 Entenpeseta	0,75 Krötendollar
1 Entenpeseta	0,70 Froschkronen
1 Entenpeseta	1,80 Wolfspfunde
1 Froschkrone	1,10 Krötendollar
1 Froschkrone	1,10 Entenpeseten
1 Krötendollar	2,00 Wolfspfunde
1 Krötendollar	1,90 Wolfspfunde
1 Krötendollar	1,05 Froschkronen
1 Mausmark	1,30 Entenpeseten
1 Mausmark	0,90 Krötendollar
1 Wolfspfund	0,70 Entenpeseten
1 Wolfspfund	0,20 Bärentaler
1 Wolfspfund	0,50 Mausmark

Er könnte z.B. pro Entenpeseta 0,75 Krötendollar bekommen, dann pro Krötendollar 2,0 Wolfspfunde und dann pro Wolfspfund 0,7 Entenpeseten. Mit solchem Tauschzyklus würde er aus je einer Entenpesete (0,75 * 2,0 * 0,7) = 1,05 Entenpeseten machen, also einen Gewinn von (5/3)% pro Tausch erzielen. Dagobert sucht natürlich, von einer beliebigen Währung ausgehend, nach einem Tauschzyklus mit maximalem Gewinn pro Tausch.

→ Aufgabe

1. Schreibe ein Programm, das die Tageskurstabelle einliest, prüft, ob es Tauschzyklen gibt, und gegebenenfalls einen mit maximalem Gewinn pro Tausch bestimmt.

2. Dokumentiere die Programmergebnisse für drei verschiedene Tageskurstabellen. Eine soll die oben angegebene Tabelle sein.

→ Lösungsidee

Die Kurstabelle wird, ausgehend von allen enthaltenen Währungen, nach Zyklen durchsucht. Für alle gefundenen Zyklen wird der Gewinn pro Tausch bestimmt. Ein Zyklus wird ausgegeben, bei dem dieser Wert maximal ist.

Die Suche selbst wird rekursiv durchgeführt: Um einen Tauschweg von einer Währung A zu einer Währung B zu finden, wird für alle Währungen C, in die A getauscht werden kann, geprüft, ob es einen Tauschweg von C nach B gibt.

Allerdings wollen wir mehrere, verschiedene Tauschkurse zwischen zwei Währungen, wie von Krötendollar nach Wolfspfund, ausschließen. Da nach dem maximalen Gewinn pro Tausch gesucht ist, braucht nur der beste Kurs betrachtet werden. Außerdem schließen wir aus, dass in einem Tauschzyklus eine Währung mehrfach vorkommt. In diesem Fall gäbe es einen „inneren Zyklus", den man prinzipiell beliebig oft durchlaufen könnte. Je nach Tauschkurs (z.B. 2,00) würde der Gewinn pro Tausch immer weiter wachsen, was nicht sinnvoll ist (würde man den Gewinn geometrisch mitteln, würden innere Zyklen gar nichts bringen).

→ Programm-Dokumentation

Wir haben die Lösungsidee in Python implementiert. Das Programm besteht aus drei Prozeduren und dem Hauptprogramm. Das Hauptprogramm liest zuerst eine Kurstabelle aus einer Datei ein (Prozedur **liesKursTabelle**). Anschließend werden für jede Währung die Tauschwege zu sich selbst bestimmt (Prozedur **findeWege**). Das Ergebnis ist eine Liste von Zyklen (die selbst wiederum Listen von Währungsnamen sind, bei denen das erste Element gleich dem letzten ist). Dann wird für jeden Zyklus berechnet, mit welchem Faktor sich die jeweilige Währung durch den Zyklus vermehrt hat (Prozedur **vermehrungsfaktor**). Zuletzt werden diese Faktoren gemittelt und dann bestimmt, an welcher Stelle der Zyklenliste ein Zyklus mit maximalem Wert zu finden ist. Dieser Zyklus wird mit Vermehrungsfaktor und Gewinn pro Tausch ausgegeben.

Zu den wichtigen Prozeduren im einzelnen: Die Prozedur **liesKursTabelle** verwendet das Python-Modul **csv** zum Einlesen einer Datei, die Zeilen der Form **Währung1, Währung2,Wechselkurs** enthält (Beispiel: **BT,MM, 2.70**), also „comma-separated values". Die Währungsnamen werden abgekürzt, so dass es kein Problem mit unterschiedlichen Formen gibt. Die Kurse werden in einem zweidimensionalen Feld gespeichert.

Die Prozedur **findeWege** durchsucht dieses Feld und führt den aktuell betrachteten Weg im letzten Parameter mit. Ist die aktuelle Währung gleich dem Ziel, kann dieser Weg abgespeichert werden. Ansonsten wird **findeWege** rekursiv aufgerufen, und zwar mit allen möglichen Tauschzielen der aktuellen Währung als neuem Ausgangspunkt.

→ Programm-Ablaufprotokolle

Aus Platzgründen kann hier nur ein Ablaufprotokoll abgedruckt werden. Die Kurstabelle aus der Aufgabenstellung befindet sich in der Datei **tabelle1.txt**, in der oben angegebenen Form. Damit ergibt sich das Ablaufprotokoll:

```
> python raff.py tabelle1.txt
Lade tabelle1.txt ...
Suche Zyklen...
Berechne Ergebnisse der Zyklen...
Der beste Tauschzyklus ist: ['WP', 'EP', 'WP']
Aus einer Währungseinheit werden in diesem Zyklus
1.26 Einheiten.
Das bedeutet einen Gewinn von 13.0 Prozent pro Tausch.
>
```

→ Programm-Text

```
#!/usr/bin/python
# -*- coding: iso-8859-15 -*-

# raff.py - Lösung für die Aufgabe "Raff" des 22. BWINF
# nach Manuel Holtgrewe

def liesKursTabelle(filename) :
    """ Diese Funktion liest eine Kurstabelle aus der Datei
    mit dem Namen "filename" und gibt sie als Feld zurück. """

    tabelle = {}
    #Hilfsmodul für die Bearbeitung von CSV-Dateien:
    import csv
    reader = csv.reader(file(filename))
    for zeile in reader :
        w1, w2, kurs = zeile[0], zeile[1], float(zeile[2])
        if not tabelle.has_key(w1) :
            tabelle[w1] = {}
        if not tabelle[w1].has_key(w2) :
            tabelle[w1][w2] = kurs
        # Tabelle nur mit besserem Kurs aktualisieren
        elif kurs > tabelle[w1][w2] :
            tabelle[w1][w2] = kurs
    return tabelle

def findeWege(tabelle, waehrung, zielw, wege, weg) :
    """ Diese Funktion durchsucht die "tabelle" nach einem Tauschweg
    von "waehrung" aus nach der Währung "zielw"
    und gibt gefundene Wege im Parameter "wege" zurück """

    # Pfad um aktuellen Knoten verlängern...
    weg = weg + [waehrung]
    # Haben wir damit einen gültigen Weg gefunden?
    if (waehrung == zielw) and (len(weg) > 1) :
        wege.append(weg)
        return
    # Prüfen, ob es von hier aus weiter geht...
    if not tabelle.has_key(waehrung) :
        return
    # Suche auf allen Kanten nach möglichen Pfaden
    for w in tabelle[waehrung] :
        if (w not in weg) or (w == zielw) :
            findeWege(tabelle, w, zielw, wege, weg)

def vermehrungsfaktor(tabelle, zyklus) :
    """ Diese Funktion berechnet, um welchen Faktor das Geld sich beim
    Umtauschen auf diesem Zyklus vermehrt/verringert. """
    ergebnis = 1
    for i in range(1, len(zyklus)) :
        ergebnis = ergebnis * tabelle[zyklus[i - 1]][zyklus[i]]
    return ergebnis

### Hauptprogramm ###

# Hole den Dateinamen der Kurstabelle aus der Kommandozeile
import sys
if len(sys.argv) != 2 :
    print "Fehler beim Aufruf. Bitte so aufrufen:"
    print " %> python raff.py kursdatei"
    raise SystemExit()
datei = sys.argv[1]

# Lade die Kurstabelle
print "Lade", datei, "..."
tabelle = liesKursTabelle(datei)

print "Suche Zyklen..."
zyklen = []
# Zyklen sind Wege von einer Währung zu sich selbst
for waehrung in tabelle :
    findeWege(tabelle, waehrung, waehrung, zyklen, [])

# Berechne Faktoren und Gewinn pro Tausch
print "Berechne Ergebnisse der Zyklen..."
faktoren = []
for i, zyklus in enumerate(zyklen) :
    faktor = vermehrungsfaktor(tabelle, zyklen[i])
    faktoren.append(faktor)
gemittelte_faktoren = []
for i, zyklus in enumerate(zyklen) :
    mittel = (faktoren[i] - 1) * 100 / (len(zyklen[i]) - 1)
    gemittelte_faktoren.append(mittel)

# Suchen des besten Faktors
max_index = gemittelte_faktoren.index(max(gemittelte_faktoren))

print "Der beste Tauschzyklus ist:", zyklen[max_index]
print "Aus einer Währungseinheit werden in diesem Zyklus"
print faktoren[max_index], "Einheiten."
print "Das bedeutet einen Gewinn von", gemittelte_faktoren[max_index], \
    "Prozent pro Tausch."
```