28. November 2013

HAW Hamburg

Dokumentation

Zu Aufgabenblatt 06 aus der Vorlesungsreihe „Algorithmen und Datenstrukturen“

Dokumentation

Zu Aufgabenblatt 06 aus der Vorlesungsreihe „Algorithmen und Datenstrukturen“

# ÜbungsAufgabe 6.1

*Beachten Sie die Problemstellung Perfektes Müsli:  
Fridolin Fröhlich mischt sich an der Müslibar sein Frühstück zusammen. Er hat dabei die  
Auswahl unter n Produkten, die wir von 1 bis n durchnummerieren. Jedes Produkt hat sein  
spezifisches Volumen Vi und den Preis Pi. Der einfachheithalber nehmen wir an, dass jedes  
Produkt genau einmal vorhanden ist. Es ist aber zulässig, dass einige der n Produkte identisch  
sind.  
Fridolin Fröhlich hat eine Müslischüssel mit dem Volumen Vs. Diese möchte er zwar mit einer  
Teilauswahl der n Produkte möglichst randvoll befüllen, gleichzeitig den Preis aber möglichst  
gering halten.  
Formal: Gegeben ist die Menge der Müslizutaten M={1,...,n}. Für jede Teilmenge I der  
Zutaten M ergibt sich das Volumen zu und der Preis zu .  
Für ein perfektes Müsli ist also eine Teilmenge I der Müslizutaten M gesucht, so dass das  
freibleibende Volumen Vs - V(I) und der Preis P(I) minimiert werden.  
Idee: Natürlich kann es kein perfektes Müsli geben, wenn alle Produkte mehr Volumen haben,  
als in der Schüssel noch Platz ist.  
Andernfalls können wir mindestens noch ein Produkt auswählen. Angenommen wir wüssten  
schon, dass das Produkt i in das perfekte Müsli muss. Dann finden wir das perfekte Müsli,  
indem wir mit den anderen Zutaten M \ {i} das perfekte Müsli für eine kleinere Schale mit  
dem Volumen Vs - Vi bestimmen.  
Nun müssen wir nur noch herausfinden, ob i tatsächlich ausgewählt werden muss....*

## Teilaufgabe 1 & 2

*Helfen Sie Fridolin Fröhlich beim Aussuchen!  
&  
Präzisieren Sie die obige Idee derart, dass das perfekte Müsli gefunden wird – und zwar so, dass die Rekursion erkennbar wird.  
Beschreiben Sie insbesondere, wie damit umzugehen ist, dass wir ja nicht wissen, dass das Produkt i in das perfekte Müsli muss.*

Da wir nicht wissen, welche Zutaten auf jeden Fall im Müsli sein müssen, werden diese zunächst nicht beachtet. Sollte es dann klar sein, welche es sein sollen, dann werden sie entsprechend aus den uebrigeZutaten in die gewaehlteZutaten geschoben, das Volumen wird um das Volumen der Zutat verringert und der Preis um den Preis der Zutat erhöht.

Function Muesli (uebrigeZutaten, gewaehlteZutaten, volumen, preis)  
 if(uebrigeZutaten.size == 0 oder bestVol > volumen)   
 if(bestVol == Volumen && bestPreis > preis)  
 Save Data  
 return  
 endif  
  
 if(uebrigeZutaten[0].gewicht <= Volumen)  
 Muesli(uebrigeZutaten.tail,  
 gewaehlteZutaten.add(uebrigeZutaten[0]),   
 Volumen - uebrigeZutaten.gewicht,   
 preis + uebrigeZutaten[0].preis)  
 endif

Muesli(uebrigeZutaten.tail,gewaehlteZutaten,volumen,preis)  
endFunction

## Teilaufgabe 3

*Definieren Sie eine Funktion I, die die gesuchte Indexmenge berechnet. Versuchen Sie dabei ohne programmiersprachliche Kontrukte auszukommen, sondern vrwenden Sie übliche mathemaische Formulierungen.*

Sei p ein Preis und v ein Volumen, dann bezeichne man eine Menge von (p,v) Tupeln  
als Z in der ein Tupel einem Produkt entspricht.

Sei P(M) die Potenzmenge von M, dann ist PM (Perfektes Müsli) definiert als

## Teilaufgabe 4

*Wie geht Ihr Algorithmus damit um, wenn es mehrere perfekte Müslis gibt?*

Sollte es mehrere perfekte Müslis geben, dann wird nur das als erstes gefundene perfekte Müsli als solche bekannt gegeben. Denn die nachfolgenden perfekten Müslis unterscheiden sich weder in Volumen noch im Preis voneinander.

## Teilaufgabe 5

**public** **static** List<Produkt> perfektesMuesli(List<Produkt> products, **int** schuesselgroesse) {  
 *bestVol* = Integer.*MAX\_VALUE*;  
 *bestPrice* = Integer.*MAX\_VALUE*;  
 *bestProducts* = **new** ArrayList<Produkt>();  
 *muesliHelper*(products, **new** ArrayList<Produkt>(),  
 schuesselgroesse, 0);  
 **return** *bestProducts*;  
 }  
 **private** **static** **void** muesliHelper(List<Produkt> uebrigeZutaten,  
 List<Produkt> gewaehlteZutaten, **int** volumen, **int** preis)   
 **if**(uebrigeZutaten.size() == 0) {  
 **if**((*bestVol* == volumen && *bestPrice* > preis)   
 || *bestVol* > volumen) {  
 *bestProducts* = gewaehlteZutaten;  
 *bestVol* = volumen;  
 *bestPrice* = preis;  
 }  
 **return**;  
 }  
 List<Produkt> uebrig = **new** ArrayList<Produkt>();  
 **for**(**int** i = 1; i < uebrigeZutaten.size(); i++) {  
 uebrig.add(uebrigeZutaten.get(i));  
 }  
 **if**(uebrigeZutaten.get(0).getVol() <= volumen) {  
 List<Produkt> gewaehlt = **new** ArrayList<Produkt>();  
 gewaehlt.addAll(gewaehlteZutaten);  
 gewaehlt.add(uebrigeZutaten.get(0));  
 *muesliHelper*(uebrig, gewaehlt, volumen -   
 uebrigeZutaten.get(0).getVol(), preis +   
 uebrigeZutaten.get(0).getPrice());  
 }  
 *muesliHelper*(uebrig, gewaehlteZutaten, volumen, preis);  
 }