Problem Plecakowy
(Knapsack problem)

Wiktor Chojnacki

Fizyka techniczna
Politechnika Krakowska

16 czerwca 2017

Spis treści

Wstęp

Problem plecakowy można porównać do sytacji w której złodziej znajduje pewne przedmioty i zastanawia się, które przedmioty zabrać, aby ich wartość była jak największa i jednocześnie nie zabrać więcej niż może unieść.

Rozwinięcie

Program, który napisałem opiera się głównie na pętlach for i if. Spis rzeczy znajduje się w liście, a poszczególne operacje pobierają sobie odpowiednią wartość z podanych danych.
Przeanalizujmy kod, który napisłem.

Kod programu

Biblioteki, stałe

```
import matplotlib.pyplot as plt
import numpy as np
udzwig = 20 plecak = {}
rzeczy = (("zeszyt", 0.2, 1), ("dlugo-
pis",0.05,0.5),("portfel",0.2,15),("kubek",0.4,8),("cegla",4,1),
("deska",1,15),("woda",2,5),("kombinerki",0.5,20),("baczka
piwa",25,200),("gwozdzie",1,20),("ksiazka",1,25) ,("okula-
ry",0.3,20),("pustak",3,10),("gips",20,15),("podkoszulek",0.5,15))
```

Program korzysta z bibliotek matplot i numpy. Plecak używa 'wąsatych' nawiasów, ponieważ nie używamy tutaj zwykłej listy, a takiej, która ułatwi nam wydobywanie danych z elementów na niej. Do listy 'rzeczy' możemy dowolnie dopisywać elementy w kolejności ("nazwa",waga, wartość).

Kod programu

Funkcja ProblemPlecakowy

```
def ProblemPlecakowy(rzeczy, udzwig): return sum([n[2] for n in rzeczy]) if sum([n[1] for n in rzeczy]) <= udzwig else 0
```

Funkcja ta sumuje wartość zebranych w plecaku przedmiotów o ile udźwig nie został przekroczony.

Kod programu

Funkcja rozwiazanie

```
def rozwiazanie(rzeczy, udzwig): if not rzeczy:
return ()
if (rzeczy,udzwig) not in plecak:
nazwa = rzeczy[0]
wartosci = rzeczy[1:]
dodaj = (nazwa,) + rozwiazanie(wartosci, udzwig - nazwa[1])
niedodaj = rozwiazanie(wartosci, udzwig)
if ProblemPlecakowy(dodaj, udzwig) >
ProblemPlecakowy(niedodaj, udzwig):
answer = dodaj
else:
answer = niedodaj
plecak[(rzeczy,udzwig)] = answer
return plecak[(rzeczy,udzwig)]
```

Analiza funkcji rozwiazanie

Pierwsza pętla if not jest zabezpieczeniem, na wypadek pustej listy. Druga pętla wrzuca przedmioty do plecaka. Poprzez nawiasy kwadratowe wybieramy sobie poszczególną część badanego elementu dzięki czemu program sam sobie sprzwdza czy czasem nie przekroczyliśmy udźwigu i czy jest to najoptymalniejszy zestaw przedmiotów. Pętla if w pętli if not porównuje czy kolejny element z listy jest bardziej otymalny od poprzednich i ewentualnie wyrzuca ten mniej optymalny.

Kod programu

Podanie wyniku końcowego

```
odpowiedz = rozwiazanie(rzeczy, udzwig)
print "Optymalne rzeczy do zabrania:"
for n in odpowiedz:
print n[0]
print "Laczna wartosc przedmiotow:",
ProblemPlecakowy(odpowiedz, udzwig)
print "Laczna waga przedmiotow:", sum([n[1] for n in odpowiedz])
```

'odpowiedz' uruchamia ponownie główną funkcję, by zapełnić plecak. Print wyrzuca na ekran potrzebne dane, a funkcja for wyrzuca wszystkie nazwy przedmiotów z listy, którą stworzyla 'odpowiedz'

Kod programu

```
45
46 obiekty = (n[0] for n in rzeczy)
47 y_pos = np.arange(len(rzeczy))
48 cenadowagi = [n[2]/n[1] for n in rzeczy]
49
50 plt.barh(y_pos, cenadowagi, align="center", alpha=0.5)
51 plt.yticks(y_pos, obiekty)
52 plt.xlabel("cena[zl]/masa[kg]")
53 plt.title("Wartosc poszczegolnych elementow")
54
55 plt.show()
```

obiekty bierze wszystkie nazwy przedmiotów z zaimplementowanej na początku listy. y_pos liczy ile tych elementów jest, by odpowiednio przeskalować wykres. cenadowagi daje wartość przedmiotu w stosunku do jego wagi. Reszta dotyczy wyświetlania wykresu barh mówi, że słupki są poziome, a nie pionowe. align wprowadza wyśrodkowanie nazw, a alpha zmniejsza intensywność koloru słupka.

Koniec

Dziękuję za uwagę