Knapsack Problem – Pakowanie Plecaka

I. Wprowadzenie

Problem plecakowy jest jednym z najczęściej poruszanych problemów optymalizacyjnych. Opracowany przeze mnie algorytm opiera się na podejmowaniu prób umieszczenia w plecaku elementów o różnej wadze, tak aby sumaryczna waga plecaka osiągnęła pewną określoną wartość. Nie ma wymogu, by w plecaku zostały umieszczone wszystkie elementy. Algorytm rekurencyjny po kolei sprawdza różne możliwości dopóki nie odnajdzie tej pożądanej lub też nie skończą się możliwe konfiguracje elementów.

II. Opis interfejsu

Aby uruchomić program należy wpisać w konsoli polecenie *„python Knapsack.py”*, a następnie poczekać na komunikat:

```
E:\UJ FAIS\PYTHON\Projekt>python knapsack.py
Do you want to add your variables [1], check proposed ones? [2] or get them from file of yours [3]?
```

Można wtedy wybrać jedną z trzech opcji, po wybraniu opcji wprowadzenia własnych danych należy nacisnąć na klawiaturze klawisz „1”, w przypadku wybrania opcji proponowanych przykładowych danych należy nacisnąć „2” zaś w przypadku wybrania opcji wprowadzenia danych z pliku należy nacisnąć „3”. Jeśli wprowadzona odpowiedź będzie niepoprawna program poprosi nas o powtórzenie swojej decyzji:

```
E:\UJ FAIS\PYTHON\Projekt>python knapsack.py
Do you want to add your variables [1], check proposed ones? [2] or get them from file of yours [3]? 4
Are you sure your answer was correct? Try again! Do you want to add your variables [1], check proposed ones? [2]
or get them from file of yours [3]?
```

Po wybraniu opcji wprowadzenia własnych danych program kolejno poprosi o wprowadzenie ilości zestawów danych jakie użytkownik chciałby wpisać, pojemność plecaka, ilość pakowanych obiektów oraz ich kolejne wagi:

```
How many of data sets you want to enter? 2
How big is your knapsack? 15
How many objects you want to pack? 5
Add object number 1 3
Add object number 2 2
Add object number 3 10
Add object number 4 4
Add object number 5 4
15 = 3 2 10
How big is your knapsack? 80
How many objects you want to pack? 3
Add object number 1 40
Add object number 2 40
Add object number 3 30
80 = 40 40
```

Jako wynik program poda znalezioną sumę elementów składających się na daną pojemność. W przypadku nieodnalezienia żadnej możliwości wypełnienia plecaka program wyświetli stosowny komunikat:

```
How big is your knapsack? 50
How many objects you want to pack? 2
Add object number 1 1
Add object number 2 1
There is no solution!
```

Po wybraniu opcji gotowych danych program uruchomi się informując nas o danych oraz wyniku przeprowadzanych operacji:

```
Do you want to add your variables [1], check proposed ones? [2] or get them from file of yours [3]? 2
For data: Size of knapsack = 30 and objects 10, 10, 9, 5, 5
30 = 10 10 5 5
For data: Size of knapsack = 300 and objects 10, 10, 90, 5, 5, 30, 10, 50, 40, 32, 14, 5, 10, 98, 36, 29, 30, 40,
 67, 30
300 = 10 10 90 5 5 30 10 50 40 14 36
For data: Size of knapsack = 38 and objects 10, 10, 9, 7, 15
There is no solution!
For data: Size of knapsack = 56 and objects 9, 6, 3, 9, 7, 10, 6, 2, 1, 3, 7
56 = 9 6 3 9 7 10 6 2 1 3
```

Gdzie ostatni z zestawów danych jest każdorazowo losowo generowany.

Po wybraniu opcji wprowadzenia danych z pliku program poprosi nas o podanie nazwy pliku, w którym znajdują się dane, które powinny zapisane być w formacie:

[Ilość zestawów danych]
Bloki tekstu * ilość zestawów danych:
[Pojemność plecaka]
[Ilość pakowanych elementów]
Bloki tekstu * ilość zestawów danych:
[Waga pakowanego obiektu]

W folderze projektu znajduje się przykładowy plik tekstowy o nazwie „Test”. Po wybraniu trybu czytania danych z pliku tekstowego i wybraniu go jako czytanego pliku powinny pojawić się następujące wyniki:

```
Do you want to add your variables [1], check proposed ones? [2] or get them from file of yours [3]? 3
What is the name of your file? Test
30 = 10 10 5 5
There is no solution!
```

III. Opis algorytmu

```
if currentObject == len(objects):

    return False

elif objects[currentObject] < capacity:

    if not knapsack(objects, capacity - objects[currentObject], currentObject + 1):

        return knapsack(objects, capacity, currentObject + 1)

    else:

        result = str(objects[currentObject]) + ' ' + result
        return True

elif objects[currentObject] == capacity:

    result = str(objects[currentObject]) + ' ' + result
    return True

else:

    return knapsack(objects, capacity, currentObject + 1)
```

Algorytm rekurencyjny szukający rozwiązania dla danych: lista obiektów - „*objects*”, pojemność plecaka - „*capacity*” oraz obecnie sprawdzany obiekt - „*currentObject*” kończy swoje działanie, gdy znajdzie rozwiązanie zgodne z wymaganiami (całkowicie spakowany plecak) zwracając wartość ”*True*” lub gdy osiągnie koniec tablicy obiektów, nie wypełniając wcześniej plecaka, zwracając wartość „*False*”. Gdy podczas jego działania znajdzie obiekt możliwy do zapakowania sprawdza wynik dalszego pakowania z nim umieszczonym już w plecaku, jeśli nie odnajdzie takiego rozwiązania próbuje zrobić to samo z następnym elementem, jeśli znajdzie rozwiązanie kończy pracę zwracając „*True*”. Podobnie w przypadku znalezienia obiektu całkowicie wypełniającego plecak. Jeżeli obiekt jest zbyt duży by spakować go do pustego plecaka jest on pomijany i algorytm wywoływany jest dla następnego elementu w liście obiektów.

IV. Podsumowanie

Aplikacja jest przyjazna użytkownikom, zwraca jasne i czytelne komunikaty zarówno w przypadku odnalezienia rozwiązania jak i jego braku. Sam algorytm działa bardzo szybko, z wykorzystaniem rekurencji w optymalny sposób odnajdując korzystne wyniki dla każdego zestawu danych. Do korzystania z niej nie są potrzebne żadne zewnętrzne biblioteki, a sama implementacja jest możliwie najbardziej przejrzysta i czytelna.

V. Literatura

- *Ze zbioru skryptów z wykładów Wydziału Matematyki i Informatyki, przedmiotu Metody Programowania - W06_Rekurencja, dr Jan Rosek*