**Imię i nazwisko:** Jakub Pranica **Grupa:** Informatyka -13i **Przedmiot:** Metody Programowania

**Wykładowca:** dr inż. Zbigniew Kokosiński **Semestr:** 2 **Data:** 08-05-2017 **Algorytm z nawrotami**

**Temat:** Napisać algorytm znajdujący optymalne upakowanie plecaka dla wprowadzonych danych (tj. uzyskanie maksymalnej wartości przedmiotów w plecaku, nie przekraczając jego pojemności). Zastosować algorytm z nawrotami, dla wariantu dyskretnego (0-1). Każdy przedmiot można użyć (włożyć do plecaka) maksymalnie jeden raz.

**Dyskretny problem plecakowy** (ang. discrete knapsack problem) jest problemem wyboru przedmiotów, tak by ich sumaryczna wartość była jak największa i jednocześnie mieściły się w plecaku.

**Algorytm z nawrotami** (ang. backtracking) – ogólny algorytm wyszukiwania wszystkich (lub kilku) rozwiązań niektórych problemów obliczeniowych, który stopniowo generuje kandydatów na rozwiązanie jednak, gdy stwierdzi, że znaleziony kandydat c nie może być poprawnym rozwiązaniem, nawraca (ang. "backtracks") do punktu, gdzie może podjąć inną decyzje związaną z jego budową.

**Opis programu:** Użytkownik podaje pojemność plecaka (pojemnosc) oraz liczbę rozpatrywanych przedmiotów (n). Użytkownik może wybrać, czy samemu chce wprowadzić wagi oraz wartości przedmiotów, czy program ma je wygenerować. Dane te przechowywane są odpowiednio w tablicach: wartości[], wagi[]. Następnie program korzysta z zasady działania algorytmu Ichiro Semby (z nawrotami) przedstawionego w pseudokodzie (na odwrocie kratki) wypisując przy tym kolejne kroki algorytmu, co pozwala na obserwację działania programu. Algorytm pozwala wyznaczyć najbardziej wartościową mieszczącą się w plecaku kombinację przedmiotów (bez powtórzeń). Na koniec program podaje jeszcze uzyskaną wartość oraz zajmowane przez daną kombinację przedmiotów miejsce w plecaku.

**Napotkane problemy: 1.**Reprezentacja macierzowa właściwości przedmiotów zapisana, za pomocą tablic w języku programowania c++ **2.**Określenie i wygenerowanie potrzebnych danych. **3.** Integracja właściwości przedmiotu zapisanych w różnych tablicach (waga/wartość/nazwa przedmiotu) **4.**Wypisanie na ekran i do pliku rozwiązań problemu plecakowego. **5.** Implementacja algorytmu z nawrotami w c++

**Przykładowy wynik programu:**

Program rozwiązuje dyskretny problem plecakowy (0-1) dla wprowadzonych danych. Gdzie wypisać wyniki? 1 - na ekran 2 - do pliku **1**

**(Przykład 1)** Skąd wziąć dane? 1-wygeneruj losowe 2-pozwol mi podać **2**

Podaj pojemność plecaka: **17**

Podaj liczbę przedmiotów (n): **4**

Podaj wagę 1-go przedmiotu: **4**

Podaj wartość 1-go przedmiotu: **5**

Podaj wagę 2-go przedmiotu: **7**

Podaj wartość 2-go przedmiotu: **10**

Podaj wagę 3-go przedmiotu: **8**

Podaj wartość 3-go przedmiotu: **11**

Podaj wagę 4-go przedmiotu: **9**

Podaj wartość 4-go przedmiotu: **13**

Reprezentacja działania programu: (! oznacza nowa największa wartość)

(\* oznacza przekroczenie pojemności plecaka i "nawrot" algorytmu)

1! 12! 123\*

1 12 124\*

1 12 134\*

1 12 234\*

1 12

1 13 134\*

1 13 234\*

1 13

1 14

1 23 234\*

1 23

1 24

1 34

1

2 23 234\*

2 23

2 24

2 34

2

3 34

3

4

Plecak zawiera najbardziej wartościową mieszczącą sie kombinacje przedmiotów (bez powtórzeń)

Przedmioty w plecaku: **1 2** Wartość tych przedmiotów: **24** Zajęte miejsce w plecaku: **17/17**

**(Przykład 2)** Skąd wziąć dane? 1-wygeneruj losowe 2-pozwol mi podać **1**

Podaj pojemność plecaka: **10**

Podaj liczbę przedmiotów (n): **8**

Podaj maksymalna wartość jednego elementu: **9**

Podaj maksymalna wagę jednego elementu: **20**

Wygenerowałem i posortowałem (malejąco względem wag) 8-przedmiotów: (Indeks elementu: waga ; wartość)

1: 20 ; 2

2: 15 ; 3

3: 14 ; 3

4: 12 ; 9

5: 9 ; 7

6: 6 ; 6

7: 4 ; 6

8: 1 ; 2

Reprezentacja działania programu: (! oznacza nowa największą wartość)

(\* oznacza przekroczenie pojemności plecaka i "nawrot" algorytmu)

1\*

2\*

3\*

4\*

5! 56\*

5 57\*

5 58!

5 67! 678\*

5 67

5 68

5 78

5

6 67 678\*

6 67

6 68

6 78

6

7 78

7

8

Plecak zawiera najbardziej wartościową mieszczącą sie kombinacje przedmiotów (bez powtórzeń)

Przedmioty w plecaku: **6 7**

Wartość tych przedmiotów: **12**

Zajęte miejsce w plecaku: **10/10**

**Pseudokod:**

**Wypisz na ekran:** "Program rozwiązuje dyskretny problem plecakowy (0-1) dla wprowadzonych danych. \n"

**Wypisz na ekran:** "Podaj pojemność plecaka: \n" (cin >> pojemnosc;)

**Wypisz na ekran:** "Podaj liczbę przedmiotów (n): \n" (cin >> n;)

**Utwórz tablice: wartosci[], wagi[], a[] oraz zmienne: int waga, int wartość, bool czyZmienicElementy, int poziom = 0; int najw = 0;**

**Wypisz na ekran:** "Skąd wziąć dane? 1-wygeneruj losowe 2-pozwol mi podać \n"

**Jeżeli użytkownik wpisał 2, wygeneruj potrzebne dane - w przeciwnym wypadku poproś użytkownika o podanie wag oraz wartości przedmiotów.**

**Wprowadzone lub wygenerowane dane zapisz odpowiednio w tablicach wartosci i wagi, a następnie posortuj je malejąco względem wag (używając np. sortowania bąbelkowego). Niech tablica a reprezentuje kolejne analizowane kombinacje przedmiotów.**

**Wykonuj w pętli:**

waga = 0**;** wartosc = 0**;** czyZmienicElementy = false**;**

**Dla** i:=0 **do** poziom-1 **wykonuj:**

**Wypisz na ekran:** a[i]

waga += wagi[ a[i]-1 ]**;**

wartosc += wartosci[ a[i]-1 ]**;**

**Jeżeli** waga<=pojemnosc **to:**

poziom++;

**Wypisz na ekran:** " "

**Jeżeli** wartosc>najw **to** najw = wartosc**; oraz wypisz na ekran:** "!"

**Jeżeli** poziom>n || a[poziom-2]==n **to:**

czyZmienicElementy = true; **oraz przejdź do nowej linii** //dokonaj nawrotu

**w przeciwnym wypadku:** a[poziom-1] = a[poziom-2]+1**;**

**w przeciwnym wypadku:**

**Wypisz na ekran:** "\*" **oraz przejdź do nowej linii** //dokonaj nawrotu

czyZmienicElementy = true;

**Jeżeli** a[0]==n **to przerwij wykonywanie pętli**

**Jeżeli** czyZmienicElementy==true **to wykonaj funkcję** zamienElementy(a); //Funkcja ta tak zmienia wartości tablicy a, aby reprezentowała ona kolejną możliwą kombinację przedmiotów, zgodnie z zasadami wykorzystywanego algorytmu z nawrotami.

**Wypisz na ekran uzyskane optymalne rozwiązania danego problemu plecakowego (najbardziej wartościową mieszczącą się kombinację przedmiotów, wraz z jej wartością oraz wagą).**

**Wnioski:** Na podstawie analizy wyników zwracanych przez program i obliczeń wykonanych poza nim, można stwierdzić, iż program działa i prawidłowo rozwiązuje dyskretny problem plecakowy (0-1: każdy przedmiot można użyć maksymalnie jeden raz) dla podanych przez użytkownika danych, korzystając z algorytmu nawrotami. Takim algorytmem jest algorytm Ichiro Semby, którego zasada działania została użyta w programie.

**Literatura:**

* *T.Asano, T.Ibaraki, H. Imai, T. Nishizeki Clifford Stein*: **Algorithms: International Symposium SIGAL '90**, Tokyo, Japan, August 16-18, 1990. Proceedings
* **pl.wikipedia.org/wiki/Problem\_plecakowy**
* **pl.wikipedia.org/wiki/Algorytm\_z\_nawrotami**