**Imię i nazwisko:** Jakub Pranica **Grupa:** Informatyka -13i **Data:** 10-04-2017 **Przedmiot:** Metody Programowania

**Wykładowca:** dr inż. Zbigniew Kokosiński **Semestr:** 2

**Temat:** Napisać algorytm znajdujący optymalne upakowanie plecaka dla wprowadzonych danych (tj. uzyskanie maksymalnej wartości przedmiotów w plecaku, nie przekraczając jego pojemności). Dozwolone jest wielokrotne użycie przedmiotów.

**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. Przy podanym zbiorze elementów o podanej wadze i wartości, należy wybrać taki podzbiór by suma wartości była możliwie jak największa, a suma wag była nie większa od danej pojemności plecaka.

**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 algorytmu dynamicznego przedstawionego w pseudokodzie (na odwrocie kratki) wypisując przy tym wartości tablic w kolejnych krokach, co pozwala na obserwację działania programu. Na podstawie uzyskanych danych, program wyznacza najbardziej wartościową mieszczącą się w plecaku kombinację przedmiotów (przedmioty dostępne w wielu egzemplarzach). Na koniec 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.

**Przykładowy wynik programu:**

Program rozwiązuje dyskretny problem plecakowy (przedmioty dostępne w wielu egzemplarzach) dla wprowadzonych danych.

Gdzie wypisać wyniki? 1 - na ekran 2 - do pliku **1**

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

Podaj pojemność plecaka: **17**

Podaj liczbę przedmiotów (n): **5**

Podaj wagę przedmiotu A: **3**

Podaj wartość przedmiotu A: **4**

Podaj wagę przedmiotu B: **4**

Podaj wartość przedmiotu B: **5**

Podaj wagę przedmiotu C: **7**

Podaj wartość przedmiotu C: **10**

Podaj wagę przedmiotu D: **8**

Podaj wartość przedmiotu D: **11**

Podaj wagę przedmiotu E: **9**

Podaj wartość przedmiotu E: **13**

k 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

j=1

cost[k] 0 0 4 4 4 8 8 8 12 12 12 16 16 16 20 20 20

best[k] \_ \_ A A A A A A A A A A A A A A A

j=2

cost[k] 0 0 4 5 5 8 9 10 12 13 14 16 17 18 20 21 22

best[k] \_ \_ A B B A B B A B B A B B A B B

j=3

cost[k] 0 0 4 5 5 8 10 10 12 14 15 16 18 20 20 22 24

best[k] \_ \_ A B B A C B A C C A C C A C C

j=4

cost[k] 0 0 4 5 5 8 10 11 12 14 15 16 18 20 21 22 24

best[k] \_ \_ A B B A C D A C C A C C D C C

j=5

cost[k] 0 0 4 5 5 8 10 11 13 14 15 17 18 20 21 23 24

best[k] \_ \_ A B B A C D E C C E C C D E C

Koniec. Wyniki:

Plecak zawiera najbardziej wartościową mieszczącą się kombinacje przedmiotów:

Przedmioty w plecaku: C C A

Wartość tych przedmiotów: 24

Zajęte miejsce w plecaku: 17/17

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

Podaj pojemność plecaka: 13

Podaj liczbę przedmiotów (n): 4

Podaj maksymalna wartość jednego elementu: 5

Podaj maksymalna wagę jednego elementu: 6

Wygenerowałem 4-przedmioty:

Indeks elementu: waga ; wartość

A: 3 ; 3

B: 1 ; 2

C: 2 ; 5

D: 4 ; 3

Aby kontynuować, naciśnij dowolny klawisz . . .

k 1 2 3 4 5 6 7 8 9 10 11 12 13

j=1

cost[k] 0 0 3 3 3 6 6 6 9 9 9 12 12

best[k] \_ \_ A A A A A A A A A A A

j=2

cost[k] 2 4 6 8 10 12 14 16 18 20 22 24 26

best[k] B B B B B B B B B B B B B

j=3

cost[k] 2 5 7 10 12 15 17 20 22 25 27 30 32

best[k] B C C C C C C C C C C C C

j=4

cost[k] 2 5 7 10 12 15 17 20 22 25 27 30 32

best[k] B C C C C C C C C C C C C

Koniec. Wyniki:

Plecak zawiera najbardziej wartościową mieszczącą się kombinacje przedmiotów:

Przedmioty w plecaku: C C C C C C B

Wartość tych przedmiotów: 32

Zajęte miejsce w plecaku: 13/13

**Pseudokod:**

**Wypisz na ekran:** "Program rozwiązuje dyskretny problem plecakowy (przedmioty dostępne w wielu egzemplarzach) 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[], best[], cost[]**

**Wypisz na ekran:** "Skąd wziąć dane? 1-wygeneruj losowe 2-pozwól 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**

**Dla** j:=1 **do** n **wykonuj:**

**Dla** i:=1 **do** pojemnosc **wykonuj:**

**Jeżeli** i >= wagi[j]**:**

**Jeżeli** cost[i] < cost[i-wagi[j]]+wartosci[j]**:**

cost[i] = cost[i-wagi[j]]+wartosci[j];

best[i] = j;

**Utwórz wartość minimalna i przypisz do niej wartość najmniejszej wagi ze wszystkich przedmiotów.**

**Wypisz na ekran:** "Plecak zawiera najbardziej wartościową mieszczącą się kombinację przedmiotów. \n"

**Wypisz na ekran:** "Przedmioty w plecaku: "

int p:=pojemnosc;

int waga:=0;

**Dopóki** p-minimalna >= 0 **wykonuj:**

**Wypisz na ekran:** best[p] **<<** " "**;**

waga += wagi[best[p] - 'A' + 1];

p -= wagi[best[p] - 'A' + 1];

**Wypisz na ekran:** "Wartość tych przedmiotów: " **<<** cost[pojemnosc] **<<** " \n"

**Wypisz na ekran:** "Zajęte miejsce w plecaku: " **<<** waga **<<** "/" **<<** pojemnosc **<<** " \n"

**Wnioski:** Na podstawie analizy wyników zwracanych przez program i obliczeń wykonanych poza nim, można stwierdzić, iż program działa i prawidłowo rozwiązuje dyskretny problem plecakowy (przedmioty dostępne w wielu egzemplarzach) dla podanych przez użytkownika danych, korzystając z algorytmu programowania dynamicznego. A(i) jest zdefiniowane rekurencyjnie: A(0) = 0; A(i) = max { cj + A(i − wj) : wj ≤ i }

Obliczenie wyników kolejno dla A(0), A(1)... aż do A(W) pozwala obliczyć wynik.

**Literatura:**

* *Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein*: **Wprowadzenie do algorytmów.** Warszawa: Wydawnictwa Naukowo-Techniczne, 2003.
* *Zbigniew Kokosiński* **wykład i prezentacja 5** Metody programowania, Politechnika Krakowska 2017.
* **pl.wikipedia.org/wiki/Problem\_plecakowy**