/tikz/,/tikz/graphs/
conversions/canvas coordinate/.code=1 , conversions/coordinate/.code=1

layered
trees

1 Knapsack Problem - Problem plecakowy

Problem plecakowy jest zagadnieniem z zakresu optymalizacji. Problem ten swoją nazwę wziął z analogii do rzeczywistego problemu pakowania plecaka. Rozwiązując go zarówno w praktyce jak i teorii trzeba zachować reguły określające ładowność plecaka dotyczące objętości i nośności plecaka. "Knapsack Problem" zaczął być intensywnie badany po pionierskiej pracy Dantziga[?] w późnych latach 50 XX wieku. Znalazł on natychmiast zastosowanie w przemyśle oraz w zarządzaniu finansami. Z teoretycznego punktu widzenia, problem plecakowy często występuję jako relaksacja różnorodnych problemów programowania całkowitego[?].

1.1 Zastosowanie

Problem plecakowy stosowany jest nie tylko w sytuacji wynikającej bezpośrednio z nazwy. Znajduje on zastosowanie w wielu dziedzinach życia oraz nauki. Diffi i Helman[?] w 1976 roku oraz Merkle i Helman[?] w 1978 roku zaproponowali problem plecakowy jako podstawę do enkrypcji kluczy prywatnych. Jednakże klucze oparte na tym algorytmie w latach późniejszych zostały złamane przez środowisko kryptograficzne i jego miejsce zajęły standardy które są bardziej odporne na złamanie (przykładowo XTR).

"Knapsack Problem" jest stosowany również podczas załadunku kontenerów służacych do przewozu materiałów drogą morską. Ładowność oraz gabaryty ładowanych elementów są ograniczane przez budowę i wytrzymałość kontenera.

Problem ten stosowany jest również w dziedzinie finansów. Jest on podstawowym narzędziem do optymalizacji portfela inwestycyjnego. Poprzez uogólnienie i modyfikacje problemu plecakowego zjawiska ekonomiczne mogą być modelowane z większą dokładnością. Przykładowo możliwe jest zakupienie 0, 1, 2 lub więcej akcji inwestycyjnych, a zakup kolejnych akcji może przynieść obniżenie przychodu.

Wiele problemów związanych z planowaniem może być przyrównana do problemu plecakowego, dla przykładu czas wykonywania operacji na maszynie jest zasobem deficytowym. Jest on szczególnie uwydatniony gdy od aktywności maszyny zależy zysk przedsiębiorstwa. Poprzez rozwiązanie problemu plecakowego możliwe jest przewidzenie zapotrzebowania na materiały podaczas procesu tak aby warunki zamówienia zostały spełnione[?].

Kolejnym zagadnieniem wynikającym z problemu plecakowego jest problem optymalnego rozkroju, zostanie on przedstawiony w rozdziale sec:cuttingStockProblem.

1.2 Różnorodność problemu plecakowego

Wszystkie elementy z rodziny tego problemu wymagają pewnego zestawu elementów które mogą zostać wybrane w taki sposób aby zysk został zmaksymalizowany, a pojemość placaka lub wielu plecaków nie została przekroczona. Wszystkie typy problemu należą do rodziny problemów $NP-trudnych$ co oznacza, że mozliwe jest rozwiązanie problemu z użyciem algorytmów wielomianowych. Możliwe są różne warinaty problemu zależne od rozmieszczenia elementów oraz ilości plecaków[**?**]:

- *Problem plecakowy 0-1* - każdy element może być wybrany tylko raz. Problem polega na wyborze n elementów dla których suma zysków p_j jest największa, bez konieczności osiągnięcia całkowitej pojemności c. Może być sformułowany jako problem maksymalizacji:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c, \\ & && x_j \in \{0,1\}, \qquad j=1,\ldots,n \end{aligned} \tag{1}$$

  gdzie x_j jest wartością binarną. Jeżeli $x_j = 1$ wtedy j-ty element powinien znaleźć się w plecaku, w innym przypadku $x_j = 0$.

- *Ograniczony problem plecakowy* - każdy element może być wybrany ograniczoną ilość razy. Zmianą w obecnym problemie względem problemu 0-1 jest ograniczona m_j ilość elementów j:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c, \\ & && x_j \in \{0,1\ldots,m_j\}, \quad j=1,\ldots,n \end{aligned} \tag{2}$$

- *Nieograniczony problem plecakowy* - jest rozszerzeniem problemu ograniczonego o nielimitowaną liczbę dostępnych elementów:

$$\begin{aligned} &\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, \\ &\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c, \\ & && x_j \in N_0, \qquad j=1,\ldots,n \end{aligned} \tag{3}$$

  Każda zmienna x_j w metodzie niograniczonej zostanie ograniczona poprzez pojemność c, gdy waga każdego z elementów jest równa przynajmniej jeden. W ogólnym przypadku transformacja problemu nieograniczonego w ograniczony nie przynosi korzyści

- *Problem plecakowy wielokrotnego wyboru* - elementy powinny być wybierane z klas rozłącznych. Problem ten jest generalizacją problemu 0-1. Możliwy jest wybór dokładnie jednego elementu j z każdej grupy N_i, $i = 1, \ldots, k$:

$$\begin{aligned}
&\text{maksymalizacja} && \sum_{i=1}^{k}\sum_{j\in N_i} p_{ij}x_{ij}, && \\
&\text{w odniesieniu do} && \sum_{i=1}^{k}\sum_{j\in N_i} w_{ij}x_{ij} \le c, && \\
& && \sum_{j\in N_i} x_{ij} = 1, && i = 1,\ldots,k, \\
& && x_j \in \{0,1\}, && i = 1,\ldots,k, \quad j \in N_i.
\end{aligned} \tag{4}$$

  Zmienna binarna $x_{ij} = 1$ określa że j-ty element został wybrany z i-tej grupy. Ograniczenie $\sum_{j\in N_i} x_{ij} = 1, \quad i = 1,\ldots,k$ wymusza wybór dokładnie jednego elementu z każdej grupy.

- *Wielokrotny problem plecakowy* - możliwość wypełnienia wielu plecaków. Jeśli jest możliwość załadowania n elmentów do m pleckaów o różnych pojemnościach c_i w taki sposób że zysk będzie jak największy:

$$\begin{aligned}
&\text{maksymalizacja} && \sum_{i=1}^{k}\sum_{j\in N_i} p_{ij}x_{ij}, && \\
&\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_{ij} \le c_i, && i = 1,\ldots,m \\
& && \sum_{j\in N_i} x_{ij} \le 1, && i = 1,\ldots,k, \\
& && x_j \in \{0,1\}, && i = 1,\ldots,m, \quad j = 1,\ldots,n.
\end{aligned} \tag{5}$$

  Zmienna $x_{ij} = 1$ określa że j-ty element powinien zostać umiesczony w i-tym plecaku, podczas gdy ogranicznie $\sum_{j=1}^{n} w_{ij}x_{ij} \le c_i$ zapewnia że restrykcja dotycząca pojemności plecaka zostanie zachowana. Ogranicznie $\sum_{j\in N_i} x_{ij} \le 1$ zapewnia że każdy element zostanie wybrany tylko raz.

- *Bin-packing problem* - bardzo często spotykana wersja problemu plecakowego. Problem ten polega na umieszczeniu n elementów w jak najmniejszej liczbie opakowań:

$$
\begin{aligned}
&\text{maksymalizacja} && \sum_{i=1}^{n} y_i && \\
&\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_{ij} \leq c y_i, && && i = 1, \ldots, n, \\
& && \sum_{i=1}^{n} x_{ij} = 1, && && j = 1, \ldots, n, \\
& && y_i \in \{0, 1\}, && && i = 1, \ldots, n, \\
& && x_{ij} \in \{0, 1\} && i = 1, \ldots, m, && j = 1, \ldots, n,
\end{aligned}
\tag{6}
$$

gdzie y_i określa czy i-te opakowanie zostało użyte, a x_{ij} stanowi czy j-ty element powinen zostać umieszczony w i-tym opakowaniu

- *Welokrotnie ograniczony problem plecakowy* - najbardziej ogólny typ, który jest problemem programowania całkowitego z dodatnimi współczynnikami:

$$
\begin{aligned}
&\text{maksymalizacja} && \sum_{j=1}^{n} p_j x_j, && \\
&\text{w odniesieniu do} && \sum_{j=1}^{n} w_j x_j \leq c_i, && i = 1, \ldots, m, \\
& && x_j \in N_0, && j = 1, \ldots, n.
\end{aligned}
\tag{7}
$$

1.3 Możliwe rozwiązania

Problem plecakowy należy do grupy problemów $\mathcal{NP}$-Trudnych. Rozwiązanie problemów z tej grupy jest co najmniej tak trudne, jak rozwiązanie każdego problemu z całej klasy $\mathcal{NP}$. Problem $\mathcal{NP}$-Trudny to problem obliczeniowy dla którego znalezienie rozwiązania problemu możliwe jest z wielomianową złożonością obliczeniową. Problemy $\mathcal{NP}$-Trudne obejmują zarówno problemy decyzyjne jak również problemy przeszukiwania czy też problemy optymalizacyjne.

Rozwiązanie problemu plecakowego jest możliwe przy użyciu różnych metod:

- *Metoda podziału i ograniczeń* - Metoda ta często jest stosowana do problemu plecakowego od momentu gdy Kolesar [?] zaprezentował pierwszy algorytm w 1967 roku.

- *Programownaie dynamiczne* - Gdy zostaną dodane warunki brzegowe wtedy algorytm ten staje się "zaawansowaną" formą metody podziału i ograniczeń.

- *Relaksacja przestrzeni stanów* - relaksacja programowania dynamicznego gdzie współczynniki są skalowane przez pewną stałą wartość.

1.3.1 Metoda podziału i ograniczeń

Algorytm ten polega na wypisaniu wszystkich możliwych rozwiązań używając struktury drzewa. Algorytm przechodzi kolejno po gałęziach które reprezentują podzbiory rozwiązania. Każda gałąź jest sprawdzana zadanymi warunkami brzegowymi i zostaje odrzucona jeśli nie poprawia rozwiąznaia. Przedstawione zostanie rozwiązanie nieograniczonego problemu plecakowego (unboundedKnapsack) [?]. Współczynniki $w_1, \ldots, w_m$, $p_1, \ldots, p_m$ oraz c są nieujemne. Stosunek p_j/w_j jest wartością jednej jednostki długości j-tego elementu. Stosunek ten jest nazywany *wydajnością* zmiennej x_j. Pierwszym krokiem algorytmu jest posortowanie zmiennych w porządku malejącym względem wydajności:

$$p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_m/w_m \tag{8}$$

Dla posortowanych elementów każde rozwiązanie optymalne (unboundedKnapsack) spełnia warunek:

$$c - \sum_{j=1}^{m} w_j x_j < w_m \tag{9}$$

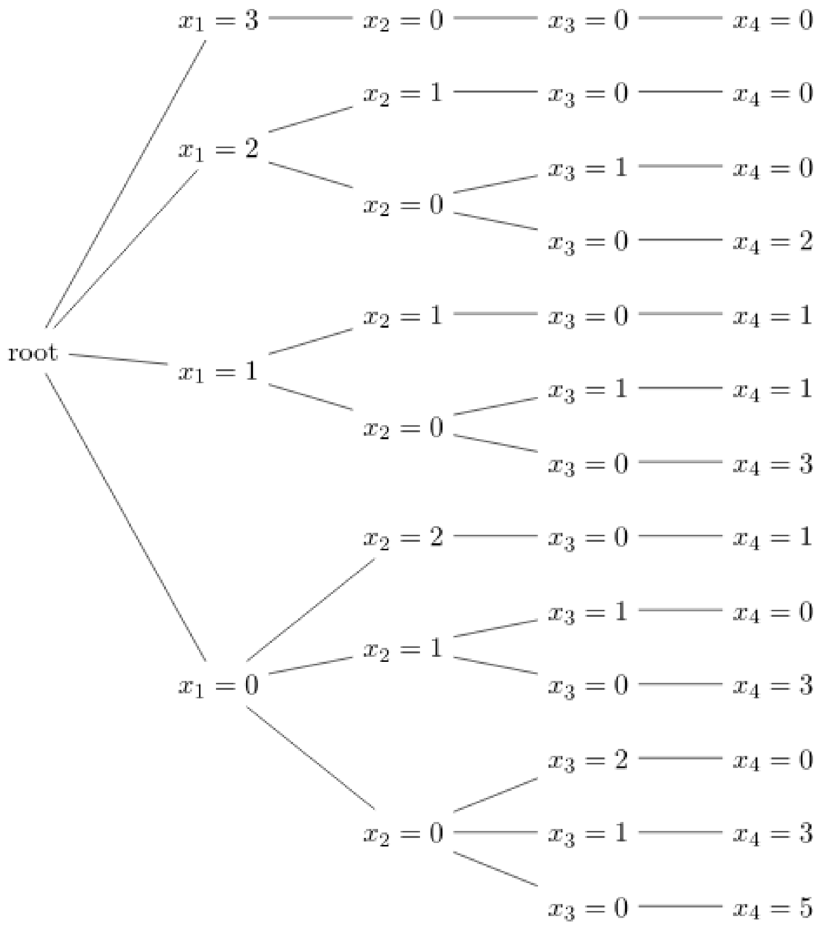
Głównym elementem algorytmu jest stworzenie drzewa wyliczeń oraz przeprowadzenie jego redukcji. Przykładowo dla problemu który zawiera 13 rozwiązań:

$$\begin{array}{ll} \text{maksymalizacja} & 4x_1 + 5x_2 + 5x_3 + 2x_4 \\ \text{w odniesieniu do} & 33x_1 + 49x_2 + 51x_3 + 22x_4 \leq 120 \\ & x_j \in N_0 \end{array}$$

drzewo będzie miało 13 liści (fig:chvatalBBTree). Jeśli dany węzeł posiada więcej niż jedno dziecko, wówczas potomek o większej przechowywanej wartości zostaje umieszczony wyżej. Każdy następny węzeł jest obliczany według wzoru:

$$\begin{aligned} x_j &= \lfloor (c - \sum_{i=1}^{j-1} w_i x_i)/w_j \rfloor \qquad i = 1, 2, \ldots, m \\ x_1 &= \lfloor c/w_1 \rfloor \end{aligned}$$

Podczas poszukiwania węzłów które nie mogą polepszyć rozwiązania i gałęzi które dają szansę na rozwiązanie optymalne $x_1, x_2, \ldots, x_m$ ustawione zostaje $k = m - 1$. Jeśli zachodzi taka potrzeba zmienna k jest dekrementowana



Rysunek 1: Drzewo wyliczeń możliwych rozwiązań

dopóki nie zostanie znalezione takie x_k, że $x_k > 0$. Wówczas $x_k = x_{k-1}$, a wartości $x_{k+1}, x_{k+2}, \ldots, x_m$ są otrzymywane ze wzoru (eq:chvatalBBTree).

Dla bieżącego rozwiązania $x_1^*, \ldots, x_m^*$ zachodzi $\sum_{i=1}^m p_i x_i^* = M$. Maksymalne k takie, że $k \leq m-1$ oraz $x_k > 0$ zostaje określone przechodząc od węzłów $x_1, x_2, \ldots, x_m$ w kierunku korzenia. Podobnie jak wcześniej, niech $\bar{x}_i = x_i$ dla $i = 1, 2, \ldots, k-1$ oraz $\bar{x}_k = x_k - 1$ będą zmiennymi kandydującymi do rozwiązania. Aby okreslić czy $\bar{x}_i$ polepszy rozwiązanie x_i^*. Zgodnie z (eq:BBeff) dla każdej zmiennej $x_{k+1}, x_{k+2}, \ldots, x_m$ wydajność wynosi maksymalnie p_{k+1}/w_{k+1}, tak więc

$$\sum_{i=k+1}^{m} p_i \bar{x}_i \leq \frac{p_{k+1}}{w_{k+1}} \sum_{i=k+1}^{m} w_i \bar{x}_i$$

połączone razem z (unboundedKnapsack) zwraca:

$$\sum_{i=1}^{m} p_i \bar{x}_i \leq \sum_{i=1}^{m} w_i \bar{x}_i + \frac{p_i}{w_i}(c - \sum_{i=1}^{k} w_i \bar{x}_i). \tag{10}$$

Zgodnie z zasadami drzewa wyliczeń, nierówność

$$\sum_{i=1}^{k} p_x \bar{x}_i + \frac{p_{k+1}}{w_{k+1}}(c - \sum_{i=1}^{k} w_i \bar{x}_i) \leq M \tag{11}$$

określa że ścieżka $\bar{x}_1, \ldots, \bar{x}_k$ jest niegorsza niż pozostałe. Jeśli wszystkie współczynniki $p_1, \ldots, p_m$ są dodatnimi liczbami całkowitymi, wówczas również M jest liczbą całkowitą, a słaba nierówność (2.11Metoda podziału i ograniczeńequation.2.11) może zostać zastąpiona mocną

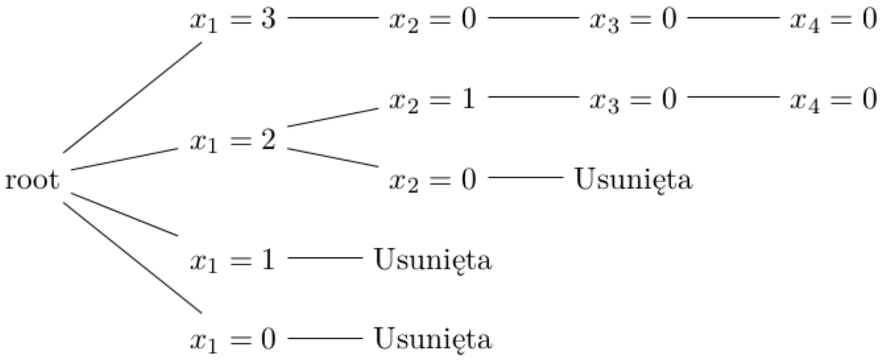
$$\sum_{i=1}^{k} p_x \bar{x}_i + \frac{p_{k+1}}{w_{k+1}}(c - \sum_{i=1}^{k} w_i \bar{x}_i) < M + 1 \tag{12}$$

Dla wcześniejszego przykładu powyższy krok mający na celu redukcję drzewa przyjmuje postać:

$$\begin{aligned} x_1 &= \lfloor 120/33 \rfloor = 3 \\ x_2 &= \lfloor (120-99)/49 \rfloor = 0 \\ x_3 &= \lfloor (120-99)/51 \rfloor = 0 \\ x_4 &= \lfloor (120-99)/22 \rfloor = 0 \end{aligned}$$

Z powyższego wynika że początkowe rozwiązanie to $x_1^* = 3, x_2^* = x_3^* = x_4^* = 0$ oraz $M = 12$. Początkowo $k = 3$, następnie wystepuje redukcja k dopóki nie zostanie znalezione takie $k = 1$ dla którego istnieje $x_k > 0$. Wówczas $x_1 = 3$ zostaje zaminione na $x_1 = 2$. Przed sprawdzeniem gałęzi $x_1 = 2$ przeprowadzony zostaje test (2.12Metoda podziału i ograniczeńequation.2.12) z $k = 1$ oraz $\bar{x}_1 = 2$. Wówczas lewa strona nierówności wynosi

$$8 + \frac{5}{49}(120 - 66) = 13.5$$



Rysunek 2: Drzewo wyliczeń możliwych rozwiązań

i jest nie mniejsza niż $M + 1 = 13$ z czego wynika że gałąź może być warta sprawdzenia. Następnie zostaje obliczona kolejna ścieżka

$$\begin{aligned} x_2 &= \lfloor (120 - 66)/49 \rfloor = 1 \\ x_3 &= \lfloor (120 - 115)/51 \rfloor = 0 \\ x_4 &= \lfloor (120 - 115)/22 \rfloor = 0 \end{aligned}$$

i zastąpuje ona poprzednie rozwiązanie $x_1^* = 2, x_2^* = 1, x_3^* = x_4^* = 0$ oraz $M = 13$. Powtórzony zostaje krok z redukcją $k = 3$ dopóki nie zostanie znalezione takie $k = 2$ dla którego istanieje $x_k > 0$. Wówczas $x_2 = 1$ zostaje zamienione na $x_2 = 0$. Aby określić czy ścieżka $x_1 = 2, x_2 = 0$ jest warta sprawdzenia, zostaje przeprowadzony test (2.12Metoda podziału i ograniczeńequation.2.12) z $k = 2$ oraz $\bar{x}_1 = 2, \bar{x}_2 = 0$. Lewa strona nierówności wynosi

$$8 + \frac{5}{51}(120 - 66) = 13.3$$

Jest ona mniejsza niż $M + 1 = 14$, więc gałąź ta jest odcinana. Następnie k dalej jest zmniejszane, a kroki sa powtarzane. Dla $x_1 = 1$ wynik testu to $12.9 < 14$, a dla $x_1 = 0$ wynik to $12.2 < 14$ więc gałęzie te są odcinane. Tak więc optymalnym rozwiązaniem jest $x_1^* = 2, x_2^* = 1, x_3^* = x_4^* = 0$. Drzewo wyliczeń zostało zredukowane do postaci 2.2Drzewo wyliczeń możliwych rozwiązańfigure.2.2.

Jeśli odcięta jest gałąź $\bar{x}_1, \ldots, \bar{x}_k$ wówczas odcięt również zostaje pozostała część gałęzi bez przeprowadzania dodatkowych testów.

Algorytm dla metody podziału i ograniczeń do rozwiązania problemu plecakowego, został przedstawiony poniżej

Algorytm 1 Metoda podziału i ograniczeń - problem plecakowy

1: M := 0
2: k := 0
3: **for** j := k+1 TO m **do**
4: $\quad x_j = \lfloor (c - \sum_{i=1}^{j-1} w_i x_i)/w_j \rfloor$
5: k := m
6: **if** $\sum_{i=1}^{m} p_i x_i > M$ **then**
7: $\quad$ M := $\sum_{i=1}^{m} p_i x_i$
8: $\quad$ **for** j := 1 TO m **do**
9: $\qquad x_j^* = x_j$
10: **if** $k = 1$ **then**
11: $\quad$ stop
12: **else**
13: $\quad k = k - 1$
14: **if** $x_k = 0$ **then**
15: $\quad$ idź do linii 10Metoda podziału i ograniczeńfigure.2.2
16: **else**
17: $\quad x_k = x_k - 1$
18: **if** $!\sum_{i=1}^{k} p_x \bar{x}_i + \frac{p_{k+1}}{w_{k+1}}(c - \sum_{i=1}^{k} w_i \bar{x}_i) < M + 1$ **then**
19: $\quad$ idź do linii 3Metoda podziału i ograniczeńfigure.2.2
20: **else**
21: $\quad$ idź do linii 10Metoda podziału i ograniczeńfigure.2.2

1.3.2 Programowanie dynamiczne

Metoda ta używana jest w przypadku gdy problem można podzielić na małe podproblemy które mogą zostać rozwiązane rekursywnie. Rozwiązanie optymalne podproblemu jest również optymalnym rozwiązaniem problemu głównego. Przedstawione zostanie rozwiązanie problemu plecakowego rodzaju 0-1 [?].

Jeśli elementy są oznaczone jako $1, \ldots, n$ wtedy podproblem będzie odpowiedzialny za znalezienie optymalnego rozwiązania dla $S_k = \{1, 2, \ldots, k\}$. Niemożliwe jest opisanie rozwiązania końcowego S_n na podstawie podproblemów S_k. Rekursywne sformułowanie podproblemu:

$$B[k,w] = \begin{cases} B[k-1,w] & \text{jeśli} \quad w_k > w, \\ max\{B[k-1,w], B[k-1,w-w_k] + b_k\} & \text{jeśli} \quad w_k \leq w. \end{cases} \tag{13}$$

Z powyższego równania wynika że najlepszy podzbiór podproblemu S_k z całkowitą wagą w jest najlepszym podzbiorem dla S_{k-1} którego całkowita waga wymosi w lub jest najlepszym podzbiorem dla S_{k-1} którego całkowita waga wynosi $w - w_k$ plus k-ty element. Złożoność programowania dynamicznego to $O(n * W)$. Algorytm jako dane wejściowe przyjmuje maksymalną wartość ciężaru W, oraz dwie listy: listę wag $w_1, \ldots, w_n$ oraz odpowiadającą jej listę zysku $b_1, \ldots, b_n$.

Algorytm 2 Programowanie dynamiczne - problem plecakowy 0-1

for w := 0 TO W **do**
    B[0,w] := 0
for i := 1 TO n **do**
    B[i,0] := 0
for i := 1 TO n **do**
    for w := 0 TO W **do**
        if $w_i \leq w$ **then**
            if $b_i + B[i-1, w-w_i] > B[i-1, w]$ **then**
                $B[i,w] := b_i + B[i-1, w-w_i]$
            else
                $B[i,w] := B[i-1,w]$
        else
            $B[i,w] := B[i-1,w]$