Programowanie dynamiczne

Sprawozdanie z laboratorium 9 – Piotr Sarna LK1

Cel ćwiczenia

Podczas zajęć zapoznaliśmy się z techniką programowania dynamicznego. Następnie wykorzystaliśmy ją rozwiązania problemu plecakowego.

Wstęp teoretyczny

Programowanie dynamiczne to technika rozwiązywania problemów, poprzez rozwiązanie problemu dla mniejszego zestawu danych, zapamiętaniu go, a następnie wykorzystaniu go przy wyznaczaniu powiększonego zestawu danych, do momentu rozwiązania problemu dla całego zestawu danych.

Jednak aby rozwiązać problem techniką programowania dynamicznego, problem musi być sformułowany rekurencyjnie. Dodatkową wadą jest fakt, że w celu rozwiązania problemu musimy zająć dodatkową pamięć, na tablicę przechowującą wyniki dla mniejszych zestawów danych.

Jeśli jednak możemy rozwiązać problem techniką programowania dynamicznego, możemy wykorzystać jego następujące zalety:

* Problemy o strukturze rekurencyjnej podproblemów można tą metodą rozwiązać w czasie wielomianowym.
* Po wyznaczeniu rozwiązań wszystkich podproblemów czas wyznaczenia głównego problemu jest liniowy.

Programowanie dynamiczne możemy wykorzystać do rozwiązania następujących problemów:

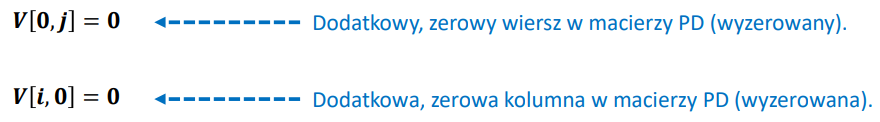
* Wyznaczenie n-tego wyrazu ciągu Fibonacciego
* Rozwiązanie problemu plecakowego
* Wyznaczenie wartości silni
* Wyznaczenie współczynnika dwumiennego
* Problem podziału zbioru

Opis algorytmu

Problem plecakowy polega na wyznaczeniu takiego ułożenia elementów w plecaku przedmiotów (każdy o określonej wadze i wartości), aby uzyskać maksymalną wartość plecaka nie przekraczającą określonej maksymalnej wagi plecaka.

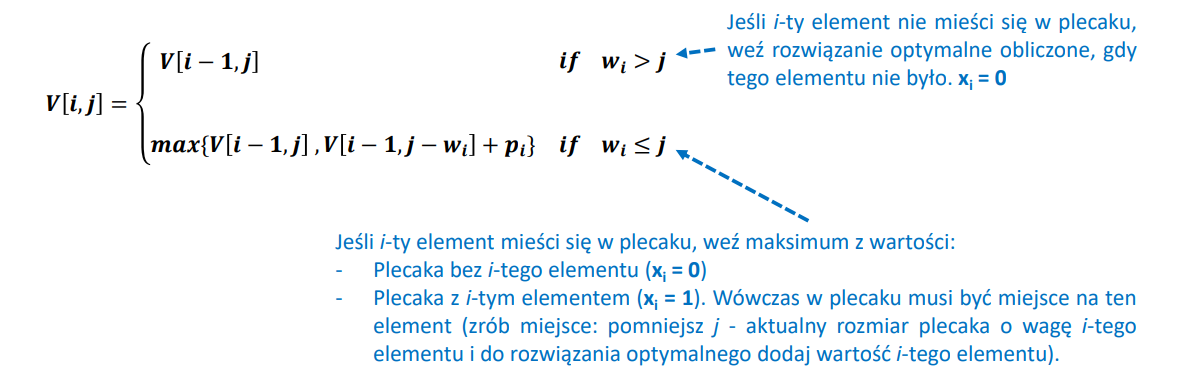
W rozwiązaniu tego problemu, skorzystamy z macierzy kosztów. Będzie ona przechowywać informacje o najwyższej możliwej wartości plecaka, dla danego podproblemu.

Podproblemem będzie plecak o mniejszej pojemności, oraz zmniejszony zestaw przedmiotów. Dzięki temu wyznaczymy maksymalne ułożenie elementów w plecaku dla każdej pojemności i każdego zestawu elementów, na bazie czego zbudujemy ostateczny wynik.



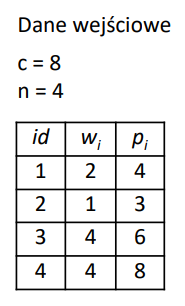
Wiersze i kolumny o macierzy indeksie „0” pozostają wyzerowane, ponieważ nie będziemy z nich korzystać.

Dzieje się tak, ponieważ iterację zaczynamy od „1”. 1 to minimalna możliwa maksymalna waga oraz minimalna możliwa liczba wykorzystanych elementów.



Gdzie oznacza wagę elementu i, a oznacza jego wartość.

Ostatnia komórka macierzy będzie zawierała rozwiązanie głównego problemu:

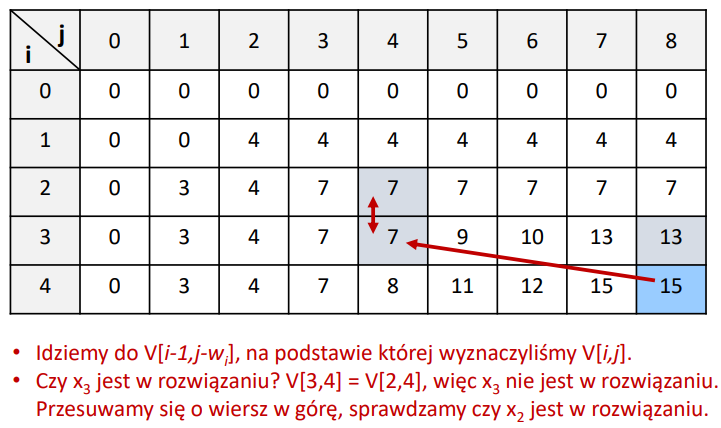


Aby odczytać, które przedmioty zostały zapakowane do plecaka, ustawiamy się w ostatniej komórce macierzy, i porównujemy jej wartość z komórką bezpośrednio wyżej. Jeśli komórka wyżej ma mniejszą wartość, oznacza to że element o indeksie wiersza macierzy, w którym się znajdujemy jest w plecaku.

Jeśli tak jest, przesuwamy się o wiersz w górę, oraz o kolumn w lewo i dokonujemy tego samego porównania.

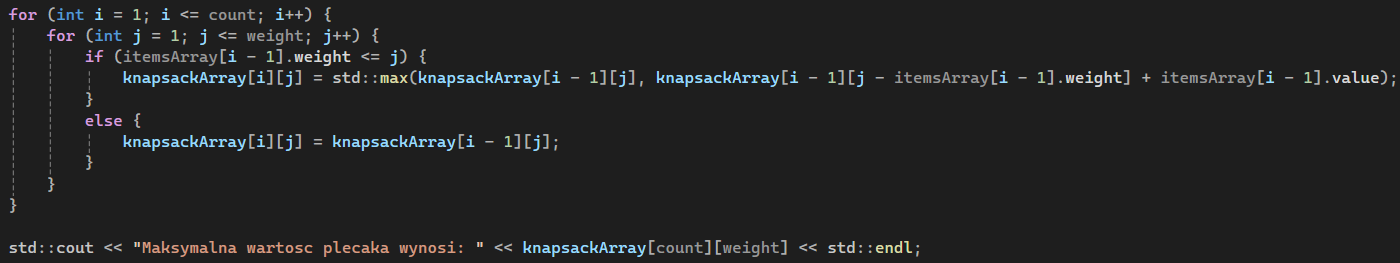
W przeciwnym wypadku przesuwamy się o wiersz w górę i wykonujemy to samo porównanie.

Czynność powtarzamy tak długo, aż dojdziemy do najmniejszego indeksu macierzy, co oznacza przejście przez jej wszystkie elementy lub aż waga plecaka, którą zmniejszamy po każdym napotkaniu elementu w plecaku będzie równa 0, co będzie oznaczało wypakowanie wszystkich jego elementów.

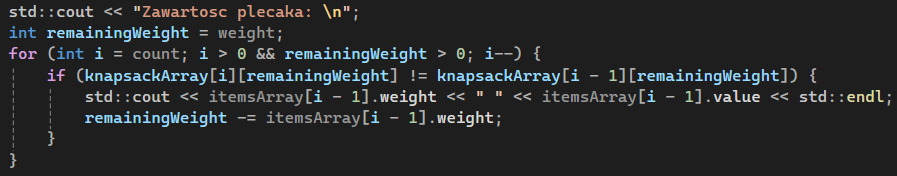


Implementacja rozwiązania problemu plecakowego poprzez technikę programowania dynamicznego w C++.

Znajdywanie maksymalnej wartości plecaka:

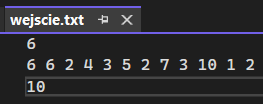


Oraz wypisywanie elementów plecaka o najwyższej wartości:

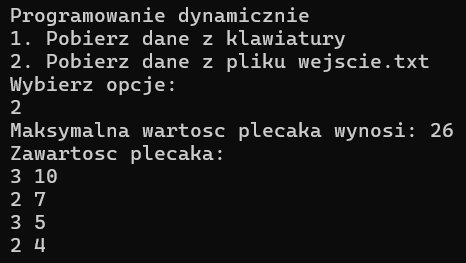


Prezentacja działania mojej implementacji w C++

Dla następujących danych:



Otrzymujemy następujące wyniki:



Wnioski

Algorytm cechuje złożoność czasowa oraz pamięciowa O(p \* w)

W celu rozwiązania problemu musimy przejść przez wszystkie podproblemy dla każdej kombinacji ilości elementów „p”, oraz wagi „w”, w wyniku czego złożoność jest ilorazem ilości elementów oraz maksymalnej wagi plecaka.

Bibliografia

<https://www.cs.put.poznan.pl/arybarczyk/TeoriaAiSD3.pdf>

<https://www.cs.put.poznan.pl/mszachniuk/mszachniuk_files/lab_aisd/Szachniuk-ASD-t5.pdf>