Problemy optymalizacyjne

1 Przykłady

Problem plecakowy (Knapsack) Dany jest plecak o określonej pojemności W oraz zbiór przedmiotów o określonej wartości v_i i wadze w_i. Celem jest wybranie zestawu przedmiotów o jak największej sumarycznej wartości, który może zmieścić się w plecaku.

Problem komiwojażera (Travelling Salesman) Dany jest zbiór miast z określonymi odległościami między każdą parą. Celem jest znalezienie najkrótszej ścieżki, która odwiedza każde miasto i wraca do miasta początkowego.

2 Metody

2.1 Brute Force

Przestrzeń rozwiązań jest przeszukiwana sekwencyjnie. Ta metoda zawsze daje optymalne rozwiązanie, ale często jest niemożliwa do zastosowania ze względu na rozmiar przestrzeni rozwiązań.

2.2 Podejście zachłanne

Rozwiązanie jest konstruowane przez dodawanie kolejno elementów, które lokalnie najbardziej zwiększają jakość rozwiązania (mierzoną funkcją celu). Często istnieje wiele algorytmów zachłannych dla danego problemu.

Zadania

Zadanie 1. Problem plecakowy

Dany jest plecak o pojemności W oraz zestaw przedmiotów:

w_i	2	3	5	7	1	4	1
v_i	10	5	15	7	6	18	3

$W = 15$

w_i	1	3	4	3	3	1	5	10
v_i	2	9	3	8	10	6	4	10

$W = 12$

- Jaka jest przestrzeń rozwiązań w problemie plecakowym?
- Zapisz funkcję celu i ograniczenia w problemie plecakowym.
- Ile jest możliwych rozwiązań?

- Zaproponuj algorytm zachłanny i znajdź rozwiązania dla powyższych przykładów. Czy podejście zachłanne jest optymalne?

Zadanie 2. Problem komiwojażera

Dane są następujące macierze sąsiedztwa:

	A	B	C	D
A	0	10	15	20
B	10	0	9	10
C	15	9	0	2
D	20	10	2	0

	A	B	C	D	E
A	0	1	6	8	4
B	1	0	8	5	6
C	6	8	0	9	7
D	8	5	9	0	8
E	4	6	7	8	0

- Jaka jest przestrzeń rozwiązań w problemie komiwojażera?
- Zapisz funkcję celu.
- Ile jest możliwych rozwiązań?
- Zaproponuj podejście zachłanne i znajdź rozwiązania dla powyższych przykładów.

Mini-projekt: problem plecakowy

Celem jest implementacja metody **brute-force** dla problemu plecakowego. Podzbiory należy reprezentować przez wektory binarne i generować kolejno wszystkie takie wektory (od $000\ldots000$ do $111\ldots111$, gdzie '1' na i-tej pozycji oznacza, że i-ty element znajduje się w podzbiorze). Wektory najlepiej zapisywać w zmiennej typu `int` i stosować operacje bitowe.

Program powinien wypisać sumaryczną wartość i wagę optymalnego podzbioru oraz odpowiadający mu wektor. Dane wejściowe znajdują się w pliku `knapsack.txt` (można wkleić je bezpośrednio w kodzie).

Wskazówki:

- W tym projekcie najlepiej zastosować język kompilowany (np. C lub Java) ze względu na wydajność.
- Wektory należy przechowywać w zmiennej 64-bitowej (typ danych `long` (Java) lub `long long` (C)).
- Konwersja `int -> string` lub wypisywanie dużej ilości informacji diagnostycznych znacząco spowolnią działanie programu.