Programowanie dynamiczne

Zadania

Zadanie 1.

Zaproponuj algorytm rekurencyjny dla problemu plecakowego i stwórz jego implementację. Program na wejściu powinien otrzymać tablice `weights` i `values`, liczbę przedmiotów `n` oraz pojemność `W` i zwrócić sumę wartości przedmiotów z optymalnego rozwiązania.

Przykłady:

Input:

```
weights = [1, 3, 3, 1]
values = [3, 8, 4, 7]
n = 4
W = 6
```

Output: `18`

Input:

```
weights = [3, 1, 6, 10, 1, 4, 9, 1, 7, 2, 6, 1, 6, 2, 2, 4, 8, 1, 7, 3, 6,
           2, 9, 5, 3, 3, 4, 7, 3, 5, 30, 50]
values = [7, 4, 9, 18, 9, 15, 4, 2, 6, 13, 18, 12, 12, 16, 19, 19, 10, 16,
          14, 3, 14, 4, 15, 7, 5, 10, 10, 13, 19, 9, 8, 5]
n = 32
W = 75
```

Output: `262`

Zadanie 2.

Algorytm rekurencyjny wielokrotnie rozwiązuje identyczne podproblemy. Można znacząco go przyspieszyć stosując spamiętywanie (memoization), czyli zapamiętywanie raz obliczonych wyników w tablicy.

Stwórz dwuwymiarową tablicę o rozmiarach $(\texttt{n}+1) \times (\texttt{W}+1)$, w której wiersze odpowiadają przedmiotom od `0` do `n`, a kolumny wagom od `0` do `W`. Alternatywą jest zastosowanie słownika (np. Hashtable), w którym kluczem jest para $(\texttt{i}, \texttt{w})$. Zmodyfikuj algorytm z Zadania 1 tak, aby wyniki raz obliczonego problemu były zapisywane w tablicy/słowniku.

Porównaj czas wykonania algorytmów z Zadań 1 i 2.

Zadanie 3.

Stwórz algorytm stosujący programowanie dynamiczne, który wypełni kolejno każdy wiersz w tablicy z Zadania 2, korzystając z rozwiązań we wcześniejszych wierszach.