Zadanie 7 - Oszczędny ogrodnik

Szablon rozwiązania: `zad7k.py`

W sadzie pewnego oszczędnego ogrodnika rosły drzewa owocowe. Jednak ze względu na brak wystarczającej ilości funduszy, nie ma on możliwości podlania wszystkich z nich. Musi wybrać, które drzewa podlać, aby ze sprzedaży owoców z nich zebranych osiągnąć jak największy przychód. (Aby podlać dane drzewo musimy podlać cały jego korzeń, tj. wszystkie jego fragmenty posiadające przynajmniej jeden wspólny bok). Dana jest tablica dwuwymiarowa T o wymiarach NxM która zawiera informacje o tym, czy na danej "współrzędnej" znajduje się korzeń jakiegoś drzewa i jeżeli tak, to ile litrów wody wymaga, aby został poprawnie podlany. Pierwsza współrzędna tablicy T określa głębokość, a druga lokalizację. Ze względów logistycznych ogrodnik posiada księgę, w której zapisane są lokalizacje wszystkich drzew w sadzie. Wyrażona jest poprzez tablicę liczb naturalnych D. Przykładowo, jeżeli D[i] = x, oznacza to, że w punkcie T[0][x] będzie znajdował się fragment korzenia drzewa i-tego. Można założyć, że na głębokości "zerowej" każde drzewo posiada tylko jeden fragment korzenia, oraz, że żadne dwa drzewa nie mają wspólnego korzenia. Księgowa ogrodnika przygotowała także zbiór (wyrażony tablicą liczb naturalnych Z) potencjalnych przychodów, które może osiągnąć ze zbiórki owoców (Tak, że dla drzewa w lokalizacji D[i], potencjalny przychód wynosi Z[i]). Proszę napisać algorytm, który zwróci maksymalny przychód, który ogrodnik może osiągnąć ze zbiorów, zakładając, że posiada on tylko l litrów wody, aby podlać swój ogród.

Algorytm należy zaimplementować jako funkcję postaci:

```
def ogrodnik( T, D, Z, l ):
  …
```

która przyjmuje tablicę dwuwymiarową współrzędnych T, tablicę lokalizacji drzew D, tablicę potencjalnych zysków Z oraz limit litrów wody l.

Przykład. Dla danych:

```
D = [4, 9, 12, 16]
Z = [13, 11, 15, 4]
l = 32
```

Oraz następującej wizualizacji tablicy T (gdzie puste pola to zera, czyli brak korzenia)

				1					5			1				4			
				2					6			2				1			
				1				3	1			2	2	2		2	4	2	
			1	2			1	4	6		2	1	3			3	1		

Wynikiem jest liczba **28**

Testowanie Rozwiązań

Żeby przetestować rozwiązania należy wykonać polecenie: `python3 zad7k.py`