

Algorytmy i Struktury Danych  
Zadanie offline 9 (19. VI 2023)

### Format rozwiązań

**Rozwiązanie zadania musi się składać z krótkiego opisu algorytmu (wraz z uzasadnieniem poprawności) oraz jego implementacji.** Zarówno opis algorytmu jak i implementacja powinny się znajdować w tym samym pliku Pythona (rozszerzenie `.py`). Opis powinien być na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Opis nie musi być długi—wystarczy kilka zdań, jasno opisujących ideę algorytmu. Implementacja musi być zgodna z szablonem kodu źródłowego dostarczonym wraz z zadaniem. Niedopuszczalne jest w szczególności:

1. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
2. wypisywanie na ekranie jakichkolwiek napisów innych niż wypisywane przez dostarczony kod (ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania).

Dopuszczalne jest natomiast:

1. korzystanie z następujących elementarnych struktur danych: krotka, lista, kolejka `collections.deque`, kolejka priorytetowa (`queue.PriorityQueue` lub `heapq`),
2. korzystanie z zaawansowanych struktur danych (np. słowników czy zbiorów),
3. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są),
4. korzystanie z wbudowanych funkcji sortujących (można założyć, że mają złożoność  $O(n \log n)$ ).

Wszystkie inne algorytmy lub struktury danych wymagają implementacji przez studenta. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych w pliku z szablonem rozwiązania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 punktów. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

### Testowanie rozwiązań

Żeby przetestować rozwiązanie zadania należy wykonać polecenie: `python zad9.py`

<b>Szablon rozwiązania:</b>	<code>zad9.py</code>
<b>Złożoność akceptowalna:</b>	$O(n^2)$ , gdzie $n$ to łączna liczba parkingów.
<b>Złożoność wzorcowa:</b>	$O(n \log n)$ , gdzie $n$ to łączna liczba parkingów.

Kierowca ciężarówki przewozi towary z miasta  $A$  do miasta  $B$ . W pewnych miejscach trasy przejazdu znajdują się parkingi. Przejeżdżając obok parkingu kierowca może (ale nie musi) się na nim zatrzymać i odpocząć. Przepisy transportowe narzucają jednak pewne ograniczenia związane z bezpieczeństwem:

1. Maksymalna liczba kilometrów, którą można przejechać bez zatrzymania wynosi  $T$ . Od zasady tej istnieje jeden wyjątek, opisany w punkcie 2.
2. W trakcie całego przejazdu z  $A$  do  $B$  kierowca może jeden raz przekroczyć limit  $T$  kilometrów jazdy bez zatrzymania. Może wówczas przejechać nie więcej niż  $2T$  kilometrów bez zatrzymania.

Niestety, parkingi na trasie są płatne. Co więcej, opłaty za postój różnią się pomiędzy parkingami. Kierowca musi więc wybrać miejsca postoju w taki sposób, by przejechać trasę zgodnie z obowiązującymi przepisami i równocześnie zapłacić możliwie jak najmniej za postoje.

Zaproponuj i zaimplementuj algorytm, który wylicza minimalny koszt przejechania z miasta  $A$  do miasta  $B$  zgodnie z opisanymi przepisami transportu towarów. Koszt przejazdu z  $A$  do  $B$  definiujemy jako sumę opłat za parkowanie w miejscach, w których kierowca się zatrzymał (nie liczymy ceny paliwa; nie bierzemy pod uwagę czasu postoju na parkingu). W miastach  $A$  i  $B$  opłata nie jest pobierana. Uzasadnij poprawność zaproponowanego algorytmu i oszacuj jego złożoność obliczeniową.

Algorytm należy zaimplementować jako funkcję:

```
def min_cost( O, C, T, L )
```

Argumentami funkcji są:

- Tablica  $O$  zawierająca pozycje parkingów na trasie z  $A$  do  $B$ .  $O[i]$  to liczba kilometrów (wzdłuż trasy przejazdu) od  $A$  do  $i$ -go parkingu.
- Tablica  $C$  zawierająca ceny za postój na poszczególnych parkingach.  $C[i]$  to opłata za zatrzymanie na  $i$ -ym parkingu.
- Maksymalna liczba kilometrów  $T$ , którą można przejechać bez zatrzymywania (z zastrzeżeniem wyjątku opisanego powyżej).
- Długość  $L$  trasy (liczba kilometrów od  $A$  do  $B$  wzdłuż trasy przejazdu).

Wszystkie wartości przekazane w tablicach  $O$  i  $C$  oraz argumenty  $T$  i  $L$  to dodatnie liczby naturalne. Tablice nie muszą być posortowane. Funkcja `min_cost` powinna zwrócić jedną liczbę naturalną: minimalny koszt przejazdu z  $A$  do  $B$ . Można założyć, że parkingi są tak rozmieszczone, że da się przejechać z  $A$  do  $B$  zgodnie z obowiązującymi zasadami. Funkcja powinna być możliwie jak najszybsza.

**Przykład.** Dla danych:

```
O = [17, 20, 11, 5, 12]
C = [9, 7, 7, 7, 3]
T = 7
L = 25
```

wywołanie `min_cost(O, C, T, L)` powinno zwrócić wartość 10.

**Podpowiedź.** Zastanów się, jaki jest koszt dojechania do parkingu  $i$  mogąc lub nie mogąc wykorzystać wyjątek, o którym mowa w punkcie 2.