Algorytmy i Struktury Danych
Zadanie offline 8 (10.VI.2024)

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. korzystanie z wbudowanych funkcji sortujących,
2. korzystanie z zaawansowanych struktur danych (np. słowników czy zbiorów),
3. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
4. modyfikowanie testów dostarczonych wraz z szablonem,
5. 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, heapq`),
2. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są).
3. 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: `python3 zad8.py`

Zadanie offline 8.

Szablon rozwiązania: `zad8.py`

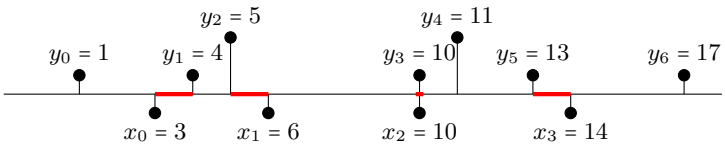
Szalony Inwestor wybudował po południowej stronie drogi n biurowców, na pozycjach $x_0 < \cdots < x_{n-1}$. Parkingi tych biurowców mają dopiero zostać wybudowane i dostępne jest w tym celu m działek ($m \geq n$), dostępnych na północnej stronie drogi, na pozycjach $y_0 < \cdots < y_{m-1}$. Inwestor chce wybudować dokładnie po jednym parkingu dla każdego biurowca (żadne dwa biurowce nie mogą dzielić tego samego parkingu). Zasady bezpiecznego ruchu wymagają, że i-ty biurowiec musi mieć parking na pozycji wcześniejszej niż $i+1$-szy. Inwestor chce wybudować parkingi na takich pozycjach, żeby suma odległości parkingów od biurowców była minimalna. Odległość i-go biurowca od j-ej działki to $|x_i - y_j|$. Zadanie polega na implementacji funkcji:

```
parking( X, Y )
```

która na wejściu otrzymuje listę `X` zawierającą n pozycji biurowców oraz listę `Y` zawierającą m pozycji działek na parkingi (listy `X` oraz `Y` zawierają nieujemne liczby całkowite). Funkcja powinna być możliwie jak najszybsza.

Przykład. Dla wejścia:

```
X = [3,6,10,14]
Y = [1,4,5,10,11,13,17]
```



wynikiem jest 3:

1. Biurowiec z pozycji $X[0] = 3$ dostaje parking na pozycji $Y[1] = 4$ (odlełgość 1),
2. Biurowiec z pozycji $X[1] = 6$ dostaje parking na pozycji $Y[2] = 5$ (odlełgość 1),
3. Biurowiec z pozycji $X[2] = 10$ dostaje parking na pozycji $Y[3] = 10$ (odlełgość 0),
4. Biurowiec z pozycji $X[3] = 14$ dostaje parking na pozycji $Y[5] = 13$ (odlełgość 1).

Podpowiedź. W realizacji algorytmu może pomóc obliczanie funkcji $f(i, j)$, zdefiniowanej jako:

> $f(i, j)$ = minimalna suma odległości biurowców z pozycji $X[0], \ldots, X[i]$ do przydzielonych im działek, przy założeniu że biurowiec z pozycji $X[i]$ ma przydzieloną działkę z pozycji $Y[j]$.