Algorytmy i Struktury Danych
Zadanie offline 5 (24.IV.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. 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
2. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są).

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 zad5.py`

Zadanie offline 5.

Szablon rozwiązania: `zad5.py`

Układ planetarny Algon składa się z n planet o numerach od 0 do $n-1$. Niestety własności fizyczne układu powodują, że nie da się łatwo przelecieć między dowolnymi dwiema planetami. Na szczęście mozolna eksploracja kosmosu doprowadziła do stworzenia listy E dopuszczalnych bezpośrednich przelotów. Każdy element listy E to trójka postaci (u, v, t), gdzie u i v to numery planet (można założyć, że $u < v$) a t to czas podróży między nimi (przelot z u do v trwa tyle samo co z v do u). Dodatkową nietypową własnością układu Algon jest to, że niektóre planety znajdują się w okolicy osobliwości. Znajdując się przy takiej planecie możliwe jest zagięcie czasoprzestrzeni umożliwiające przedostanie się do dowolnej innej planety leżącej przy osobliwości w czasie zerowym.

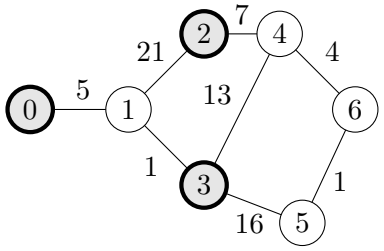
Zadanie polega na zaimplementowaniu funkcji:

```
def spacetravel( n, E, S, a, b )
```

która zwraca najkrótszy czas podróży z planety a do planety b, mając do dyspozycji listę możliwych bezpośrednich przelotów E oraz listę S planet znajdujących się koło osobliwości. Jeśli trasa nie istnieje, to funkcja powinna zwrócić `None`.

Rozważmy następujące dane:

```
E = [(0,1, 5),
     (1,2,21),
     (1,3, 1),
     (2,4, 7),
     (3,4,13),
     (3,5,16),
     (4,6, 4),
     (5,6, 1)]
S = [ 0, 2, 3 ]
a = 1
b = 5
n = 7
```



wywołanie `startravel(n, E, S, a, b)` powinno zwrócić liczbę 13. Odwiedzamy po kolei planety 1, 3, 2, 4, 6 i kończymy na planecie 5 (z planety 2 do 3 dostajemy się przez zagięcie czasoprzestrzeni). Gdyby $a = 1$ a $b = 2$ to wynikiem byłby czas przelotu 1.