

Algorytmy i Struktury Danych  
Kolokwium 2 (23 maja 2024r.)

## Format rozwiązań

Wysłać należy tylko jeden plik: `kol2.py`

Plik można wysłać wielokrotnie, liczy się ostatnia wersja zapisana w systemie.

**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 zaawansowanych struktur danych (np. słowników czy zbiorów),
2. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
3. 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`),
2. 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. `.ZIP`, `.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 kol1.py`

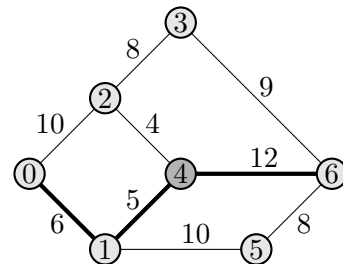
<b>Szablon rozwiązania:</b>	<code>kol1.py</code>
<b>Złożoność akceptowalna (1.0pkt):</b>	$O(V^3)$
<b>Złożoność lepsza (3.0pkt):</b>	$O(E \log V)$
<b>Złożoność wzorcowa (4.0pkt):</b>	$O(E + V)$
Gdzie $V$ to liczba wierzchołków w grafie a $E$ to liczba krawędzi.	

Magiczny wojownik chce się przedostać przez góry Bajtocji. Wyrusza z miasteczka  $s$  i chce się dostać do miasteczka  $t$ . Wojownik dysponuje mapą z zaznaczonymi schroniskami, miasteczkami  $s$  i  $t$ , oraz łączącymi je szlakami (mapa ma formę grafu gdzie miasteczka i schroniska to wierzchołki a szlaki to krawędzie). Każdy szlak ma przypisaną liczbę godzin potrzebnych, żeby go przebyć (są to liczby naturalne z zakresu od 1 do 16, zapisane jako wagi krawędzi grafu). Kodeks honorowy magicznych wojowników mówi, że wojownik nie może być w drodze bez odpoczynku dłużej niż 16 godzin. Taki odpoczynek musi trwać 8 godzin i musi się odbyć w schronisku. Wojownik chce się dostać z  $s$  do  $t$  jak najszybciej, ale nie może łamać zasad kodeksu. Gdy wojownik rusza z  $s$  jest w pełni wypoczęty, ale nie musi być wypoczęty gdy dotrze do  $t$ .

Proszę zaimplementować funkcję `warrior(G, s, t)`, która zwraca ile godzin trwa najszybsza droga wojownika z  $s$  do  $t$ , przy użyciu mapy opisanej jako graf  $G$ . Graf  $G$  reprezentowany jest jako lista krawędzi. Każda krawędź to trójka postaci  $(u, v, w)$ , gdzie  $u$  i  $v$  to numery wierzchołków a  $w$  to liczba godzin potrzebna na przebycie drogi z  $u$  do  $v$  (oraz z  $v$  do  $u$ ; graf jest nieskierowany). Numery wierzchołków to kolejne liczby naturalne od 0 do  $n - 1$ , gdzie  $n$  to liczba wierzchołków. Algorytm powinien być możliwie jak najszybszy. Proszę uzasadnić poprawność zaproponowanego algorytmu oraz oszacować jego złożoność czasową i pamięciową.

**Przykład 1.** Dla wejścia:

```
G = [ (1,5,10), (4,6,12), (3,2,8),
      (2,4,4), (2,0,10), (1,4,5),
      (1,0,6), (5,6,8), (6,3,9)]
s = 0
t = 6
```



wywołanie `warrior(G,s,t)` powinno zwrócić wartość 31, odpowiadającą trasie  $0 \rightarrow 1 \rightarrow 4(\text{nocleg}) \rightarrow 6$ , której przebycie trwa  $6 + 5 + 8 + 12 = 31$  godzin.