Algorytmy i Struktury Danych
Egzamin 1 (13.VII.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. modyfikowanie testów dostarczonych wraz z szablonem,
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, słownik, 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 egz1A.py`

Szablon rozwiązania:	`egz1A.py`
Złożoność akceptowalna (+2.0pkt):	$O(V^3 \log V)$
Złożoność wzorcowa (+2.0pkt):	$O(V^2 \log V)$
Gdzie V to liczba wierzchołków grafu.	

Złycerz (czyli zły rycerz) wędruje po średniowiecznym grafie $G = (V, E)$, gdzie waga każdej krawędzi to liczba sztabek złota, którą trzeba zapłacić za przejazd nią (myta, jedzenie, itp.). W każdym wierzchołku znajduje się zamek, który zawiera w skarbcu pewną daną liczbę sztabek złota. Złycerz może napaść na jeden zamek i zabrać całe jego złoto, ale od tego momentu zaczyna być ścigany i każdy przejazd po krawędzi jest dwa razy droższy, oraz dodatkowo na każdej drodze musi zapłacić r sztabek złota jako łapówkę (zatem od tej pory koszt przejazdu danej krawędzi jest równy dwukrotności wagi tej krawędzi plus wartość r). Co więcej, Złycerz nie może napaść więcej niż jednego zamku, bo jest trochę leniwy (oprócz tego, że zły). Proszę wskazać trasę Złycerza z zamku s do t o najmniejszym koszcie (lub największym zysku, jeśli to możliwe).

Uwaga. Złycerz może przejechać po danej krawędzi więcej niż raz (np. raz jadąc do zamku, który chce napaść, a potem z niego wracając).

Zadanie polega na implementacji funkcji:

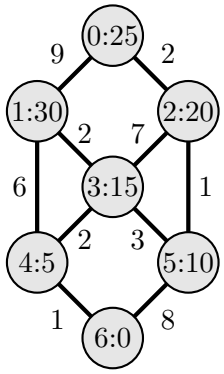
```
gold( G,V,s,t,r )
```

która na wejściu otrzymuje: graf G reprezentowany w postaci listowej, tablicę V zawierającą liczby sztabek złota w kolejnych zamkach, zamek początkowy s, zamek końcowy t oraz wysokość łapówki r. Funkcja powinna zwrócić najmniejszy koszt drogi uwzględniający ewentualny napad. Jeżeli zysk z napadu jest większy, od kosztu drogi należy, powstały zysk należy zwrócić jako liczbę ujemną (przykład).

Przykład. Dla wejścia:

```
G = [[(1,9), (2,2)],                 # 0
     [(0,9), (3,2), (4,6)],          # 1
     [(0,2), (3,7), (5,1)],          # 2
     [(1,2), (2,7), (4,2), (5,3)],   # 3
     [(1,6), (3,2), (6,1)],          # 4
     [(2,1), (3,3), (6,8)],          # 5
     [(4,1), (5,8)] ]                # 6

V = [25,30,20,15,5,10,0]
s = 0, t = 6, r = 7
```



wynikiem jest 6.

Gdyby nie rabować żadnego z zamków, najmniejszy koszt wyniósłby $2 + 1 + 3 + 2 + 1 = 9$, droga $[0, 2, 5, 3, 4, 6]$. Jednak jeżeli obrabujemy zamek 1 (30 sztabek złota), koszt wyniesie $2 + 1 + 3 + 2 - 30 + 19 + 9 = 6$, droga $[0, 2, 5, 3, 1, 4, 6]$. Gdyby łapówka r wynosiła 10, żadna kradzież nie byłaby opłacalna, jednak gdyby łapówka r wynosiła 3, wtedy (po obrabowaniu zamku 1), funkcja powinna zwrócić wartość -3.