Algorytmy i Struktury Danych
Kolokwium Zaliczeniowe II (1. IX 2020)

Format rozwiązań

Rozwiązanie każdego zadania musi składać się z opisu algorytmu (wraz z uzasadnieniem poprawności i oszacowaniem złożoności obliczeniowej) 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). 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, kolejka `collections.deque`,
2. korzystanie z wbudowanych algorytmów sortowania (**poza pierwszym zadaniem**),
3. korzystanie ze struktur danych dostarczonych razem z zadaniem.

Wszystkie inne algorytmy lub struktury danych (w tym słowniki) wymagają implementacji. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych w pliku z szablonem rozwiązania. Jeśli ktoś zaimplementuje standardowe drzewo BST, to może w analizie zakładać, że złożoność operacji na nim jest rzędu $O(\log n)$.

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

Proszę pamiętać, że rozwiązania trochę wolniejsze niż oczekiwane, ale poprawne, mają szanse na otrzymanie 1 punktu. Rozwiązania szybkie ale błędnie otrzymają 0 punktów. Proszę mierzyć siły na zamiary!

Testowanie rozwiązań

Żeby przetestować rozwiązania zadań należy wykonać:

```
python3 zad1.py

python3 zad2.py

python3 zad3.py
```

[2pkt.] Zadanie 1.

Szablon rozwiązania: `zad1.py`

Mówimy, że punkt (x, y) słabo dominuje punkt (x', y') jeśli $x \leq x'$ oraz $y \leq y'$ (w szczególności każdy punkt słabo dominuje samego siebie). Dana jest tablica P zawierająca n punktów. Proszę zaimplementować funkcję `dominance(P)`, która zwraca tablicę S taką, że:

1. elementami S są indeksy punktów z P,
2. dla każdego punktu z P, S zawiera indeks punktu, który go słabo dominuje,
3. S zawiera minimalną liczbę elementów.

Przykład. Dla tablicy:

```
P = [ (2,2), (1,1), (2.5,0.5), (3,2), (0.5,3) ]
#       0      1        2        3        4
```

wynikiem jest, między innymi:

```
S = [ 1, 4, 2 ]
```

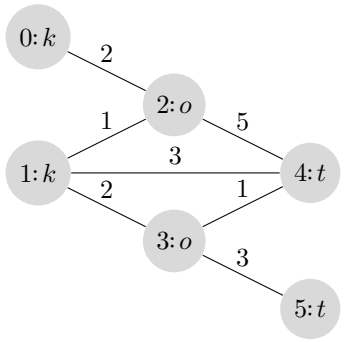
[2pkt.] Zadanie 2.

Szablon rozwiązania: `zad2.py`

Dany jest graf nieskierowany $G = (V, E)$, gdzie każdy wierzchołek z V ma przypisaną małą literę z alfabetu łacińskiego, a każda krawędź ma wagę (dodatnią liczbę całkowitą). Dane jest także słowo $W = W[0], \ldots, W[n-1]$ składające się małych liter alfabetu łacińskiego. Należy zaimplementować funkcję `letters(G,W)`, która oblicza długość najkrótszej ścieżki w grafie G, której wierzchołki układają się dokładnie w słowo W (ścieżka ta nie musi być prosta i może powtarzać wierzchołki). Jeśli takiej ścieżki nie ma, należy zwrócić -1.

Struktury danych. Graf G ma n wierzchołków ponumerowanych od 0 do $n-1$ i jest reprezentowany jako para (L, E). L to lista o długości n, gdzie $L[i]$ to litera przechowywana w wierzchołku i. E jest listą krawędzi i każdy jej element jest trójką postaci (u, v, w), gdzie u i v to wierzchołki połączone krawędzią o wadze w.

Przykład. Rozważmy graf G przedstawiony poniżej:



W reprezentacji przyjętej w zadaniu mógłby być zapisany jako:

```
    # 0   1   2   3   4   5
L = ["k","k","o","o","t","t"]
E = [(0,2,2), (1,2,1), (1,4,3), (1,3,2), (2,4,5), (3,4,1), (3,5,3) ]
G = (L,E)
```

Rozwiązaniem dla tego grafu i słowa `W = "kto"` jest 4 i jest osiągane przez ścieżkę $1 - 4 - 3$. Inna ścieżka realizująca to słowo to $1 - 4 - 2$, ale ma koszt 8.

[2pkt.] Zadanie 3.

Szablon rozwiązania: `zad3.py`

Dana jest tablica T zawierająca N liczb naturalnych. Z pozycji a można przeskoczyć na pozycję b jeżeli liczby $T[a]$ i $T[b]$ mają co najmniej jedną wspólną cyfrę. Koszt takego skoku równy $|T[a] - T[b]|$. Proszę napisać funkcję, która wyznacza minimalny sumaryczny koszt przejścia z najmniejszej do największej liczby w tablicy T. Jeżeli takie przejście jest niemożliwe, funkcja powinna zwrócić wartość -1.

Przykład Dla tablicy `T = [123,890,688,587,257,246]` wynikiem jest liczba 767, a dla tablicy `T = [587,990,257,246,668,132]` wynikiem jest liczba -1.