Algorytmy i Struktury Danych
Egzamin Poprawkowy (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. Nie dopuszczalne jest w szczególności:

1. zmiana nazwy funkcji implementującej algorytm lub listy jej argumentów,
2. modyfikacja 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,
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 szybsze, ale błędne 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`

Żab Zbigniew skacze po osi liczbowej. Ma się dostać z zera do $n-1$, skacząc wyłącznie w kierunku większych liczb. Skok z liczby $i$ do liczby $j$ ($j > i$) kosztuje Zbigniewa $j-i$ jednostek energii, a jego energia nigdy nie może spaść poniżej zera. Na początku Zbigniew ma 0 jednostek energii, ale na szczęście na niektórych liczbach—także na zerze—leżą przekąski o określonej wartości energetycznej (wartość przekąki dodaje się do aktualnej energii Zbigniewa).

Proszę zaimplementować funkcję `zbigniew(A)`, która otrzymuje na wejściu tablicę `A` długości `len(A) = n`, gdzie każde pole zawiera wartość energetyczną przekąski leżącej na odpowiedniej liczbie. Funkcja powinna zwrócić minimalną liczbę skoków potrzebną, żeby Zbigniew dotarł z zera do `n-1` lub $-1$ jeśli nie jest to możliwe.

**Podpowiedź.** Warto rozważyć funkcję $f(i, y)$ zwracającą minimalną liczbę skoków potrzebną by dotrzeć do liczby $i$ mając w zapasie dokładnie $y$ jednostek energii.

**Przykład.** Dla tablicy `A = [2,2,1,0,0,0]` wynikiem jest 3 (Zbigniew skacze z 0 na 1, z 1 na 2 i z 2 na 5, kończąc z zerową energią). Dla tablicy `A = [4,5,2,4,1,2,1,0]` wynikiem jest 2 (Zbigniew skacze z 0 na 3 i z 3 na 7, kończąc z jedną jednostką energii).

## [2pkt.] Zadanie 2.

**Szablon rozwiązania:** `zad2.py`

W pewnym państwie, w którym znajduje się $N$ miast, postanowiono połączyć wszystkie miasta siecią autostrad, tak aby możliwe było dotarcie autostradą do każdego miasta. Ponieważ kontynent, na którym leży państwo jest płaski położenie każdego z miast opisują dwie liczby $x, y$, a odległość w linii prostej pomiędzy miastami liczona w kilometrach wyraża się wzorem $len = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Z uwagi na oszczędności materiałów autostrada łączy dwa miasta w linii prostej.

Ponieważ zbliżają się wybory prezydenta, wszystkie autostrady zaczęto budować równocześnie i jako cel postanowiono zminimalizować czas pomiędzy otwarciem pierwszej i ostatniej autostrady. Czas budowy autostrady wyrażony w dniach wynosi $\lceil len \rceil$ (sufit z długości autostrady wyrażonej w km).

Proszę zaimplementować funkcję `highway(A)`, która dla danych położeń miast wyznacza minimalną liczbę dni dzielącą otwarcie pierwszej i ostatniej autostrady.

**Przykład** Dla tablicy `A =[(10,10),(15,25),(20,20),(30,40)]` wynikiem jest 7 (Autostrady pomiędzy miastami 0-1, 0-2, 1-3).

## [2pkt.] Zadanie 3.

**Szablon rozwiązania:** `zad3.py`

Dany jest zbiór $N$ zadań, gdzie niektóre zadania muszą być wykonane przed innymi zadaniami. Wzajemne kolejności zadań opisuje dwuwymiarowa tablica $T[N][N]$. Jeżeli $T[a][b] = 1$ to wykonanie zadania $a$ musi poprzedzać wykonanie zadania $b$. W przypadku gdy $T[a][b] = 2$ zadanie $b$ musi być wykonane wcześniej, a gdy $T[a][b] = 0$ kolejność zadań $a$ i $b$ jest obojętna. Proszę zaimplementować funkcję `tasks(T)`, która dla danej tablicy $T$, zwraca tablicę z kolejnymi numerami zadań do wykonania.

**Przykład** Dla tablicy `T = [ [0,2,1,1], [1,0,1,1], [2,2,0,1], [2,2,2,0] ]` wynikiem jest tablica `[1,0,2,3]`.