Bit Algo
START

BFS i DFS

Uwaga o reprezentacji grafu

Jeżeli zadanie nie podaje explicite reprezentacji grafu, to zwykle jego pierwszą częścią jest wybranie najkorzystniejszej reprezentacji.

Wybór np. na kolokwium trzeba **uzasadnić**.

BFS - Breadth First Search

```
from queue import Queue

### Implementacja BFS dla grafu w postaci list sąsiedztwa

def bfs(graph, s):
    queue = Queue()
    visited = [False]*len(graph)

    queue.put(s)
    visited[s] = True

    while not queue.empty():
        u = queue.get()
        for v in graph[u]:
            if not visited[v]:
                visited[v] = True
                queue.put(v)
```

DFS - Depth First Search

```
### Implementacja DFS dla grafu w postaci list sąsiedztwa

def dfs(graph, s):
    visited = [False]*len(graph)

    def dfs_visit(u):
        nonlocal graph, visited

        visited[u] = True
        for v in graph[u]:
            if not visited[v]:
                dfs_visit(v)

    dfs_visit(s)
```

Zadanie 1: Detekcja cyklu

Napisz algorytm sprawdzający, czy graf nieskierowany posiada cykl.

Zadanie 1 - rozwiązanie

Rozwiązanie: zauważmy, że jeżeli wejdziemy w cykl w grafie nieskierowanym, to przeglądając poddrzewo pierwszego wierzchołka cyklu do którego weszliśmy, będziemy ponownie próbowali się do niego dostać, przed wyjściem z niego.

Łopatologicznie: jak już coś pokolorowaliśmy, to tam byliśmy, więc jak tam chcemy wejść, to znaczy, że mamy cykl.

```
def DFSvisit(Graph, vertex):
    vertex.color = Gray
    isCycle = false
    for v in vertex.adj:
        if Graph[v].color == WHITE:
            isCycle = isCycle or DFSvisit(Graph, Graph[v])
        elif Graph[v].color ==      GRAY:
                isCycle = True
        vertex.color = BLACK
        return isCycle
```

Zadanie 2: Wiadomość

Otrzymujemy na wejściu listę par ludzi, które się wzajemnie znają. Osoby są reprezentowane przez liczby od 0 do n - 1.

Dnia pierwszego osoba 0 przekazuje pewną wiadomość wszystkim swoim znajomym. Dnia drugiego każdy ze znajomych przekazuje tę wiadomość wszystkim swoim znajomym, którzy jej jeszcze nie znali, i tak dalej.

Napisz algorytm, który zwróci dzień, w którym najwięcej osób poznało wiadomość oraz ilość osób, które tego dnia ją otrzymały.

Zadanie 3: Jeziora

Dana jest dwuwymiarowa tablica N x N, w której każda komórka ma wartość “W” - reprezentującą wodę lub “L” - ląd. Grupę komórek wody połączonych ze sobą brzegami nazywamy jeziorem.

a) Policz, ile jezior jest w tablicy

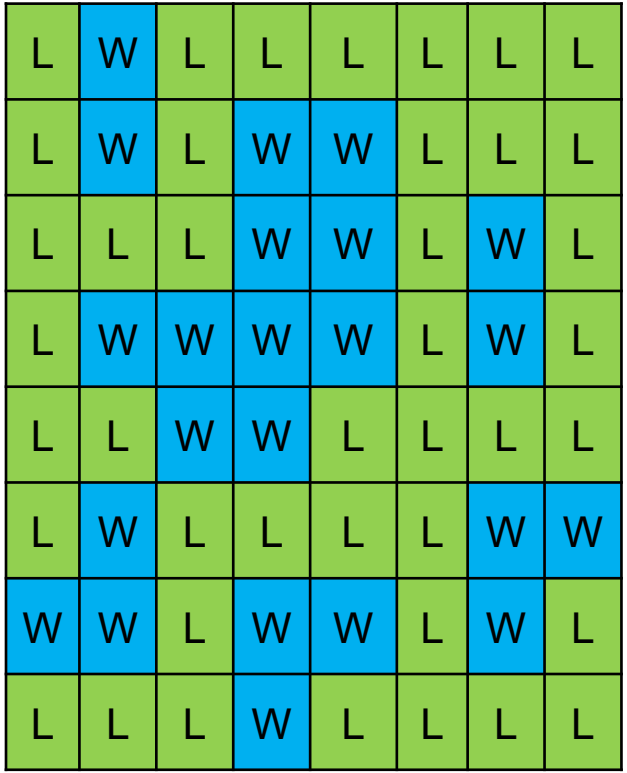
b) Policz, ile komórek zawiera największe jezioro

L	W	L	L	L	L	L	L
L	W	L	W	W	L	L	L
L	L	L	W	W	L	W	L
L	W	W	W	W	L	W	L
L	L	W	W	L	L	L	L
L	W	L	L	L	L	W	W
W	W	L	W	W	L	W	L
L	L	L	W	L	L	L	L

Zadanie 3: Jeziora - cd.

c) Zakładając, że pola o indeksach [0][0] i [n-1][n-1] są lądem, sprawdź czy da się przejść drogą lądową z pola [0][0] do pola [n-1][n-1]. Można chodzić tylko na boki, nie na ukos.

d) Znajdź najkrótszą ścieżkę między tymi punktami. Wypisz po kolei indeksy pól w tej ścieżce



Zadanie 4: Sklejanie przedziałów

Dany jest ciąg przedziałów postaci $[a_i, b_i]$. Dwa przedziały można skleić, jeśli mają dokładnie jeden punkt wspólny. Podaj algorytm, który sprawdza, czy da się uzyskać przedział [a, b] poprzez sklejanie odcinków.

Zadanie 4 - stworzenie grafu

Na dobry początek trzeba w ogóle stworzyć graf z tych odcinków. Każdy wierzchołek to początek lub koniec odcinka, krawędzie są między tymi wierzchołkami, które tworzą odcinek (początek -> koniec).

Optymalna reprezentacja: zależy od grafu, listowo będzie pewnie prościej

Rozwiązanie: idziemy po odcinkach, z końca i początku tworzymy wierzchołek, jeżeli jeszcze go nie ma (trzymamy je np. w słowniku); po ew. utworzeniu dodajemy krawędź.

Zadanie 4 - uzyskanie przedziału [a, b]

Rozwiązanie: jeżeli w słowniku **nie** ma klucza a, to zwracamy False. Jeżeli klucz a jest, to puszczamy DFSa po grafie, aż dojdziemy do b i wtedy zwracamy True. Jeżeli po przeszukaniu całego grafu nie doszliśmy, to zwracamy False.

Złożoność: O(Vlog(V)) + O(V + E) = O(Vlog(V) + E), konstrukcja grafu + algorytm

Zadanie 5: Sejf

Dostałeś sejf, który odblokowuje się czterocyfrowym PINem (0000 - 9999). Pod wyświetlaczem znajduje się kilka przycisków z liczbami od 1 do 9999 - przykładowo (13, 223, 782, 3902). Sejf ten działa inaczej niż normalny: wciśnięcie przycisku z liczbą powoduje dodanie liczby z przycisku do liczby na wyświetlaczu. Jeżeli suma jest większa niż 9999, to pierwsza cyfra zostaje obcięta.

Jest tobie znany PIN oraz cyfry, które są aktualnie wyświetlane. Znajdź najkrótszą sekwencję naciśnięć przycisków, która pozwoli ci odblokować sejf. Jeżeli taka sekwencja nie istnieje, zwróć None.

Zadanie 6: Rozmiary poddrzew

Dostajemy na wejściu listę krawędzi drzewa (niekoniecznie binarnego!) oraz wyróżniony wierzchołek - korzeń. Każdy wierzchołek tworzy swoje własne poddrzewo. Dla każdego wierzchołka, wyznacz ilość wierzchołków w jego poddrzewie.

Zadanie 7: Domy i sklepy

Mamy mapę miasteczka, w którym są domy i sklepy. Na mapie są również drogi (każda długości 1), które łączą dom z domem, albo dom ze sklepem. Naszym zadaniem jest, dla każdego domu, znaleźć odległość do najbliższego sklepu.

Kliknij, aby dodać tekst

Zadanie 8: Średnica drzewa

Średnicą drzewa nazywamy odległość między jego najbardziej oddalonymi od siebie wierzchołkami. Napisz algorytm, który przyjmując na wejściu drzewo (niekoniecznie binarne!) w postaci listy krawędzi zwróci jego średnicę.

Bit Algo
START