Warunki

1. Zadania można wysyłać, jeśli ktoś ma najwyżej 2 plusy (licząc od 18 III).
2. W implementacjach można korzystać tylko z elementarnych konstrukcji Python'a (funkcje, instrukcje warunkowe, pętle, `range`, klasy użyte do definiowania struktur danych, wbudowana funkcja sortująca, itp.). **Nie wolno korzystać ze słowników i zbiorów.**
3. Rozwiązania muszą być efektywne obliczeniowo (także w zadaniach, w których nie podajemy wprost ograniczenia na złożoność obliczeniową). Zadania o zbyt wysokiej złożoności będą oceniane na brak plusa.

Zadanie 1 (BFS)

Proszę zaimplementować algorytm BFS dla macierzowej reprezentacji grafu skierowanego. Państwa implementacja powinna zwracać tablicę postaci

$$[(parent_0, d_0), (parent_1, d_1), \ldots, (parent_{n-1}, d_{n-1})],$$

gdzie $parent_i$ to poprzednik wierzchołka i na najkrótszej ścieżce z wierzchołka źródłowego a d_i to odległość i-go wierzchołka od źródłowe. Państwa kod powinien mieć następującą postać (będzie uruchamiany; proszę nie usuwać fragmentu testującego; sprawdzający może także dołożyć swoje testy):

```
def BFS( G, s ):
# G to macierz opisująca graf: G[i][j]==1 jeśli jest
# wierzchołek z i do j. W przeciwnym razie G[i][j]=0
# s to numer wierzchołka źródłowego

# tu proszę umieścić swoją implementację


# elementarny test, powinien wypisać
# [(None,0), (0,1), (0,1), (2,2)]
# lub
# [(None,0), (0,1), (0,1), (1,2)]
G = [[0,1,1,0],[0,0,0,1],[0,1,0,1], [0,0,0,0]]
print( BFS(G,0) )
```

Zadanie 2 (DFS)

Proszę zaimplementować algorytm DFS dla grafu nieskierowanego reprezentowanego przez listy sąsiedztwa. Państwa implementacja powinna zwracać tablicę

postaci $[parent_0, parent_1, \ldots, parent_{n-1}]$, gdzie $parent_i$ to poprzednik wierzchołka i w drzewie DFS. Państwa kod powinien mieć następującą postać (będzie uruchamiany; proszę nie usuwać fragmentu testującego; sprawdzający może także dołożyć swoje testy):

```
def DFS( G ):
  # G to lista list z informacją o istnieniu krawędzi
  # G[i] to lista numerów wierzchołków, które są połączone
  #      krawędzią z wierzchołkiem i

  # tu proszę umieścić swoją implementację


  # elementarny test. Może wypisać np.
  # [None, 0, 1, 2]
  G = [[1,2],[0,2,3],[3,1,0],[]]
  print( DFS(G) )
```