МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

Отчет по лабораторной работе №3
Вариант 12

Бовта Тимофея Анатольевича
студента 3 курса
специальности «прикладная математика»

Преподаватель:

Д. Ю. Кваша

Минск, 2024 г.

Лабораторная работа №3

Задача 1.

N шестеренок пронумерованы от 0 до N ($N \leqslant 10$). Задана матрица соединений пар шестеренок в виде $C[i, j] = 1$, если шестерня с номером i находится в зацеплении с шестерней j, где $1 \leqslant i < j \leqslant N$. Можно ли повернуть шестерню с заданным номером? Если да, то найти количество шестерен, пришедших в движение.

Программная реализация алгоритма.

```
from collections import deque

def count_rotatable_gears(adj_matrix, start_gear):
    N = len(adj_matrix)
    directions = [-1] * N
    directions[start_gear] = 0
    queue = deque([start_gear])
    replaced = False

    while queue and not replaced:
        current_gear = queue.popleft()
        incident_gears = []

        for j in range(current_gear + 1, N):
            if adj_matrix[current_gear][j]:
                incident_gears.append(j)

        for gear in incident_gears:
            if directions[gear] == -1:
                directions[gear] = 1 - directions[current_gear]
                queue.append(gear)
            elif directions[gear] == directions[current_gear]:
                replaced = True
                break

    if replaced:
        return 0  # Система шестерёнок не будет вращаться
    else:
        count = directions.count(1)
        return count  # Количество вращающихся шестерёнок

# Пример использования
adjacency_matrix = [
    [0, 1, 0, 1, 1],
    [0, 0, 1, 1, 0],
    [0, 0, 0, 0, 1],
    [0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0]
]
start_gear = 0
```

Примеры

входной файл	выходной файл
5 2 5 3 1 2 4 3 5 4 1	YES
3 2 1 0	NO 2
5 2 3 1 3 1 2 5 4	NO 2

Рис. 1:

```
41
42  result = count_rotatable_gears(adjacency_matrix, start_gear)
43  print(f"Количество вращающихся шестерёнок: {result}")
```

Задача 2.

Имеется страна с N городами. Между некоторыми парами городов построены магистрали. Длина любого прямого соединения равна единице. Движение по магистрали возможно в обе стороны. Систему магистралей будем называть чётно-нечётной, если каждая пара различных городов соединена маршрутом как чётной длины, так и нечётной (причём маршрут может проходить через один город несколько раз). Необходимо определить, является ли система магистралей чётно-нечётной. Если ответ на вопрос отрицательный, то нужно найти одно из подмножеств множества городов, в котором наибольшее число элементов и которое удовлетворяет следующему условию: если какие-либо два различных города из этого подмножества соединены маршрутом, то его длина чётна.

Формат входных данных
Первая строка содержит число N городов ($1 \leqslant N \leqslant 300$). Следующие N строк файла задают магистрали: $(i + 1)$-я строка содержит номера городов, которые связаны напрямую 7 с городом i (если таких городов нет, то строка содержит нуль; числа в строке разделены пробелами).

Формат выходных данных
Если система магистралей является чётно-нечётной, то выведите единственную строку YES. В противном случае в первой строке выведите NO, а во второй — мощность описанного наибольшего подмножества.

Программная реализация алгоритма.

```
1  from collections import deque
2
3  n = 0
4  count = 0
5  graph = []
6  met = []
7  two = False
8  connect = False
9  ma = 0
```

```
10
11  def bfs(v):
12      global count, two, met, ma
13      dols = [1, 0]
14      count = 1
15      two = True
16      met[v] = 0
17      queue = deque([v])
18
19      while queue:
20          tmp = queue.popleft()
21          for i in range(len(graph[tmp])):
22              if met[graph[tmp][i]] == -1:
23                  count += 1
24                  met[graph[tmp][i]] = (met[tmp] + 1) % 2
25                  dols[(met[tmp] + 1) % 2] += 1
26                  queue.append(graph[tmp][i])
27              elif met[graph[tmp][i]] == met[tmp]:
28                  two = False
29
30      if two:
31          if dols[0] > dols[1]:
32              ma += dols[0]
33          else:
34              ma += dols[1]
35      else:
36          ma += 1
37
38  def main():
39      global n, count, graph, met, two, ma
40      with open("input.txt", "r") as fin:
41          ma = 0
42          j = 0
43          n = int(fin.readline())
44          graph = [[] for _ in range(n)]
45          met = [-1] * n
46
47          for line in fin:
48              line = line.strip()
49              values = list(map(int, line.split()))
50              for v in values:
51                  if v != 0:
52                      graph[j].append(v - 1)
53              j += 1
54
55      count = bfs(0)
56      times = 1
57
58      with open("output.txt", "w") as fout:
59          if count == n and not two:
```

```
60              fout.write("YES")
61          elif count == n and two:
62              fout.write("NO\n")
63              fout.write(str(ma))
64          elif count != n:
65              for i in range(n):
66                  if met[i] == -1:
67                      count += bfs(i)
68                      times += 1
69              fout.write("NO\n")
70              fout.write(str(ma))
```