

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных
технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа № 5

По дисциплине «Введение в цифровую культуру и программирование»

Работа с графом

Выполнил студент группы М3101

Дудко Матвей Владимирович

Проверил:

Хлопотов Максим Валерьевич

САНКТ-ПЕТЕРБУРГ

2019

ЛАБОРАТОРНАЯ РАБОТА 5. РАБОТА С ГРАФОМ. ВАРИАНТ 76

Описание работы

В вашу тему на Piazza загружены два текстовых файла.

graphedgesN.txt

graphtaskN.txt

N – это ваш уникальный номер варианта.

В файле graphedgesN.txt перечислены пары в таком виде:

0 259

...

Эти пары являются рёбрами графа. Например, 0 259 означает, что в графе есть ребро между вершинами 0 и 259. Всего граф содержит 1000 вершин, с номерами от 0 до 999.

Вопрос 1. Сколько в нём рёбер?

Изолятом называется вершина, не связанная ни с одной другой вершиной.

Степенью вершины называется количество ребёр, которые связывают её с другими вершинами.

Вопрос 2. Сколько в графе изолятов? Выведите полный список, упорядоченный по возрастанию

Вопрос 3. Найдите вершину(вершины) с самой большой степенью.

Компонента связности - это максимальный связный подграф. Диаметр - это самый длинный кратчайший путь

На рисунке ниже красным цветом выделена компонента связности. Её диаметр равен 4. Изолят выделен синим цветом.

Вопрос 4. Найдите диаметр компоненты связности графа

Кратчайший путь - минимальная сумма ребёр, составляющих путь от одной вершины к другой.

Вопрос 5. Найдите кратчайший путь от A до B.

Вопрос 6. Найдите кратчайший путь от С до D.

Вопрос 7. Найдите кратчайший путь от E до F.

Значения вершин A-F в каждом варианте свои. Они есть в файле `graphtaskN.txt`

Обратите внимание, что ответ должен включать в себя длину пути и последовательность вершин.

Если путь отсутствует, то нужно сделать соответствующую пометку.

Дальше требуются удалить из графа несколько вершин. Эти вершины в каждом варианте свои. Они есть в файле `graphtaskN.txt`

ВАЖНО!

В файле `graphtaskN.txt` пропущено одно условие!

В каждом варианте из графа удаляются также вершины, в которых номер кратен 17, включая вершину 0. То есть кроме тех, вершин, которые перечислены в задании, удаляются вершины 0, 17, 34, 51 и т.д.

После удаления этих вершин, нужно снова ответить на вопросы, на которые уже отвечали ранее

Вопрос 8. Сколько ребёр в графе?

Вопрос 9. Сколько в графе изолятов? Выведите полный список, упорядоченный по возрастанию

Вопрос 10. Найти вершину(вершины) с самой большой степенью

Вопрос 11. Найдите диаметр компоненты связности графа

Вопрос 12. Найдите кратчайший путь от A до B

Вопрос 13. Найдите кратчайший путь от C до D

Вопрос 14. Найдите кратчайший путь от E до F

В ответе на сообщение, в котором текстовые файлы с заданием дайте ответы на вопросы 1-14. Также приложите отчёт в формате pdf. В отчёте должен быть код, описание хода работы и ответы на вопросы 1-14.

Код, при помощи которого получены ответы на вопросы

```
Matrix = [] # Матрица смежности
List_connect = [] # Список связности

for i in range(1000):
    Matrix.append([0] * 1000)
    List_connect.append([])

# Будем считать, что бесконечность будет равна 1000000

# ONE: Чтение файла и первичный расчет
# -----

count = 0

with open("graphedges76.txt", "r") as f:
    for i in f.readlines():
        count += 1
        x, y = list(map(int, i.split()))
        Matrix[x][y] = 1
        Matrix[y][x] = 1
        List_connect[x].append(y)
        List_connect[y].append(x)

print("Количество ребер:", count)

# TWO: Количество изолятов
# -----

count = 0
string_list = ""
for i in range(1000):
    if sum(Matrix[i]) == 0:
        count += 1
        string_list += str(i) + " "

print("Количество изолятов:", count)
print("Список изолятов упорядоченный по возрастанию:", string_list)

assert len(list(filter(lambda x: len(x) == 0, List_connect))) == count

# THREE: Вершина с самой большой степенью
# -----

a = [sum(Matrix[i]) for i in range(1000)] # Количество ребер у каждой вершины
max_pow = max(a)
print("Наибольшая степень вершины:", max_pow)
print("Вершины с наибольшей степенью: ", end='')
for i in range(1000):
    if a[i] == max_pow:
        print(i, end=' ')
print()
```

```
# FOUR: Диаметр компоненты связности графа
# -----

# Компоненты связности
components = []
is_visited = [False] * 1000
viewed = set()

def dfs(vertex):
    is_visited[vertex] = True
    for i in List_connect[vertex]:
        if not is_visited[i]:
            dfs(i)

# Находим все компоненты связности
for i in range(1000):
    if not is_visited[i]:
        dfs(i)

    current_component = []

    for j in range(1000):
        if j not in viewed and is_visited[j]:
            current_component.append(j)
            viewed.add(j)

    if len(current_component) != 0:
        components.append(current_component)

# N - количество вершин, S - номер стартовой вершины (от нуля), matrix - матрица смежности
def Dijkstra(N, S, matrix):
    valid = [True]*N
    weight = [1000000]*N
    weight[S] = 0
    for i in range(N):
        min_weight = 1000001
        ID_min_weight = -1
        for i in range(N):
            if valid[i] and weight[i] < min_weight:
                min_weight = weight[i]
                ID_min_weight = i
        for i in range(N):
            if matrix[ID_min_weight][i] != 0 and weight[ID_min_weight] +
matrix[ID_min_weight][i] < weight[i]:
                weight[i] = weight[ID_min_weight] + matrix[ID_min_weight][i]
            valid[ID_min_weight] = False
    return weight

max_weight = 0

for component in components:
    for j in component:
        weight = Dijkstra(1000, j, Matrix)
        for i in weight:
            if max_weight < i != 1000000:
                max_weight = i

print("Диаметр компоненты связности графа:", max_weight)
```

```
# FIVE: пути до вершин
# -----

"""
A: 152 B: 403
C: 968 D: 334
E: 140 F: 888
"""

# Note: Все вершины находятся в первой (самой большой) компоненте связности

def find_path(names, first, second):
    weight = Dijkstra(1000, first, Matrix)

    if weight[second] == 1000000:
        print("Пути от {:}={:} до {:}={:} не существует".format(names[0], first, names[1],
second))
    else:
        path = []
        current_vertex = second
        while weight[current_vertex] != 0:
            path.append(current_vertex)
            current_search_weight = weight[current_vertex] - 1
            search_prev_vertex = 0
            while (weight[search_prev_vertex] != current_search_weight or
Matrix[search_prev_vertex][current_vertex] == 0) and search_prev_vertex < 1000:
                search_prev_vertex += 1
            assert search_prev_vertex < 1000
            current_vertex = search_prev_vertex

        print("Путь от {:}={:} до {:}={:}: длина: {:}, путь: ".format(names[0], first,
names[1], second, weight[second]), end='')
        for i in ([first] + path[::-1])[:-1]:
            print(i, end="->")
        print(second)

find_path(["A", "B"], 152, 403)
find_path(["C", "D"], 968, 334)
find_path(["E", "F"], 140, 888)

# SIX: Удаление вершин
# -----

# Удаление заданных вершин
to_remove = [647, 489, 395, 248, 410, 190, 767]

for i in to_remove:
    for j in range(1000):
        Matrix[j][i] = 0
        Matrix[i][j] = 0
        if i in List_connect[j]:
            List_connect[j].remove(i)
    List_connect[i] = []
```

```
# Удаление вершин, кратных 17
for i in range(0, 1000, 17):
    for j in range(1000):
        Matrix[j][i] = 0
        Matrix[i][j] = 0
        if i in List_connect[j]:
            List_connect[j].remove(i)
    List_connect[i] = []

# SEVEN: Перерасчет
# -----

print("-" * 100)
print("Перерасчет после удаления вершин:\n")

# Количество ребер
count = 0
for i in range(1000):
    for j in range(i, 1000):
        count += Matrix[i][j]

print("Количество ребер:", count)

count = 0
string_list = ""
for i in range(1000):
    if sum(Matrix[i]) == 0 and i not in to_remove and i % 17 != 0:
        count += 1
        string_list += str(i) + " "

print("Количество изолятов:", count)
print("Список изолятов упорядоченный по возрастанию:", string_list)

a = [sum(Matrix[i]) for i in range(1000)] # Количество ребер у каждой вершины
max_pow = max(a)
print("Наибольшая степень вершины:", max_pow)
print("Вершины с наибольшей степенью: ", end='')
for i in range(1000):
    if a[i] == max_pow:
        print(i, end=' ')
print()

# Компоненты связности
components = []
is_visited = [False] * 1000
viewed = set()
```

```
# Находим все компоненты связности
for i in range(1000):
    if not is_visited[i]:
        dfs(i)

    current_component = []

    for j in range(1000):
        if j not in viewed and is_visited[j]:
            current_component.append(j)
            viewed.add(j)

    if len(current_component) != 0:
        components.append(current_component)

max_weight = 0

for component in components:
    for j in component:
        weight = Dijkstra(1000, j, Matrix)
        for i in weight:
            if max_weight < i != 1000000:
                max_weight = i

print("Диаметр компоненты связности графа:", max_weight)

find_path(["A", "B"], 152, 403)
find_path(["C", "D"], 968, 334)
find_path(["E", "F"], 140, 888)
```

Описание хода работы

1. Считываем данные из файла, обнуляем переменные для хранения информации о ребрах в графе. Храним данные в виде матрицы смежности и списка смежности
2. Подсчитываем количество ребер: количество строк в исходном файле
3. Подсчитываем количество изолятов: считаем количество нулевых строк в матрице смежности
4. Ищем вершину с самой большой степенью: суммируем количество единиц построчно в матрице смежности, выбираем максимум, смотрим по каждой вершине и выводим с максимальной степенью
5. Ищем диаметр компоненты связности:
 - a. Ищем компоненты связности: при помощи алгоритма DFS (поиска в глубину) определяем компоненты связности
 - b. По каждой компоненте (по “максимальной”) вычисляем расстояние от каждой вершины до каждой, входящих в данную компоненту, и берем максимальное кратчайшее расстояние – это и будет диаметром компоненты связности
6. Ищем пути до вершин:
 - a. Воспользуемся алгоритмом Дейкстры, для нахождения длины кратчайшего пути до вершины
 - b. Восстановление пути: от конечной вершины, если длина до нее не бесконечность, идем в обратном порядке, начиная от конечной вершины, выбирая каждый раз ту вершину, у которой длина от исходной на 1 меньше
 - c. Выводим пути в обратном порядке
7. Производим удаление вершин, заданных в тексте задания, а также кратных 17
8. Повторяем расчеты после удаления (шаги 2 - 6). Количество ребер – количество ребер, выше главной диагонали матрицы смежности

Ответы на вопросы

Вопрос 1: Количество ребер: 2498

Вопрос 2: Количество изолятов: 7

Список изолятов упорядоченный по возрастанию: 105 297 341 357 390 424 748

Вопрос 3: Наибольшая степень вершины: 13

Вершины с наибольшей степенью: 647 767

Вопрос 4: Диаметр компоненты связности графа: 9

Вопрос 5: Путь от A=152 до B=403: длина: 5, путь: 152->410->924->395->600->403

Вопрос 6: Путь от C=968 до D=334: длина: 5, путь: 968->629->669->50->190->334

Вопрос 7: Путь от E=140 до F=888: длина: 4, путь: 140->767->489->686->888

Вопрос 8: Количество ребер: 2156

Вопрос 9: Количество изолятов: 6

Список изолятов упорядоченный по возрастанию: 105 297 341 390 424 541

Вопрос 10: Наибольшая степень вершины: 12

Вершины с наибольшей степенью: 185 698

Вопрос 11: Диаметр компоненты связности графа: 9

Вопрос 12: Путь от A=152 до B=403: длина: 5, путь: 152->753->662->185->942->403

Вопрос 13: Путь от C=968 до D=334: длина: 5, путь: 968->466->929->858->478->334

Вопрос 14: Путь от E=140 до F=888: длина: 6, путь: 140->351->137->771->478->334->888