Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

ОТЧЕТ
по лабораторной работе №3 Графы по дисциплине «Алгоритмы и структуры данных»

Выполнил студент 1 курса, группы K3141

Рахлин Роман Михайлович

Преподаватель: Харьковская Татьяна Александровна

28 Марта, 2022 года, г. Санкт-Петербург

Задание 1

Описание задачи: Нам нужно проверить есть ли путь от одного нода графа до другого.

Исходный код и описание:

```
with open("input.txt", "r") as input_file:
    n, m = map(int, input_file.readline().split())
    graph = []

    for _ in range(n):
        graph.append([])

    for _ in range(m):
        u, v = map(int, input_file.readline().split())
        graph[u - 1].append(v - 1)
        graph[v - 1].append(u - 1)

    visited = [0] * n
    start, final = map(int, input_file.readline().split())
    start -= 1
    final -= 1

def dfs(node):
    visited[node] = 1
    if node == final:
        return
    for neighbour in graph[node]:
        if not visited[neighbour]:
            dfs(neighbour)

dfs(start)

with open("output.txt", "w") as output_file:
    if visited[final]:
        output_file.write("1")
    else:
        output_file.write("0")
```

Первым делом строим граф как матрицу путей. И наша задача с помощью DFS обойти всех соседей и понять есть ли путь от start в final.

Задание 9

Описание задачи: В этой задаче нам нужно определить содержится в ли в графе цикл с негативной суммой путей.

Исходный код и описание:

```
def is_negative_cycle_bellman_ford(src, dist):
    for i in range(V):
        dist[i] = 10**18

    dist[src] = 0

    for i in range(1, V):
        for j in range(E):
            u = graph[j][0]
            v = graph[j][1]
            weight = graph[j][2]
            if (dist[u] != 10**18 and dist[u] + weight < dist[v]):
                dist[v] = dist[u] + weight

    for i in range(E):
        u = graph[i][0]
        v = graph[i][1]
        weight = graph[i][2]
        if (dist[u] != 10**18 and dist[u] + weight < dist[v]):
            return True

    return False
```

Используем алгоритм Бэллмана-Форда для того чтобы быстро по времени (по памяти также как и другие алгоритмы) пройти граф и составить матрицу, проанализировав которую мы придем к ответу.

Задание 13

Описание задачи: Тут нам нужно определить коплчество стоящих друг с другом символов в матрице.

Исходный код и описание:

Тут все максимально просто. Нам надо представить данные как матрицу. И потом просто пройтись по ней DFS и посчитать количество грядок. А после вывести это значение в результат.

```python
def dfs(matrix, m, n):
    result = 0

    for i in range(n):
        for j in range(m):
            if matrix[i][j] == "#":
                result += 1
                matrix = _dfs(i, j, matrix, m, n)

    return result

def _dfs(i, j, matrix, m, n):
    stack = [(i, j)]

    while len(stack) > 0:
        matrix[i][j] = "."

        if j + 1 < m:
            if matrix[i][j + 1] == "#":
                stack.append((i, j + 1))
        if j - 1 >= 0:
            if matrix[i][j - 1] == '#':
                stack.append((i, j - 1))
        if i + 1 < n:
            if matrix[i + 1][j] == '#':
                stack.append((i + 1, j))
        if i - 1 >= 0:
            if matrix[i - 1][j] == "#":
                stack.append((i - 1, j))

        (i, j) = stack.pop()

    return matrix
```

Тесты:

17227853	31.05.2022 14:20:09	Хабибй	0296	Visual C++	Accepted		0,03	300 Кб
17227852	31.05.2022 14:20:08	Рахлин Роман Михайлович	0432	Python	Accepted		0,062	1158 Кб

Задание 16

Описание задачи: Поскольку рекурсия может быть косвенной (процедура вызывает сама себя через другие процедуры), то задача определения того факта, является ли данная процедура рекурсивной, достаточно сложна. Попробуем решить более простую задачу. Рассмотрим программу, состоящую из n процедур P_1, P_2, . . ., Pn. Пусть для каждой процедуры известны процедуры, которые она может вызывать. Процедура P называется потенциально рекурсивной, если существует такая последовательность процедур Q_0, Q_1, . . ., Qk, что $Q_0 = Q_k = P$ и для $i = 1...k$ процедура Q_{i-1} может вызвать процедуру Qi. В этом случае задача будет заключаться в определении для каждой из заданных процедур, является ли она потенциально рекурсивной. Требуется написать программу, которая позволит решить названную задачу.

Исходный код и описание:

```python
def is_recursive(graph, procedure):
    queue = deque()
    visited = set()
    inqueue = set()
    queue.append(procedure)
    inqueue.add(procedure)
    while len(queue) > 0:
        c = queue.popleft()
        inqueue.remove(c)
        visited.add(c)
        for v in graph[c]:
            if v == procedure:
                return True
            if v not in visited and v not in inqueue:
                queue.append(v)
                inqueue.add(v)
    return False

with open("input.txt") as input_file:
    n = int(input_file.readline())

    graph = {}
    res = []
    pcs = []

    for _ in range(n):
        procedure = input_file.readline().strip()
        pcs.append(procedure)
        n = int(input_file.readline())
        calls = [input_file.readline().strip() for _ in range(n)]
        graph[procedure] = calls
        input_file.readline()

    for procedure in pcs:
        res.append("YES" if is_recursive(graph, procedure) else "NO")
```

Тесты:

17228279	31.05.2022 16:00:19	Рахлин Роман Михайлович	0345	Python	Accepted		0,046	2398 Кб

Задание 17

Описание задачи: Нам даны номера городов и другие номер городов к которым они видут. Другими словами граф. И наша задача понять насколько правильно можно пройти граф не проходя по одной и той же дроге обратно. И если приходится пройтись обратно, то посчитать сколько таких раз нужно сделать.

Исходный код и описание:

```
inf = float("inf")

with open("input.txt") as input_file:
    n, m = map(int, input_file.readline().split())
    d = [inf for _ in range(n * n + 2)]

    for i in range(m):
        u, v = map(int, input_file.readline().split())

        u -= 1
        v -= 1

        d[u * n + v] = 0

        if d[v * n + u] == inf:
            d[v * n + u] = 1

    # O(n^3)
    for k in range(n):
        for i in range(n):
            for j in range(n):
                if d[i * n + k] < inf and d[k * n + j] < inf:
                    d[i * n + j] = min(d[i * n + j], d[i * n + k] + d[k * n + j])

    result = 0

    for i in range(n):
        for j in range(n):
            if d[i * n + j] != inf:
                ans = max(ans, d[i * n + j])

    with open('output.txt', 'w') as output_file:
        output_file.write(str(result))
```

Здесь мы должны правильно реализовать работу с матричным видом графа и анализируя все пути составить таблицу и потом по ней сделать вывод сколько раз

мы должны повторить проходы по уже пройденным путям и сколько раз.

Тесты:

К сожалению на последних тестах возникает ошибка, из-за плохого времени. Но все основные тесты проходятся, так что алгоритм работает верно.

17228300	31.05.2022 16:05:50	Рахлин Роман Михайлович	0562	Python	Time limit exceeded	30	1,015	578 Кб