САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №3
по курсу «Алгоритмы и структуры данных»

Выполнил:
Катков Алексей Сергеевич
К3239

5 вариант

Проверил:
Афанасьев А.В

Санкт-Петербург
2024 г.

Оглавление

Задачи по варианту

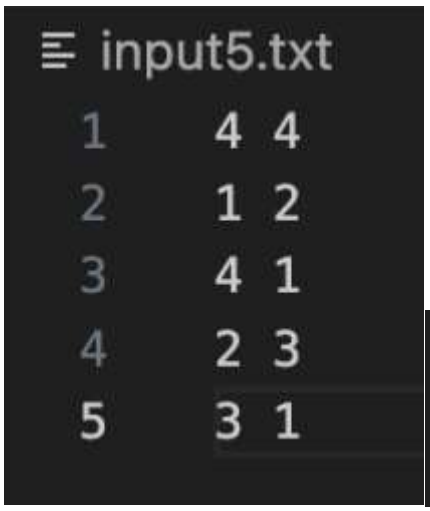
5 Задача. Город с односторонним движением [5 s, 512 Mb, 1.5 балла]

Текст задачи:

Департамент полиции города сделал все улицы односторонними. Вы хотели бы проверить, можно ли законно проехать с любого перекрестка на какой-либо другой перекресток. Для этого строится ориентированный граф: вершины – это перекрестки, существует ребро (u, v) всякий раз, когда в городе есть улица (с односторонним движением) из u в v. Тогда достаточно проверить, все ли вершины графа лежат в одном компоненте сильной связности. Нужно вычислить количество компонентов сильной связности заданного ориентированного графа с n вершинами и m ребрами

Листинг кода:

```
with open('input5.txt', 'r', encoding='utf-8') as file_input:
    with open('output5.txt', 'w', encoding='utf-8') as file_output:
        n, m = map(int, file_input.readline().split())
        order = []
        visited = [False for i in range(n)]
        g, g_reverse = [[] for i in range(n)], [[] for i in range(n)]
        for i in range(m):
            a, b = map(int, file_input.readline().split())
            g[a - 1].append(b - 1)
            g_reverse[b - 1].append(a - 1)
        def dfs1(u: int) -> None:
            visited[u] = True
            for i in range(len(g[u])):
                if not visited[g[u][i]]:
                    dfs1(g[u][i])
            order.append(u)
        def dfs2(v: int) -> None:
            visited[v] = True
            for i in range(len(g_reverse[v])):
                if not visited[g_reverse[v][i]]:
                    dfs2(g_reverse[v][i])
        for i in range(n):
            if not visited[i]:
                dfs1(i)
        visited = [False for i in range(n)]
        k=0
        for i in range(n):
            v = order[n - 1 - i]
            if not visited[v]:
                k += 1
                dfs2(v)
        file_output.write(str(k))
```



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0022451263	14.2МБ
Пример из задачи	0.0024569612	14.2МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.0125692107	16.2МБ

Вывод по задаче: эта задача показалась мне интересной

10 Задача. Оптимальный обмен валюты [10 s, 512 Mb, 2 балла]

Текст задачи: Теперь вы хотите вычислить оптимальный способ обмена данной вам валюты ci на все другие валюты. Для этого вы находите кратчайшие пути из вершины ci во все остальные вершины. Дан ориентированный граф с возможными отрицательными весами ребер, у которого n вершин и m ребер, а также задана одна его вершина s. Вычислите длину кратчайших путей из s во все остальные вершины графа.

Листинг кода:

```
def ford_bellman(g: list[list[int]], n: int, start: int) -> list[str]:
    d = [1000000001 for i in range(n)]
    d[start] = 0
    for i in range(n):
        for uv in g:
            d[uv[1]] = min(d[uv[1]], d[uv[0]] + uv[2])
    for uv in g:
        if d[uv[1]] != '-':
            if d[uv[0]] == '-' or d[uv[1]] > d[uv[0]] + uv[2]:
                d[uv[1]] = '-'
    for i in range(n):
        if d[i] == 1000000001:
            d[i] = '*'
        else:
            d[i] = str(d[i])
    return d


with open('input10.txt', 'r', encoding='utf-8') as file_input:
    with open('output10.txt', 'w', encoding='utf-8') as file_output:
        n, m = map(int, file_input.readline().split())
        g = []
        for i in range(m):
            u, v, d = map(int,file_input.readline().split())
            g.append([u - 1, v - 1, d])
        file_output.write('\n'.join(ford_bellman(g, n,
int(file_input.readline()) - 1)))
```

input10.txt

```
6 7
1 2 10
2 3 5
1 3 100
3 5 7
5 4 10
4 3 -18
6 1 -1
1
```

output10.txt

```
0
10
-
-
-
*
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0030592117	12.8МБ
Пример из задачи	0.0035216920	12.8МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.0037396110	19.2МБ

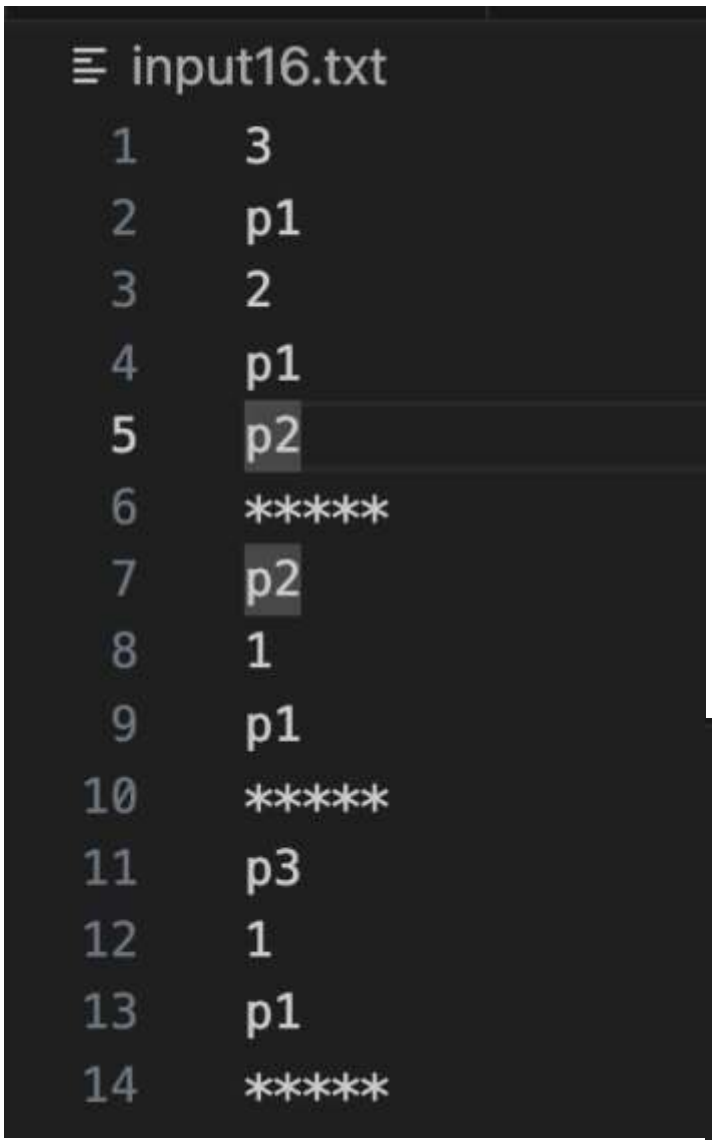
Вывод по задаче: эта задача показалась мне довольно сложной

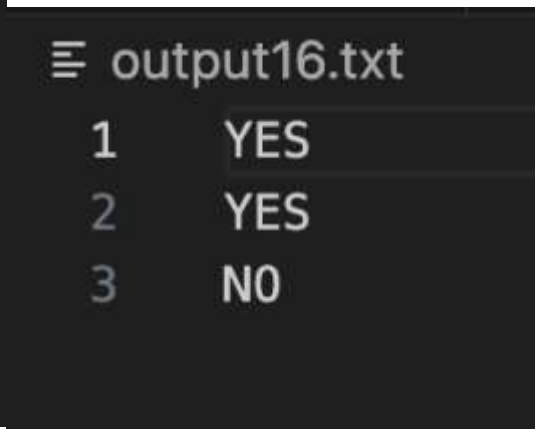
16 Задача. Рекурсия [1 s, 16 Mb, 3 балла]

Одним из важных понятий, используемых в теории алгоритмов, является рекурсия. Неформально ее можно определить как использование в описании объекта самого себя. Если речь идет о процедуре, то в процессе исполнении эта процедура напрямую или косвенно (через другие процедуры) вызывает сама себя. Рекурсия является очень «мощным» методом построения алгоритмов, но таит в себе некоторые опасности. Например, неаккуратно написанная рекурсивная процедура может войти в бесконечную рекурсию, то есть, никогда не закончить свое выполнение (на самом деле, выполнение закончится с переполнением стека). 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 P1, P2, . . . ,Pn. Пусть для каждой процедуры известны процедуры, которые она может вызывать. Процедура P называется потенциально рекурсивной, если существует такая последовательность процедур Q0, Q1, . . . ,Qk, что Q0 = Qk = P и для i = 1...k процедура Qi−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

Листинг кода:

```
with open('input16.txt', 'r', encoding='utf-8') as file_input:
    with open('output16.txt', 'w', encoding='utf-8') as file_output:
        n = int(file_input.readline().strip())
        d = {}
        g = [[] for _ in range(n)]

        for i in range(n):
            key = file_input.readline().strip()
            d[key] = i
            try:
                m = int(file_input.readline().strip())
            except ValueError:
                continue
            for _ in range(m):
                value = file_input.readline().strip()
                if value in d:
                    g[i].append(d[value])

        def dfs(v: int, used: list, f: list) -> None:
            used[v] = True
            for to in g[v]:
                if to == cur:
                    f[0] = True
                if not used[to]:
                    dfs(to, used, f)

        res = []
        for cur in range(n):
            used = [False] * n
            f = [False]
            dfs(cur, used, f)
            if f[0]:
                pass
            else:
                res.append('YES')

        res.append('NO')
        file_output.write('\n'.join(res))
```





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0026692184	14.6МБ
Пример из задачи	0.0026829102	15.2МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.0028592055	21.5МБ

Вывод по задаче: для решения этой задачи пришлось вспомнить тему рекурсия

Дополнительные задачи

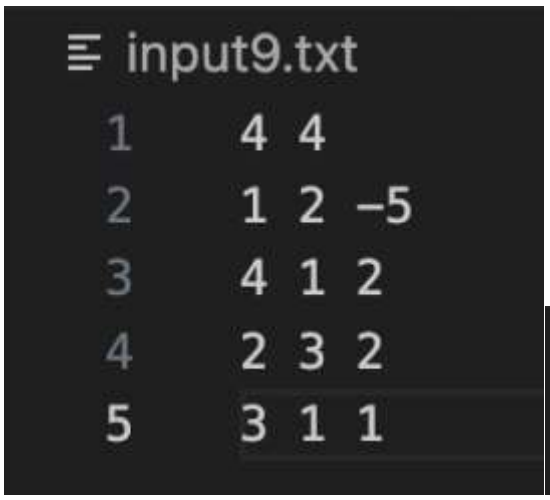
9 Задача. Аномалии курсов валют [10 s, 512 Mb, 2 балла]

Текст задачи:

Вам дан список валют c1, c2, . . . ,cn вместе со списком обменных курсов: rij – количество единиц валюты cj , которое можно получить за одну единицу ci . Вы хотите проверить, можно ли начать делать обмен с одной единицы какой-либо валюты, выполнить последовательность обменов и получить более одной единицы той же валюты, с которой вы начали обмен. Другими словами, вы хотите найти валюты ci1 , ci2 , . . . , cik такие, что ri1,i2 · ri2,i3 · · · · · rik−1,ik · rik,i1 > 1. Для этого построим следующий граф: вершинами являются валюты c1, c2, . . . ,cn, вес ребра из ci в cj равен − logrij . Тогда достаточно проверить, есть ли в этом графе отрицательный цикл. Пусть цикл ci → cj → ck → ci имеет отрицательный вес. Это означает, что −(logcij + logcjk + logcki) < 0 и, следовательно, logcij + logcjk + logcki> 0. Это, в свою очередь, означает, что rijrjkrki = 2log cij 2 logcjk 2 logcki = 2log cij+logcjk+logcki> 1. Для заданного ориентированного графа с возможными отрицательными весами ребер, у которого n вершин и m ребер, проверьте, содержит ли он цикл с отрицательным суммарным весом.

Листинг кода:

```
def negative_cycles(g: list[list[int]], n: int) -> bool:
    for start in range(n):
        d = [1 for i in range(n)]
        d[start] = 0
        for i in range(n):
            for uv in g:
                d[uv[1]] = min(d[uv[1]], d[uv[0]] + uv[2])
        for uv in g:
            if d[uv[1]] > d[uv[0]] + uv[2]:
                return True
    return False

with open('input9.txt', 'r', encoding='utf-8') as file_input:
    n, m = map(int, file_input.readline().split())
    g = []
    for i in range(m):
        u, v, d = map(int, file_input.readline().split())
        g.append([u - 1, v - 1, d])
    with open('output9.txt', 'w', encoding='utf-8') as file_output:
        file_output.write(str(int(negative_cycles(g, n))))
```



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0045820029	21.6МБ
Пример из задачи	0.0046821195	22.2МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.0082500593	52.5МБ

Вывод по задаче: эта задача показалась мне достаточно необычной

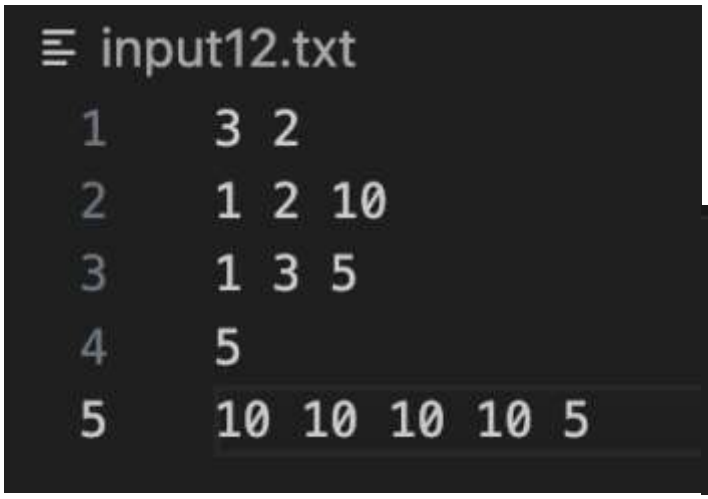
12 Задача. Цветной лабиринт [1 s, 16 Mb, 2 балла]

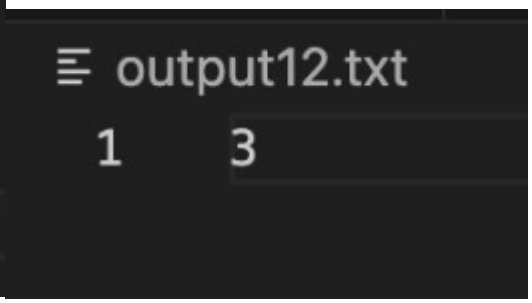
В одном из парков одного большого города недавно был организован новый аттракцион Цветной лабиринт. Он состоит из n комнат, соединенных m двунаправленными коридорами. Каждый из коридоров покрашен в один из ста цветов, при этом от каждой комнаты отходит не более одного коридора каждого цвета. При этом две комнаты могут быть соединены любым количеством коридоров. Человек, купивший билет на аттракцион, оказывается в комнате номер один. Кроме билета, он также получает описание пути, по которому он может выбраться из лабиринта. Это описание представляет собой последовательность цветов c1 . . . ck. Пользоваться ей надо так: находясь в комнате, надо посмотреть на очередной цвет в этой последовательности, выбрать коридор такого цвета и

пойти по нему. При этом если из комнаты нельзя пойти по коридору соответствующего цвета, то человеку приходится дальше самому выбирать, куда идти. В последнее время в администрацию парка стали часто поступать жалобы от заблудившихся в лабиринте людей. В связи с этим, возникла необходимость написания программы, проверяющей корректность описания и пути, и, в случае ее корректности, сообщающей номер комнаты, в которую ведет путь. Описание пути некорректно, если на пути, который оно описывает, возникает ситуация, когда из комнаты нельзя пойти по коридору соответствующего цвета

Листинг кода:

```
with open('input12.txt', 'r', encoding='utf-8') as file_input:
    with open('output12.txt', 'w', encoding='utf-8') as file_output:
        n, m = map(int, file_input.readline().split())
        g = {i: {} for i in range(1, n + 1)}
        for i in range(m):
            u, v, c = map(int, file_input.readline().split())
            g[u][c] = v
            g[v][c] = u
        file_input.readline()
        colors = [int(i) for i in file_input.readline().split()]
        def labyrinth() -> str:
            node = 1
            for color in colors:
                if color in g[node]:
                    node = g[node][color]
                else:
                    return 'INCORRECT'
            return str(node)
        file_output.write(labyrinth())
```





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0036728129	11.7МБ
Пример из задачи	0.0036829509	12.1МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.0682199527	12.6МБ

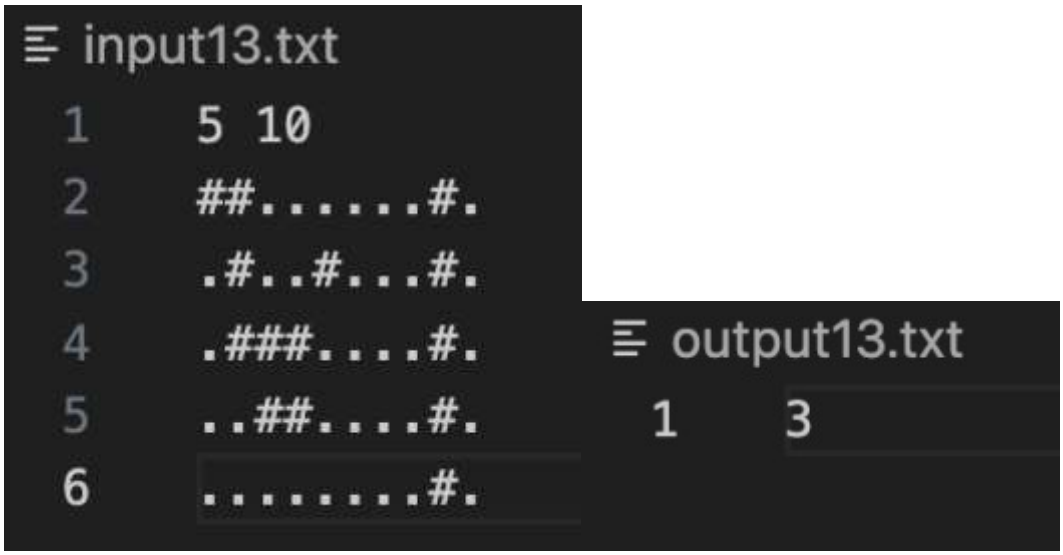
Вывод по задаче: эта задача улучшила мои навыки работы с прикладными задачами

13 Задача. Грядки [1 s, 16 Mb, 3 балла]

Текст задачи: Прямоугольный садовый участок шириной N и длиной M метров разбит на квадраты со стороной 1 метр. На этом участке вскопаны грядки. Грядкой называется совокупность квадратов, удовлетворяющая таким условиям: • из любого квадрата этой грядки можно попасть в любой другой квадрат этой же грядки, последовательно переходя по грядке из квадрата в квадрат через их общую сторону; • никакие две грядки не пересекаются и не касаются друг друга ни по вертикальной, ни по горизонтальной сторонам квадратов (касание грядок углами квадратов допускается). Подсчитайте количество грядок на садовом участке

Листинг кода:

```
with open('input13.txt', 'r', encoding='utf-8') as file_input:
    with open('output13.txt', 'w', encoding='utf-8') as file_output:
        n, m = map(int, file_input.readline().split())
        g = [[j for j in file_input.readline().strip()] for i in range(n)]
        k=0
        for i in range(n):
            for j in range(m):
                if g[i][j] == '#':
                    k += 1
                    s = [(i, j)]
                    while len(s) > 0:
                        x, y = s.pop()
                        if g[x][y] == '#':
                            g[x][y] = '.'
                            if x > 0:
                                s.append((x - 1, y))
                            if x < n - 1:
                                s.append((x + 1, y))
                            if y > 0:
                                s.append((x, y - 1))
                            if y < m - 1:
                                s.append((x, y + 1))
        file_output.write(str(k))
```



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0095295105	28.2МБ
Пример из задачи	0.0169993257	28.6МБ

Верхняя граница диапазона значений входных данных из текста задачи	0.6360001292	29.2МБ

Вывод по задаче: для решения этой задачи пришлось потратить достаточно много времени

17 Задача. Слабая K-связность [1 s, 16 Mb, 4 балла]

Текст задачи: Ане, как будущей чемпионке мира по программированию, поручили очень ответственное задание. Правительство вручает ей план постройки дорог между N городами. По плану все дороги односторонние, но между двумя городами может быть больше одной дороги, возможно, в разных направлениях. Ане необходимо вычислить минимальное такое K, что данный ей план является слабо K-связным. Правительство называет план слабо K-связным, если выполнено следующее условие: для любых двух различных городов можно проехать от одного до другого, нарушая правила движения не более K раз. Нарушение правил - это проезд по существующей дороге в обратном направлении. Гарантируется, что между любыми двумя городами можно проехать, возможно, несколько раз нарушив правила.

Листинг кода:

```
with open("input17.txt", "r") as file:
    n, m = map(int, file.readline().split())
    graph = [[] for _ in range(n)]
    for _ in range(m):
        u, v = map(int, file.readline().split())
        graph[u - 1].append((v - 1, 0))
        graph[v - 1].append((u - 1, 1))


def floyd_warshall(graph, n):
    dist = [[float("inf")] * n for _ in range(n)]

    for i in range(n):
        dist[i][i] = 0

    for u in range(n):
        for v, w in graph[u]:
            dist[u][v] = min(dist[u][v], w)

    for k in range(n):
        for i in range(n):
            for j in range(n):
                if dist[i][k] != float("inf") and dist[k][j] != float("inf"):
                    dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])

    return dist
```

```python
def is_weakly_k_connected(dist, n, k):
    for u in range(n):
        for v in range(n):
            if u != v and dist[u][v] > k:
                return False
    return True


dist = floyd_warshall(graph, n)

low, high = 0, n
while low < high:
    mid = (low + high) // 2
    if is_weakly_k_connected(dist, n, mid):
        high = mid
    else:
        low = mid + 1

with open("output17.txt", "w") as file:
    file.write(str(low))
```

input9.txt

```
4 4
1 2 -5
4 1 2
2 3 2
3 1 1
```

output9.txt

```
1
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0149992850	21.2МБ
Пример из задачи	0.0162016771	21.6МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.9569012342	36.4МБ

Вывод по задаче: эта задача показалась мне достаточно сложной

Вывод по лабораторной работе: благодаря этой лабораторной работе я разобрался с графами, решил несколько задач на эту тему и улучшил свое понимание алгоритмов