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

Отчет по лабораторной работе №3
по курсу «Алгоритмы и структуры данных»
Тема: Графы
Вариант 7

Выполнил:
Крылов Михаил Максимович
К3240

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Оглавление

Содержание отчета 2

Задачи по варианту 3

Задача №1. Лабиринт [5 s, 512 Mb, 1 балл] 3

Задача №12. Цветной лабиринт [1 s, 16 Mb, 2 балла] 6

Задача №14. Автобусы [1 s, 16 Mb, 3 балла] 9

Вывод 12

Задачи по варианту

Задача №1. Лабиринт [5 s, 512 Mb, 1 балл]

Вам дан неориентированный граф и две различные вершины u и v. Проверьте, есть ли путь между u и v.

• Формат ввода / входного файла (input.txt). Неориентированный граф с n вершинами и m ребрами по формату 1. Следующая строка после ввода всего графа содержит две вершины u и v. • Ограничения на входные данные. $2 \le n \le 103$, $1 \le m \le 103$, $1 \le u, v \le n$, $u \neq v$.

• Формат вывода / выходного файла (output.txt). Выведите 1, если есть путь между вершинами u и v; выведите 0, если пути нет.

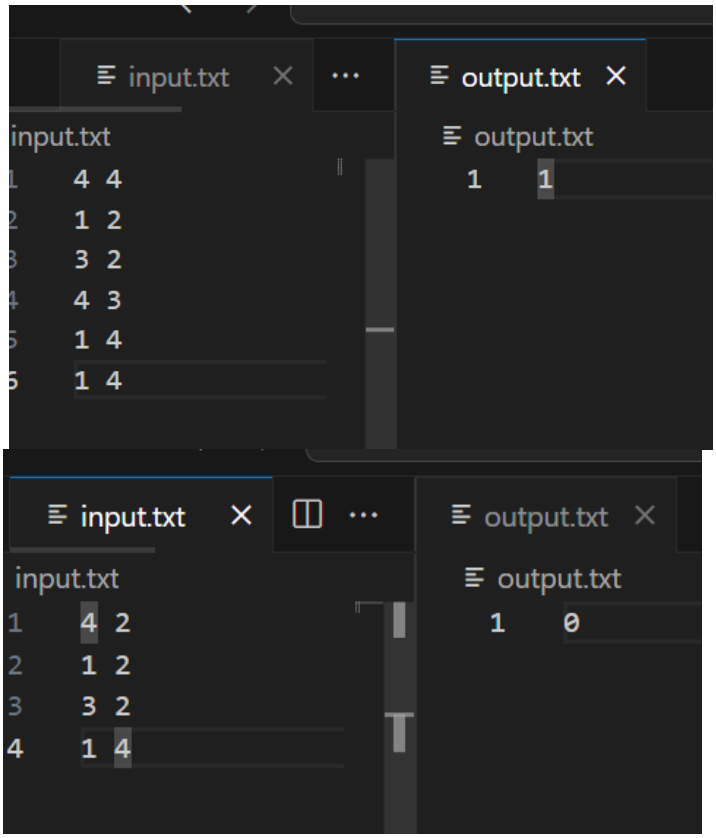
• Ограничение по времени. 5 сек.

• Ограничение по памяти. 512 мб. Листинг кода

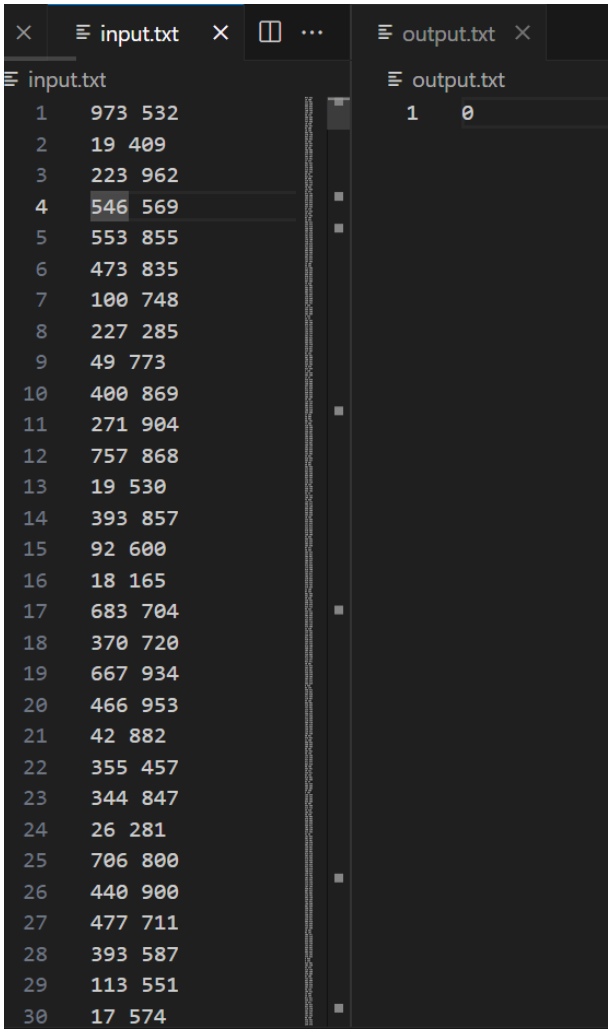
```python
from copy import deepcopy

def func(U, V, ways):
    if V in ways[U] or U in ways[V]:
        return 1

    max_ = 0

    for i in ways[U]:
        new_ways = deepcopy(ways)
        new_ways[U].remove(i)
        new_ways[i].remove(U)
        max_ = max(func(i, V, new_ways), max_)

        if max_ == 1:
            return 1

    return max_

def main():
    with open("input.txt", "r") as file:
        n, m = map(int, file.readline().split())
        ways = {i: [] for i in range(1, n+1)}
        for i in range(m):
            u, v = map(int, file.readline().split())
            ways[u].append(v)
            ways[v].append(u)

        U, V = map(int, file.readline().split())

    result = func(U, V, ways)
```

```
    with open("output.txt", "w") as file:
        file.write(str(result))

if __name__ == "__main__":
    main()
```

При запуске функции считываем данные с файла, создаем словарь, в котором храним пути от узлов к узлу. Далее запускаем функцию поиска пути. Функция рекурсивна, чтоб не было бесконечной рекурсии она копирует пройденный путь и удаляет узлы, в которых была.



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.009 секунды	0.017 МБ
Пример из задачи	0.010 секунды	0.018 МБ
Пример из задачи	0.009 секунды	0.018 МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.021 секунды	0.476 МБ

Вывод по задаче: реализовали обход графов с помощью рекурсивной функции.

Задача №12. 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.

• Формат входных данных (input.txt) и ограничения. Первая строка входного файла INPUT.TXT содержит два целых числа n ($1 \le n \le 10000$) и m ($1 \le m \le 100000$) - соответственно количество комнат и коридоров в лабиринте. Следующие m строк содержат описания коридоров. Каждое описание содержит три числа u ($1 \le u \le n$), v ($1 \le v \le n$), c ($1 \le c \le 100$) - соответственно номера комнат, соединенных этим коридором, и цвет коридора. Следующая, (m + 2)-ая строка входного файла содержит длину описания пути - целое число k ($0 \le k \le 100000$). Последняя строка входного файла содержит k целых чисел, разделенных пробелами, - описание пути по лабиринту.

• Формат выходных данных (output.txt). В выходной файл OUTPUT.TXT выведите строку INCORRECT, если описание пути некорректно, иначе выведите номер комнаты, в которую ведет описанный путь. Помните, что путь начинается в комнате номер один.

• Ограничение по времени. 1 сек.

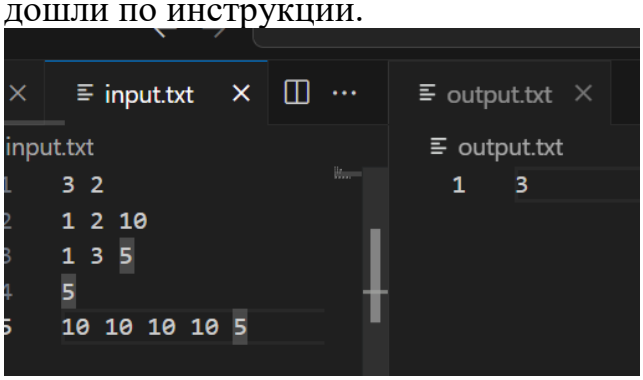
• Ограничение по памяти. 16 мб.

Листинг кода

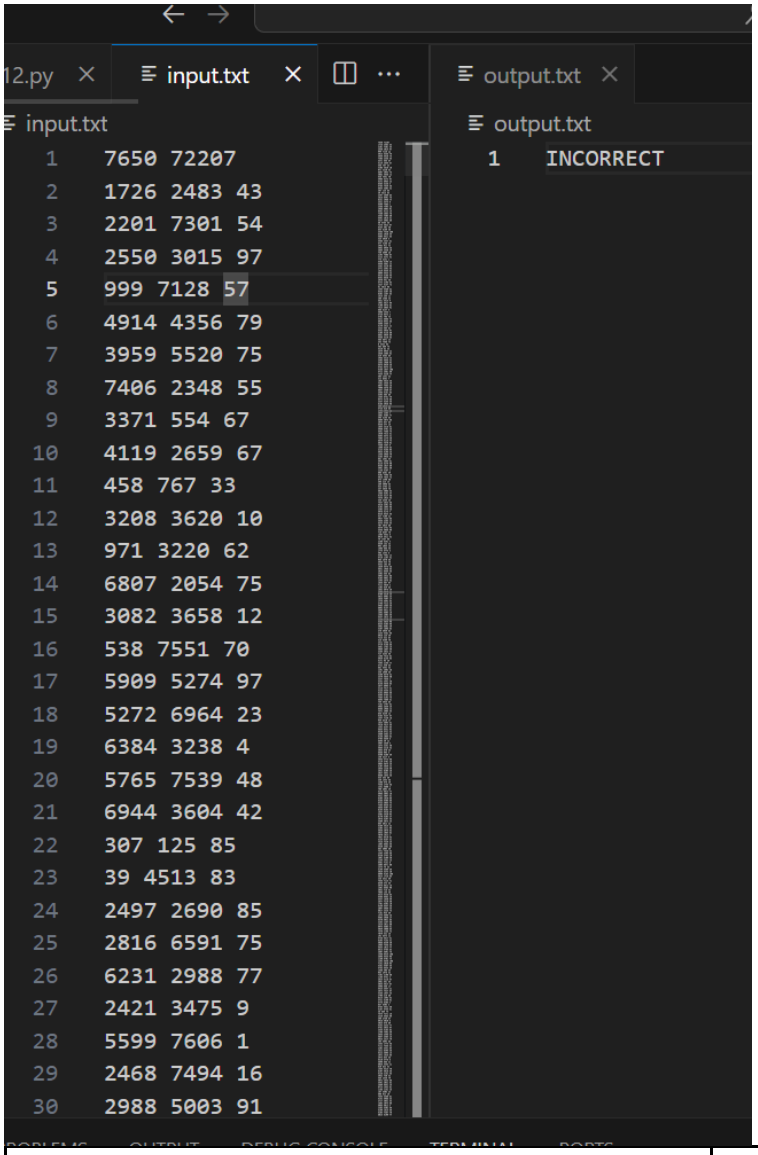
```python
def func(ways, colors):
    room = 1
    for i in colors:
        if i not in ways[room]:
            return "INCORRECT"
        room = ways[room][i]

    return room

def main():
    with open("input.txt", "r") as file:
        n, m = map(int, file.readline().split())
        ways = {i: {} for i in range(1, n+1)}
        for i in range(m):
            u, v, c = map(int, file.readline().split())
            ways[u][c] = v
            ways[v][c] = u
```

```
        file.readline()

        colors = list(map(int, file.readline().split()))

    result = func(ways, colors)

    with open("output.txt", "w") as file:
        file.write(str(result))

if __name__ == "__main__":
    main()
```

после запуска кода считываем все данные и записываем их в словарь. Не забываем про обратные пути, т.к. по коридору можно пройти в две стороны. Далее запускаем функцию. Функция циклом проходит по инструкции, в случае, если цвет коридора не найден среди доступных коридоров – выводит “INCORRECT”. Иначе – выводит конечную комнату, в которую дошли по инструкции.



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001 секунды	0.012 МБ
Пример из задачи	0.001 секунды	0.019 МБ
Верхняя граница диапазона значений входных данных из текста задачи	1.086 секунды	11.023 МБ

Задача №14. Автобусы [1 s, 16 Mb, 3 балла]

Между некоторыми деревнями края Власюки ходят автобусы. Поскольку пассажиропотоки здесь не очень большие, то автобусы ходят всего несколько раз в день. Марии Ивановне требуется добраться из деревни d в деревню v как можно быстрее (считается, что в момент времени 0 она находится в деревне d).

• Формат входных данных (input.txt) и ограничения. Во входном файле INPUT.TXT записано число N - общее число деревень ($1 \le N \le 100$), номера деревень d и v, затем количество автобусных рейсов R ($0 \le R \le 10000$). Затем идут описания автобусных рейсов. Каждый рейс задается номером деревни отправления, временем отправления, деревней назначения и временем прибытия (все времена - целые от 0 до 10000). Если в момент t пассажир приезжает в деревню, то уехать из нее он может в любой момент времени, начиная с t.

• Формат выходных данных (output.txt). В выходной файл OUTPUT.TXT вывести минимальное время, когда Мария Ивановна может оказаться в деревне v. Если она не сможет с помощью указанных автобусных рейсов добраться из d в v, вывести -1.

• Ограничение по времени. 1 сек.

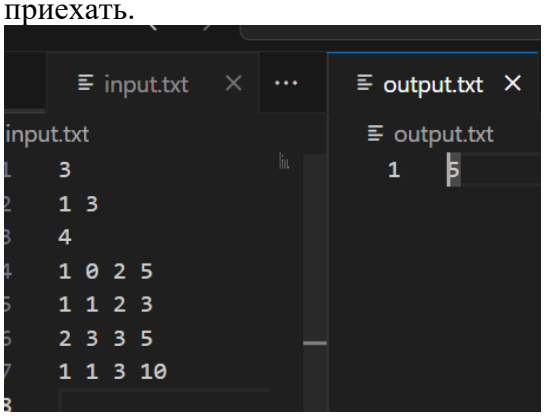
• Ограничение по памяти. 16 мб

Листинг кода

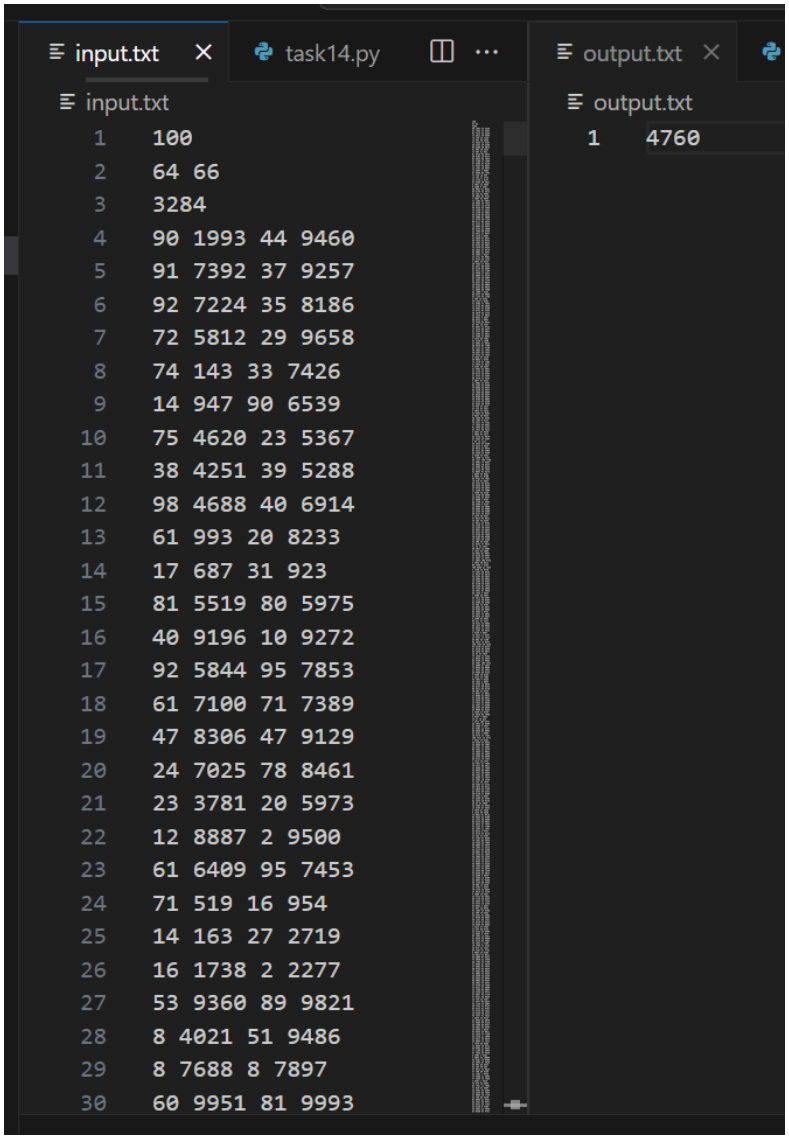
```python
def func(ways, d, v, time=0, visited=set()):
    if d == v:
        return time

    visited.add(d)
    min_time = float('inf')

    for trip in ways[d]:
        t_depart, next_village, t_arrive = trip
        if time <= t_depart and next_village not in visited:
            result_time = func(ways, next_village, v, t_arrive, visited)
            if result_time < min_time:
                min_time = result_time

    visited.remove(d)

    return min_time

def main():
    with open("input.txt", "r") as file:
        N = int(file.readline())
        d, v = map(int, file.readline().split())
        ways = {i+1: [] for i in range(N)}
        for i in range(int(file.readline())):
            a1, t1, a2, t2 = map(int, file.readline().split())
            ways[a1].append((t1, a2, t2))

    result = func(ways, d, v)

    with open("output.txt", "w") as file:
        if result == float('inf'):
            result = -1
```

```
        file.write(str(result))

if __name__ == "__main__":
    main()
```

Считываем данные с файла, создаем словарь с элементами от 1 до N, по умолчанию значение будет пустым списком. При дальнейшем считывании с файлом добавляем по ключу данные о времени отправлении, времени прибытия и город прибытия. Потом запускаем основную функцию, в которой создаем множество, в которое будем записывать посещенные города. Разбиваем кортеж и записываем его данные в 3 переменные. Проверяем, чтоб следующий по списку город не был посещен, и запускаем рекурсивную функцию. В конце возвращаем минимальное время, за которое можем приехать.



Результат работы кода на максимальных и минимальных значениях:



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.007 секунды	0.016 МБ
Пример из задачи	0.009 секунды	0.018 МБ
Верхняя граница диапазона значений входных данных из текста задачи	0.517 секунды	3.388 МБ

Вывод по задаче: реализовали алгоритм нахождения пути по графам с минимальной суммой.

Вывод

Данная лабораторная работа закрепила знания работы с графами.