|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО (УЧЕБНОЙ/ПРОИЗВОДСТВЕННОЙ) ПРАКТИКЕ**

Студент Богатырев Иван Сергеевич

*фамилия, имя, отчество*

Группа ИУ7-15Б

Тип практики Учебно-вычислительный практикум

Название практики Нумерация пути

Студент  **Богатырев И.С.**

*подпись, дата фамилия, и.о.*

Руководитель практики  **Кузнецова О.В.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. *г.*

**Оглавление**

1. Условие задачи
2. Введение
3. Основная часть
4. Схема программы
5. Заключение
6. Список литературы

**Текст задачи**

Нумерация пути

**Введение**

Задачи, которые осуществляют ввод и генерируют простой ответ «да» или «нет», называются задачами нахождения решения. Один из подклассов таких задач – завершенные задачи нумерации пути – не относится к разряду задач на нахождение эффективного решения. Другие задачи могут быть такими же простыми, как задачи нахождения решения, но подсчет всех возможных положительных ответов может оказаться очень сложным, или, по крайней мере, требующим массу времени.

**Задача**

В городе даны узловые точки, соединенные односторонними дорогами. Необходимо написать программу, которая определяет число всех различных путей между каждым узловым пунктом.

Пункты обозначены неотрицательными целыми числами. Односторонний путь задается парой пунктов. Например, j и k обозначает, что существует односторонняя улица из пункта j в пункт k. Заметьте, что двусторонний путь может быть смоделирован определением двух односторонних улиц: j k и k j.

Предположим, что город с четырьмя узловыми пунктами соединен следующими односторонними дорогами:

0 1

0 2

1 2

2 3

Существует один маршрут из пункта 0 в 1, два маршрута из 0 в 2 (0 🡪 1 🡪 2 и 0 🡪 2) и один из 2 в 3. Других путей не существует.

Возможно существование бесконечного числа различных маршрутов. Например, если к вышеуказанным пунктам добавить улицу 3 2, то количество маршрутов из пункта 0 в пункт 1 останется прежним (один), а из 0 в 2 их окажется бесконечно много. Это происходит потому, что проход по улицам из пункта 2 в пункт 3 и обратно может быть повторен бесконечно много раз, порождая новые комбинации и, следовательно, новые маршруты. Таким образом, маршруты 0 🡪 2 🡪 3 🡪 2 🡪 3 🡪 2 и 0 🡪 2 🡪 3 🡪 2 отличаются друг от друга.

**Ввод**  
Ввод представляет собой последовательность спецификаций городов. Каждая спецификация начинается с числа, обозначающего количество односторонних улиц в городе, и самими улицами, заданными как пары узловых пунктов. Каждая пара j k обозначает одностороннюю улицу из пункта j в пункт k. Во всех городах пункты нумеруются с нуля по возрастанию. Все введенные числа разделяются пробелами. Ввод прерывается символом конца файла.

Дорог, ведущих из одного пункта в тот же самый, быть не может. Город имеет не более 30 пунктов.

**Вывод**

Для каждого города напечатать матрицу, состоящую из чисел, обозначающих количество различных путей из пункта j в пункт k. Если М – имя матрицы, то M[j][k] есть количество различных путей из пункта j в пункт k. Матрица М должна быть напечатана в порядке приоритета строк. Каждой матрице должна предшествовать строка “matrix for city k” (где k – номер соответствующего города, начиная с нуля).

Если между двумя пунктами существует бесконечное количество маршрутов, вывести -1. НЕ ЗАБОТЬТЕСЬ о выравнивании и центровке строк в матрицах. Все значения следует выводить через пробел.

**Пример ввода Пример вывода**

7 0 1 0 2 0 4 2 4 2 matrix for city 0

3 3 1 4 3 0 4 1 3 2

5 0 0 0 0 0

0 2 0 2 0 2 1

0 1 1 5 2 5 2 1 0 1 0 0 0

9 0 1 0 1 0

0 1 0 2 0 3 matrix for city 1

0 4 1 4 2 1 0 2 1 0 0 3

2 0 0 0 0 0 0 0

3 0 0 1 0 0 0 2

3 1 0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

matrix for city 2

-1 -1 -1 -1 -1

0 0 0 0 1

-1 -1 -1 -1 -1

-1 -1 -1 -1 -1

0 0 0 0 0

**Введение**

Передо мной была поставлена задача, суть которой заключается в поиске количества путей между всеми узлами графа. Задача была осложнена наличием двусторонних дорог между пунктами, которые создают петлю, образующую бесконечное количество возможных путей.

Входные данные должны считываться из текстового файла, причём данные не обязательно расположены ровно по строкам. То есть на одной строке могут оказаться не все пары дорог данного города или пары нескольких городов.

Обычным перебором задачу решать неэффективно, поэтому нужно было придумать другое, более эффективное решение.

В конце программы нужно вывести полученные данные в виде матрицы.

В конечном счете можно сформулировать такой список задач:

1. Считывание файла
2. Преобразование полученных данных в более удобный вид
3. Создание быстрого и эффективного алгоритма высчитывания количества путей между узлами графа
4. Вывод полученных данных в нужной форме

**Основная часть**

Считывание файла осуществляется построчным чтением файла input.txt. Каждая строка разделяется на отдельные числа, которые впоследствии вносятся в массив numbers.

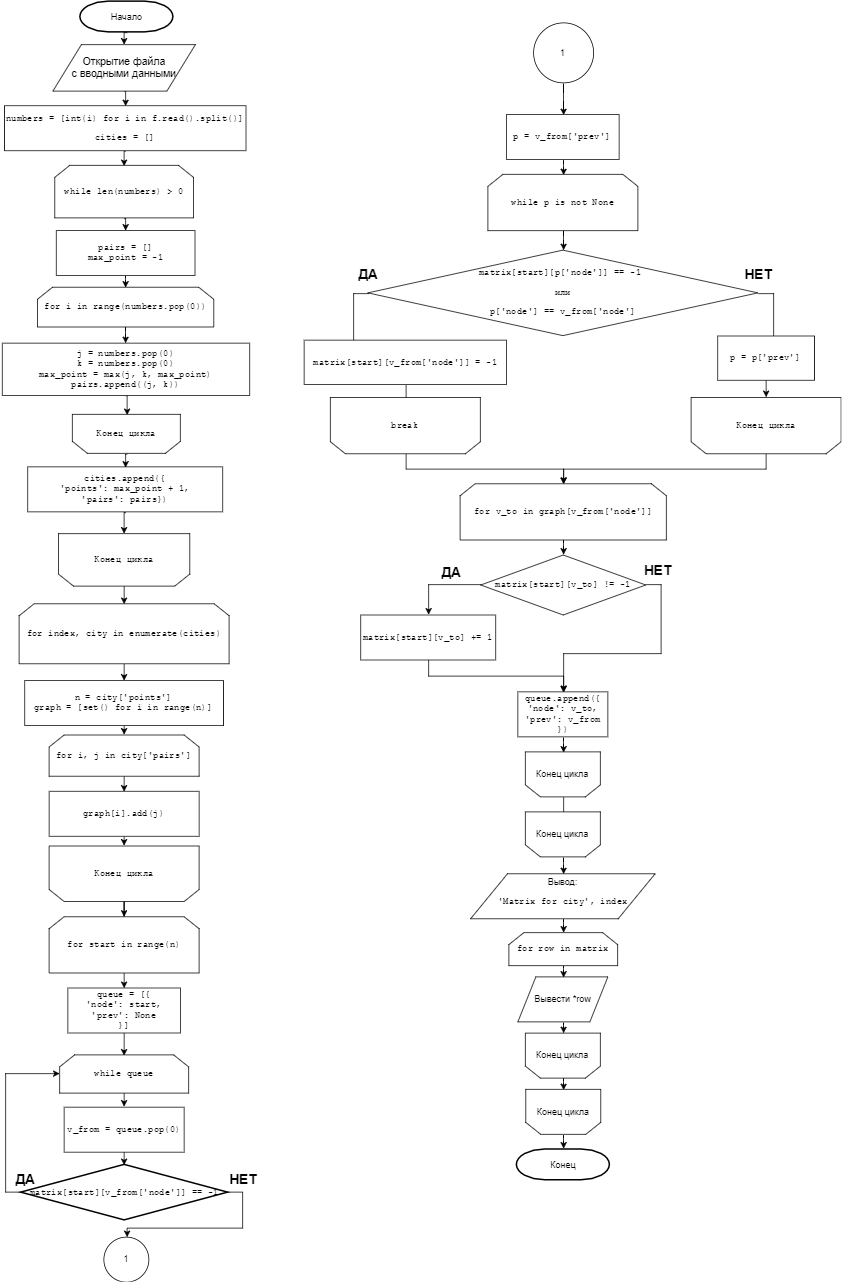
Далее для каждого города определяется количество точек, а также происходит разбивка на пары. Данные вносятся в массив cities, где каждый элемент – словарь, в котором первый ключ ‘points’ отвечает за количество пунктов в городе, а ‘pairs’ – за пары точек, которые обозначают пути.

Позже для каждого города создается массив из множеств точек, в которые можно попасть из точки с номером индекса элемента.

Главной особенностью программы является то, что в очереди (массив queue) каждый элемент - это не только номер пункта, но и путь до него в виде связного списка. Связный список является простейшим типом данных динамической структуры, состоящей из элементов (узлов). В отличие от массива, в котором порядок определяется индексами, порядок в связанном списке определяется указателями на каждый объект. Связанные списки обеспечивают простое и гибкое представление динамических множеств.

Далее весь порядок действий представлен в виде блок схемы.

**Схема программы**



**Заключение**

В ходе выполнения данной практики я укрепил знания, полученные в ходе теоретического и практического курса программирования в первом семестре, а также получил опыт работы с динамическими структурами данных.

**Список литературы**

1. Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн – “Introduction to algorithms” (третье издание), «Вильямс», 2013
2. Васильев А.Н. – “Python на примерах” (второе издание), НиТ, 2017
3. Связные списки - <https://www.ibm.com/developerworks/ru/library/l-data_structures_02/index.html>
4. http://www.williamspublishing.com/PDF/5-8459-0926-0/part.pdf