|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования 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 ЛАБОРАТОРНОЙ РАБОТЕ №8**

**«Графы»**

Студент Фам Минь Хиеу

Группа ИУ7 – 32Б

Преподаватель Никульшина Т.А.

*2022 г.*

**Цель работы**

Реализовать алгоритмы обработки графовых структур: поиск различных путей, проверку связности, построение остовых деревьев минимальной стоимости.

**Условие задачи**

В системе двусторонних дорог за проезд каждой дороги взимается некоторая пошлина. Найти путь из города A в город B с минимальной величиной S+P, где S - сумма длин дорог пути, а P - сумма пошлин проезжаемых дорог

**Входные данные**

Целое число – количество вершин

Пути (из какого города в какой), расстояния между ними, пошлины дорог

Из какого города в какой требуется найти минимальный путь

**Выходные данные**

Построенный граф, минимальный путь

**Обращение к программе:**

Запускается через ./app.exe

**Структуры данных:**

структура данных для матрицы стоимостей:

*struct table\_t*

*{*

*int size;*

*int \*\*matrix;*

*};*

где

* int size – размер матрицы
* int \*\*matrix – элементы матрицы

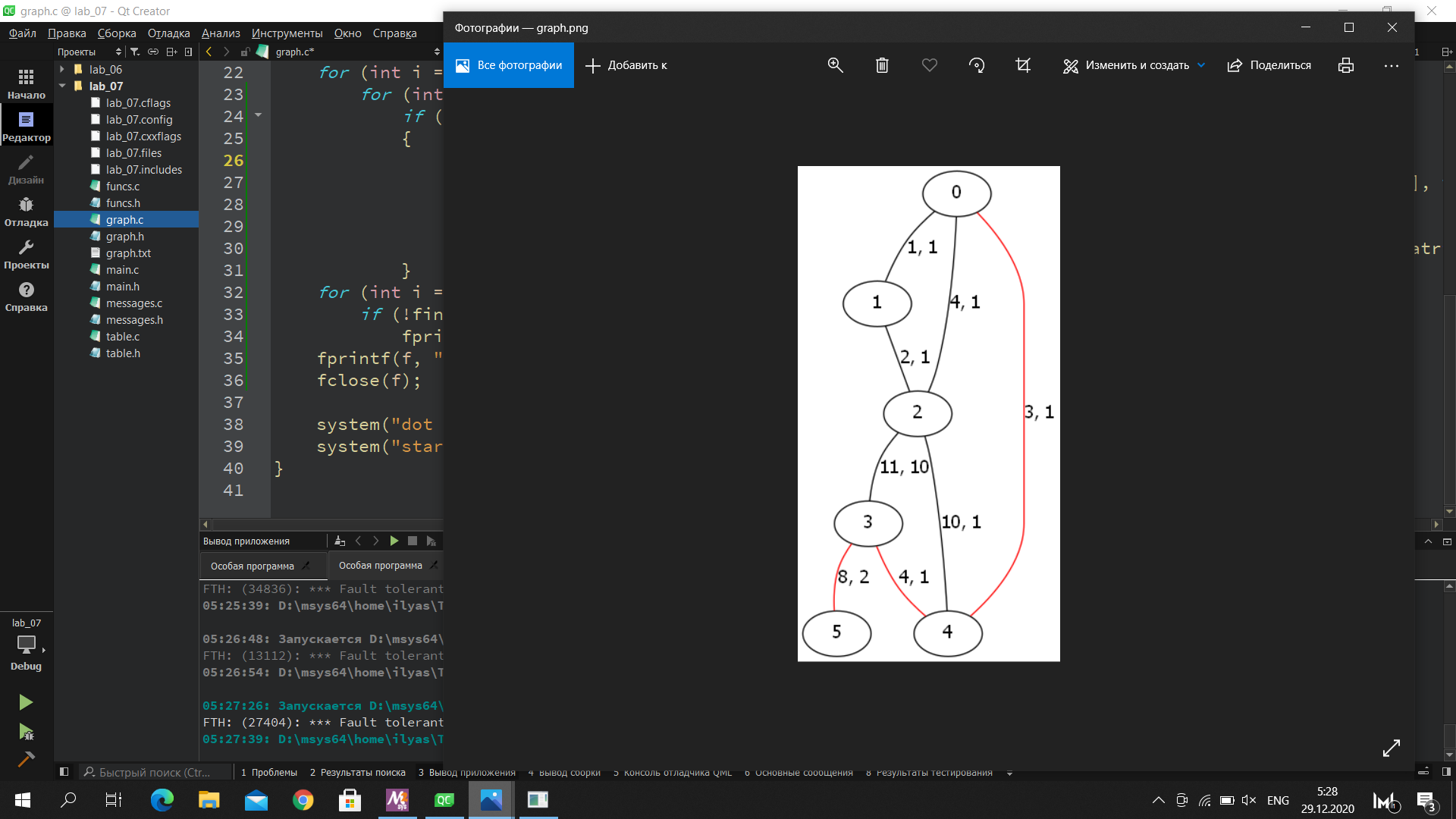
**Описание алгоритма**

У пользователя спрашивается количество вершин n в будущем графе. Затем выделяется память под матрицу с количеством строк n, а столбцов – в два раза больше, так как задача усложняется тем, что помимо длины пути есть значение пошлины. То есть заполняется по две ячейки сразу. Таким образом, получается матрица стоимостей, она симметрична. Если путь отсутствует, то роль бесконечности в матрице выполняет значения половины от максимального целого. Для нахождения кратчайшего пути использовался алгоритм Дейкстры. Перед этим создается вспомогательная матрица стоимостей, но уже квадратная. Каждый ее элемент является суммой пары длины + пошлина. Алгоритм Дейкстры позволяет найти расстояние от одной вершины до всех других. Изначально рассматриваем всех соседей исходной вершины. Кратчайшие расстояния – это и будет прямыми путями к ним. Запоминаем их. Исходная вершина отмечается посещенной. Далее рассматриваем ближайшую не посещенную вершину и смотрим на ее соседей. Если вершина уже была посещена, то в расчет ее не берем. Находим с минимальной меткой и проверяем будет ли путь через текущую вершину короче, чем уже указанный. Если да, то заменяем. Таким образом проходим по всем не посещенным вершинам и в итоге получаем метки на каждой вершине (если она достижима) и среди них уже можем выбрать нужную. Потом нужно получить сам путь. Движемся в обратную сторону от конечной вершины. Смотрим на метки соседей и отнимаемой от текущей вес ребра по отдельности. Если получившееся значение совпало с меткой, то именно отсюда был осуществлен переход. Двигаемся так до тех пор, пока не вернемся в исходную вершину. В результате получаем массив целочисленных значений, в котором содержатся вершины минимального пути. После нахождения граф записывается в текстовый файл на языке dot и строится изображение графа при помощи GraphViz, в консоль выводится информация о минимальном пути.

**Набор тесты**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользователь вводит** | **Вывод** |
| 1 | Число является отрицательным | -1 | Введено неверно! |
| 2 | Нулевое число | 0 | Введено неверно! |
| 3 | Введено не число | aa | Введено неверно! |
| 4 | Первая вершина отрицательна | 4  -1 2 2 3 | Введено неверно! |
| 5 | Вторая вершина отрицательна | 4  1 -1 2 2 | Введено неверно! |
| 6 | Значение вершины больше максимально возможного | 4  1 4 2 3 | Введено неверно! |
| 7 | Отрицательная длина дороги | 4  1 4 -2 2 | Введено неверно! |
| 8 | Нет ни одного пути между вершинами, указанными для поиска | 3  0 2 2 2  0 1 4 3  1 2 | Нет пути! |
| 9 | Корректные данные, поиск минимального пути | 6  0 1 1 1  1 2 2 1  2 3 11 10  3 5 8 2  3 4 4 1  4 0 3 1  0 2 4 1  4 2 10 1  -1 | Откуда - куда: 0 5  Минимальная S + P = 19  0 -> 4 -> 3 -> 5  Время работы: 7098  Памяти: 464 |

Графическое представление графа (красным помечан минимальный путь):



**Вывод**

Проведем анализ эффективности выбранного алгоритма Дейкстры.

|  |  |  |
| --- | --- | --- |
| Количество вершин / количество ребер | Время | Память |
| 2 / 1 | 5061 | 80 |
| 4 / 1 | 6069 | 224 |
| 4 / 3 | 6384 | 224 |
| 4 / 6 | 6678 | 224 |
| 6 / 1 | 6510 | 464 |
| 6 / 3 | 8757 | 464 |
| 6 / 6 | 11046 | 464 |
| 10 / 1 | 7245 | 1232 |
| 10 / 3 | 8169 | 1232 |
| 10 / 6 | 15981 | 1232 |

Сделаем некоторый вывод: очевидно, что граф в виде матрицы получился не таким эффективным по памяти, так как требуется дополнительная таблица для нахождения кратчайшего пути (для удобного применения алгоритма поиска кратчайшего пути) и при этом каждая таблица является симметричной, что занимает лишнюю память. Однако к плюсам такого представления можно отнести удобство использования и относительно быстрый доступ к элементам, так как он происходит по индексу. Память зависит только от количества вершин.

На время же влияет не только количество вершин, но и ребер также. Причем при малых значениях это несильно отражается на времени, тогда как при больших разница видна больше. То есть чем больше граф связан, тем быстрее растет время поиска.

Алгоритм Дейкстры выбран оправданно, так как он выполняет поставленную задачу, является удобным при использовании на выбранном представлении графа (матрице стоимостей), при малых количествах не является затратным.

**Контрольные вопросы**

1. **Что такое граф?**

Граф – это конечное множество вершин и ребер, соединяющих их:

G = <V, E>, где V – конечное непустое множество вершин; Е – множество ребер (пар вершин)

1. **Как представляются графы в памяти?**

Графы в памяти могут представляться различными способами. Например, матрица стоимостей, матрица смежности, список смежности

1. **Какие операции возможны над графами?**

К основным операциями над графовыми структурами можно отнести:

* поиск кратчайшего пути от одной вершины к другой (если он есть);
* поиск кратчайшего пути от одной вершины ко всем другим;
* поиск кратчайших путей между всеми вершинами;
* поиск эйлерова пути (если он есть);
* поиск гамильтонова пути (если он есть).

1. **Какие способы обхода графов существуют?**

Существует два основных способа обхода графа: поиск в ширину (breadth first search, BFS), поиск в глубину (depth first search, DFS)

1. **Где используются графовые структуры?**

Графовые структуры используются в тех случаях, когда между элементами устанавливается некоторая связь. В качестве примера можно привести генеалогическое древо, блок-схема, схемы авиаперелетов, лабиринт и др.

1. **Какие пути в графе Вы знаете?**

Бывают пути: Эйлеров, Гамильтонов, кратчайший путь между всеми вершинами / от одной вершины к другой.

1. **Что такое каркасы графа?**

Каркас графа – дерево, в которое входят все вершины графа, и некоторые его ребра.