**Типы и структуры данных.**

**Лабораторная работа №8**

# Обработка графов

**Выполнил**: Кузнецов Александр

**Группа**: ИУ7-33

Цель работы: реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка связности, построение остовых деревьев минимальной стоимости.

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

**Задача 12**

Задана система двусторонних дорог. Найти множество городов, расстояние от которых до выделенного города ( столицы ) больше, чем Т.

**Входные данные:**

Номер выполнения команды;

Связка городов и их отдаление друг от друга

Минимальное допустимое расстояние между городами.

**Пример:**

//ввод команды для операций над графом

>>1

//ввод связки городов и их отдаление друг от друга

(From)>> 0

(To)>> 1

(Distance)>> 10

//ввод минимального допустимого расстояния между городами

(Min distance)>> 8

**Выходные данные**

Файл с программой на языке DOT

Максимальные пути из всех вершин до главной вершины

В случае, если таковых нет, то сообщение «ZERO»

**Функция программы**

Обработка графа, поиск максимального пути к заданной вершине, ограниченным условием о максимальной дистанции, если таковых не найдено, вывод сообщения «ZERO».

**Аварийные ситуации**

**-** Ввод цикла, т.е. от 0 до 0

**-** Ввод отрицательных значений

- Ограничение типом short int входных параметров дистанции

- Не существует путей от одной вершины к другой

- Не найдено дистанций к вершинам удовлетворяющих вводимым условиям о минимальной длине.

**Описание использованных структур данных**

***//узел графа***

typedef struct{

int num; //К какой вершине идем

int color; //цвет 0/1 показывает были ли мы в //данной вершине

int val; //Значение расстояния

}Node\_graph;

**Алгоритм**

**Обработка ввода:**

1. Вывести интерфейс программы
2. Проверить корректность ввода(Интерфейс)
3. Запустить выбранную команду

-//////- Вывод сообщение об ошибке, если она есть -//////-

Иначе

**Обработка операций над графом:**

1. Ввод графа с файла
2. Ввод графа с клавиатуры
3. Выход

**Если с клавиатуры:**

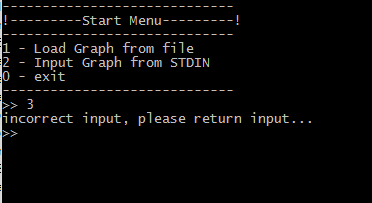
Запрос основных параметров: количество вершин, количество путей. Если был не правильный ввод, то есть вероятность исправить ошибку ввода.

**Выход:**

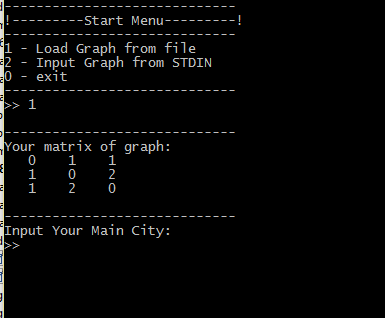
**Exit(0)**

**Тесты**

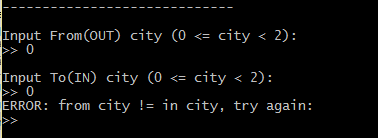
**Тест ввода:**Некорректный ввод команды графа:



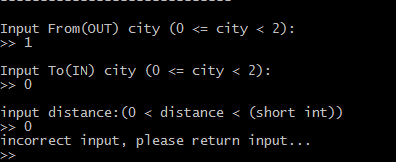
Корректный ввод номера операции над графом:



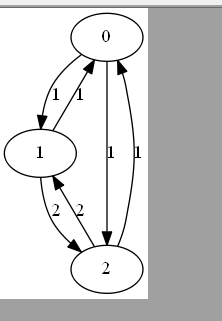
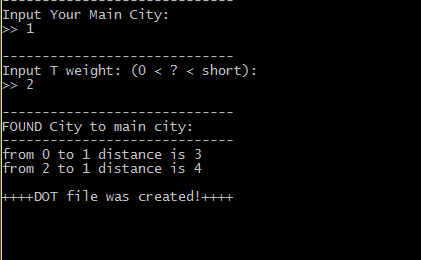
Тест на ввод цикла:



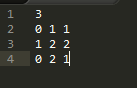
Тест на ввод длины ребра 0



**Результат:**



Исходный файл



Я выбрал алгоритм Дейкстры, т.к. алгоритм прост в реализации и подходит для этого типа задачи, т.к. выполняется поиск максимального пути и сравнение его с условием введенного с клавиатуры. Алгоритм Дейкстры в стандартном понимании – это поиск минимального пути, но можно его переделать и в алгоритм поиска максимального пути. Недостаток метода: граф не должен содержать циклов и отрицательных ребер.

### **Сложность алгоритма**

Сложность алгоритма Дейкстры зависит от способа нахождения вершины *v*, а также способа хранения множества непосещённых вершин и способа обновления меток. Обозначим через *n* количество вершин, а через *m* — количество рёбер в графе *G*.

В простейшем случае, когда для поиска вершины с минимальным *d*[*v*] просматривается всё множество вершин, а для хранения величин *d* используется массив, время работы алгоритма есть {\displaystyle O(n^{2})} O(n^2). Основной цикл выполняется порядка *n* раз, в каждом из них на нахождение минимума тратится порядка *n* операций. На циклы по соседям каждой посещаемой вершины тратится количество операций, пропорциональное количеству рёбер *m* (поскольку каждое ребро встречается в этих циклах ровно дважды и требует константное число операций). Таким образом, общее время работы алгоритма {\displaystyle O(n^{2}+m)}O(n^2 + m), но, так как {\displaystyle m\leq n(n-1)}m <= n(n-1), оно составляет {\displaystyle O(n^{2})}O(n^2).

Для разреженных графов (то есть таких, для которых m много меньше n²) непосещённые вершины можно хранить в [двоичной куче](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BA%D1%83%D1%87%D0%B0), а в качестве ключа использовать значения *d*[*i*], тогда время удаления вершины из {\displaystyle {\overline {U}}}U станет {\displaystyle \log n}Log(n) при том, что время модификации {\displaystyle d[i]}d[i] возрастёт до {\displaystyle \log n}Log(n). Так как цикл выполняется порядка *n* раз, а количество релаксаций (смен меток) не больше *m*, время работы такой реализации — {\displaystyle O(n\log n+m\log n)}O(nLog(n)+mLog(n)).

Если для хранения непосещённых вершин использовать [фибоначчиеву кучу](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8%D0%B5%D0%B2%D0%B0_%D0%BA%D1%83%D1%87%D0%B0), для которой удаление происходит в среднем за {\displaystyle O(\log n)}O(Log(n)), а уменьшение значения в среднем за {\displaystyle O(1)}O(1), то время работы алгоритма составит {\displaystyle O(n\log n+m)}O(nLog(n)+mLog(n)). Однако согласно лекциям Алексеева и Таланова

**Вопрос – Ответ**

1. **Что такое граф?**

Граф – это конечное множество вершин и ребер, соединяющих их, т. е.: 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 различным способом. Один из видов представления графов – это матрица смежности B(n\*n); В этой матрице элемент b[i,j]=1, если ребро, связывающее вершины Vi и Vj существует и b[i,j]=0, если ребра нет. У неориентированных графов матрица смежности всегда симметрична.

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

Перечислим основные операции по работе с графовыми структурами:

1. поиск кратчайшего пути от одной вершины к другой (если он есть);
2. поиск кратчайшего пути от одной вершины ко всем другим;
3. поиск кратчайших путей между всеми вершинами;
4. поиск эйлерова пути (если он есть);
5. поиск гамильтонова пути (если он есть).
6. **Какие способы обхода графов существуют?**

Один из основных методов проектирования графовых алгоритмов – это поиск (или обход графа) в глубину (depth first search, DFS), при котором начиная с произвольной вершины **v0**, ищется ближайшая смежная вершина **v**, для которой в свою очередь осуществляется поиск в глубину (т.е. снова ищется ближайшая, смежная с ней вершина) до тех пор, пока не встретится ранее просмотренная вершина, или не закончится список смежности вершины **v** (то есть вершина полностью обработана).

Указанного недостатка лишен другой метод обхода графа – поиск в ширину (breadth first search, BFS). Обработка вершины **v** осуществляется путем просмотра сразу всех новых соседей этой вершины. При этом полученный путь является кратчайшим путем из одной вершины в другую.

Поиск кратчайших путей до всех вершин из одной указанной вершины для взвешенного орграфа (имеющего значение, т.е. вес или стоимость, ребра) с неотрицательными ребрами осуществляется с использованием алгоритма Дейкстры. Алгоритм Дейкстры основан на выборе для включения в путь всякий раз той вершины, которая имеет наименьшую оценку кратчайшего пути (по весам ребер), то есть наименьший путь до этой вершины из всех возможных путей, которые были рассмотрены ранее

1. **Где используются графовые структуры?**

Гра́фовая база данных — разновидность баз данных с реализацией сетевой модели в виде графа и его обобщений. Графовая СУБД — система управления графовыми базами данных.

Например, типичное применение остовых деревьев минимальной стоимости – это построение коммуникационных линий между городами, где стоимости ребер – это стоимость коммуникационных сетей.

1. **Какие пути в графе Вы знаете?**

**Путь** в графе — последовательность вершин, в которой каждая вершина соединена со следующей ребром.

Путь, для которого никакие рёбра графа не соединяют две вершины пути, называется индуцированным путём.

Простая цепь, содержащая все вершины графа без повторений, известна как Гамильтонов путь.

Простой цикл, содержащий все вершины графа без повторений, известен как Гамильтонов цикл.

Цикл, получаемый добавлением ребра графа к остовному дереву исходного графа, известен как Фундаментальный цикл.

**7. Что такое каркасы графа?**

***Оптимальным каркасом*** *взвешенного графа* называется каркас, минимизирующий некоторую функцию от весов входящих в него ребер. Чаще всего в качестве такой функции выступает сумма весов ребер, реже — произведение. *Оптимальный каркас* еще называют кратчайшей *связывающей сетью* для данного графа.