|  |  |
| --- | --- |
|  | **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 №7 по предмету

“Типы и структуры данных”

Графы.

Выполнила:

Кузнецова Анастасия

Группа:

ИУ7-31Б

Вариант 13 (по списку 13)

Проверил(а):

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

**Описание условия задачи**

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

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.

**2. Описание ТЗ:**

Описание исходных данных и результатов

*Входные данные:*

Программа представляет собой меню, пользователь должен ввести свой выбор (целое число от 0 до 2)

Во входном файле хранятся вершины, дороги между ними, их длина и пошлины в виде: “V1->V2 duty, length”.

Ограничения для входных данных:

* Количество вершин не более 100
* Дорога не может идти в ту же вершину, из которой исходит
* Пошлина > 0
* Длина > 0
* Пошлина целое число
* Длина целое число
* Любая запись, не соответствующая “V1->V2 duty, length” некорректна

*Выходные данные:*

Программа выводит меню и в зависимости от выбора пользователя выводит:

1. Вывод графа
2. Вывод кратчайшего пути от A до B

Описание задачи, реализуемой программой

Программа выводит меню и выполняет задачу, запрошенную пользователем. В программе реализовано меню с выборами действий:

1. Выход
2. Вывод графа
3. Вывод кратчайшего пути от A до B

Способ обращения к программе

Программу следует вызывать из командной строки с помощью команды:

./app.exe input.txt output.gv

Описание возможных аварийных ситуаций и ошибок пользователей

* Количество вершин более 100
* Дорога идёт в ту же вершину, из которой исходит
* Пошлина <= 0
* Длина дороги <= 0
* Пошлина буква
* Длина дороги буква
* Любая запись, не соответствующая “V1->V2 duty, length”
* Входной файл не существует
* Входной файл пустой
* Количество аргументов командной строки не равно 3
* Выбор в меню не целое число, а буква

**3. Описание внутренних СД**

Матрица смежности:

#define N 100

typedef struct

{

int duty;

int length;

} weight\_t;

weight\_t matrix[N][N] = { 0 };

Массив вершин графа, содержащих название вершины и её текущий вес:

#define N 100

typedef struct

{

char name;

int weight;

} vertex\_t;

vertex\_t vertex[N] = { 0 };

**4. Описание алгоритма**

1. Чтение графа из файла в матрицу смежности, заполнение массива вершин
2. Печать меню
3. В зависимости от выбора пользователя:
   1. Выход
   2. Вывод графа
      1. Поиск кратчайшего пути с помощью Алгоритма Дейкстры
         1. Проход по матрице и запись для каждой вершины минимального для неё веса
         2. Запись пути до вершины B
      2. Вывод кратчайшего пути от А до В и его значения

**5. Набор тестов**

Позитивные тесты:

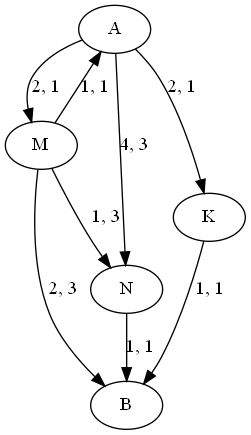
1. Вывод графа

Input:

Enter a choice:

1

Output:



1. Вывод кратчайшего пути от A до B, кратчайший путь один

Input:

Enter a choice:

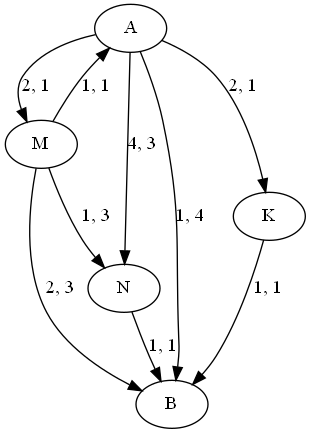
2

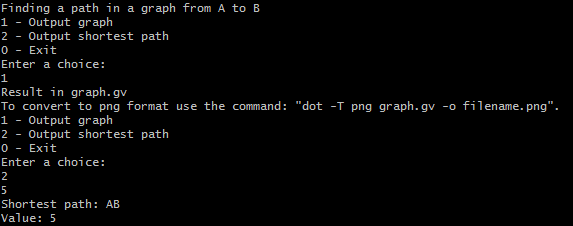
Output:

Shortest path: AKB

Value: 5

1. Несколько одинаковых кратчайших путей от A до B





Негативные тесты:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Input | Output |
| 1 | Количество аргументов командной строки не равно 3 | Вызов программы в командной строке:  ./app.exe input.txt | Key error |
| 2 | Входной файл не существует | Вызов программы в командной строке:  ./app.exe sovhsdib.txt | File error. |
| 3 | Входной файл пустой | Вызов программы в командной строке:  ./app.exe empty.txt graph.gv | File error. |
| 4 | В файле одна из записей некорректна | Во входном файле одна из записей:  A-> 2, 1 | Input error |
| 5 | Вместо длины дороги буква | Во входном файле одна из записей:  M->A a, 1 | Input error |
| 6 | Вместо пошлины буква | Во входном файле одна из записей:  M->A 1, a | Input error. |
| 7 | Длина дороги <= 0 | Во входном файле одна из записей:  M->A -1, 1 | Input error |
| 8 | Пошлина <= 0 | Во входном файле одна из записей:  M->A 1, -1 | Input error |
| 9 | Дорога идёт в ту же вершину, из которой исходит | Во входном файле одна из записей:  A->A 1, 1 | Input error |
| 10 | Количество вершин больше 100 | Во входном файле:  A->M 2,1  A->A 2,1  M->A 1,1  A->N 4,3  N->B 1,1  M->B 2,3  M->N 1,3  A->K 2,1  K->B 1,1  … | Input error |

**5. Выводы по проделанной работе**

В результате выполненной лабораторной работы я приобрела навыки работы с графами, узнала об алгоритмах нахождения кратчайшего пути в графах от одной вершины к другой.

В своей задаче я использовала матрицу смежности для представления графа в памяти, так как они обеспечивают наиболее быстрый доступ к элементу. Данный способ эффективен, когда между городами – вершинами много дорог.

Я использовала для нахождения кратчайшего пути алгоритм Дейкстры, так как он обеспечивает для моего варианта, где вес каждого ребра положительный по определению для моей задачи, наиболее эффективный результат.

Практический пример использования моей задачи:

Если семья должна добраться из одного города в другой, ей необходимо узнать наиболее дешевую и быструю дорогу.

Можно делать вывод, что хеш-таблицы являются наиболее эффективной структурой данных для поиска необходимого элемента.

**6. Ответы на вопросы**

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?

Граф можно представить как матрицу смежности или как список смежности.

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

Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной. Каждый элемент (ZAP[u]) списка смежностей является записью, содержащей данную вершину и указатель на следующую запись в списке (для последней записи в списке этот указатель – пустой). Входы в списки смежностей для каждой вершины графа хранятся в таблице (массиве) (BEG [u]).

1. Какие операции возможны над графами?

Основные операции по работе с графовыми структурами:

• 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);

- Добавление и удаление вершины

- Добавление и удаление ребра

1. Какие способы обхода графов существуют?

Графы можно обойти, используя поиск в глубину ли поиск в ширину.

Поиск в глубину (depth first search, DFS): 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). Если нет новых вершин, смежных с v, то вершина v считается использованной, идет возврат в вершину, из которой попали в вершину v, и процесс продолжается до тех пор, пока не получим v = v0. Иными словами, поиск в глубину из вершины v основан на поиске в глубину из всех новых вершин, смежных с вершиной v.

Поиск в ширину (breadth first search, BFS). Обработка вершины v осуществляется путем просмотра сразу всех новых соседей этой вершины.

1. Где используются графовые структуры?

Графовые структуры используются в программировании для решения различных социальных задач, задач информатики, задач биоинформатики и тд.

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?

* Эйлеровый путь - произвольный путь в графе, проходящий через каждое ребро графа точно один раз. При этом, если по некоторым вершинам путь проходит неоднократно, то он является непростым.
* Гамильтонов путь - путь в графе, проходящий в точности один раз через каждую вершину графа (а не каждое ребро).
* Простой путь
* Непростой путь

7. Что такое каркасы графа?

Каркас графа (или остовное дерево) – это подграф, который содержит все вершины графа и является деревом.