

Лабораторная работа №8

по дисциплине «Типы и структуры данных»

Тема: Графы.

Горохова Ирина ИУ7-31

Условие задачи:

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.

Задана система двусторонних дорог. Для каждой пары городов найти длину кратчайшего пути между ними.

Исходные данные:

Граф задается матрицей стоимостей.

Матрица стоимости графа $G=(V,E)$ – квадратная матрица размерности $|V|$:

$$w_{ij} = \begin{cases} \text{вес ребра, если вершины } i \text{ и } j \text{ смежные} \\ \infty, \text{ если вершины } i \text{ и } j \text{ не смежные} \end{cases}, \quad i = \overline{1, |V|}, j = \overline{1, |V|}.$$

Данная матрица получается программой из файла: первое число в файле - размерность матрицы, далее - сама матрица, такая, что элемент $[i][j]$ = весу ребра, если из i в j есть дорога, $[i][j] = 0$, если дороги нет.

Либо граф можно задать вручную: пользователя просят ввести вершину i , вершину j и вес ребра $i-j$ через пробел.

Введите через пробел номера двух вершин и расстояние между ними: 1 2 5
=> Дорога между 1 и 2 городами равна 5. (Вес ребра 1-2 равен 5)

Выходные данные:

Выходными данными программы являются:

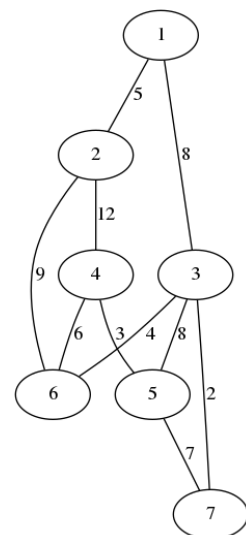
1. Изображение графа в формате .png, заданного матрицей стоимостей, с указанием весов ребер (рис).
2. Таблица кратчайших расстояний между каждым из двух городов. Таблица представлена в формате:

Город 1	Город 2	Кратчайшее расстояние
1	2	5
1	3	8

3. Матрица кратчайших расстояний, полученная вследствие работы алгоритма Флойда - Уоршелла.

∞	5	8	17	16	12	10
5	∞	13	12	15	9	15
8	13	∞	10	8	4	2
17	12	10	∞	3	6	10
16	15	8	3	∞	9	7
12	9	4	6	9	∞	6
10	15	2	10	7	6	∞

0	5	8	0	0	0	0
5	0	0	12	0	9	0
8	0	0	0	8	4	2
0	12	0	0	3	6	0
0	0	8	3	0	0	7
0	9	4	6	0	0	0
0	0	2	0	7	0	0



Взаимодействие с программой:

Работа с программой осуществляется с помощью меню программы:

Меню работы с графами:

- (1) Загрузка графа из файла
- (2) Задание графа вручную
- (3) Печать графа
- (4) Длина кратчайшего пути между каждой парой городов
- (5) Длины кратчайшего пути из одного города до других
- (6) Вывод на экран матрицы кратчайших расстояний
- (7) Проверка связности графа
- (0) Завершение работы

- (1) - загружает матрицу стоимостей из файла
- (2) - запрашивает ввод количества вершин, а затем связей этих вершин в формате первая_вершина вторая_вершина вес_ребра(расстояние). создает матрицу стоимостей по данной информации, полученной от пользователя.
- (3) - генерирует файл на языке DOT, запускает bash-скрипт, который преобразует файл с расширением .gv в изображение .png с графом.
- (4) - выводит таблицу с длинами кратчайших путей между любыми двумя городами).
- (5) - запрашивает ввод вершины v_0 , от которой требуется найти кратчайшие расстояния до всех других вершин. Выводит кратчайшее расстояние от v_0 до других вершин либо сообщение о том, что из v_0 нельзя попасть в какую-то из вершин.
- (6) - выводит на экран матрицу кратчайших расстояний, где элемент $[i][j]$ - расстояние между i -тым и j -тым городом.
- (7) - проверяет связный ли граф. Выводит сообщение с результатом.

Представление графа в программе:

Граф представлен матрицей стоимостей ($n \times n$) $matrix[i][j]$, где n - количество вершин.

В этой матрице элемент $matrix[i][j]$ = вес ребра $i - j$, если ребро, связывающее вершины V_i и V_j существует и $matrix[i][j] = 0$, если ребра нет.

У неориентированных графов матрица смежности всегда симметрична.

Используемые алгоритмы:

Для поиска кратчайших расстояний между любыми двумя городами используется **алгоритм Флойда - Уоршела** - алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа.

Алгоритм:

W - матрица стоимостей.

После работы алгоритма W - матрица кратчайших расстояний.

```

for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      W[i][j] = min(W[i][j], W[i][k] + W[k][j])

```

Алгоритм Флойда — Уоршелла является эффективным для расчёта всех кратчайших путей в плотных графах, когда имеет место большое количество пар рёбер между парами вершин. Алгоритм имеет кубическую сложность $O(n^3)$

Для поиска кратчайших расстояний из одного города до всех остальных используется **алгоритм Дейкстры**. Так как расстояние между городами не может быть представлено отрицательными числами, значит эффективнее использовать алгоритм Дейкстры, нет нужды в алгоритме Беллмана - Форда, который уступает по времени.

Алгоритм:

already_used - массив посещенных вершин
 min_rasst - массив минимальных расстояний
 do {

```

    min_ind = INF;
    min = INF;
    for (int i = 0; i < count; i++) {
        if (already_used[i] == 1 && min_rasst[i] < min) {
            min = min_rasst[i];
            min_ind = i;
        }
    }
    if (min_ind != INF) {
        for (int i = 0; i < count; i++) {
            if (matrix[min_ind][i] > 0) {
                tmp = min + matrix[min_ind][i];
                if (tmp < min_rasst[i])
                    min_rasst[i] = tmp;
            }
        }
        already_used[min_ind] = 0;
    }
}
while (min_ind < INF);

```

Для построения минимального остова используется **алгоритм Прима** - алгоритм построения минимального остовного дерева взвешенного связного неориентированного графа.

Алгоритм:

На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берется произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих ребер выбирается ребро

наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Таким образом, при выполнении каждого шага алгоритма, высота формируемого дерева увеличивается на 1. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

Вопросы к лабораторной работе:

1. *Что такое граф?*

Граф – конечное множество вершин и соединяющих их рёбер; $G = \langle V, E \rangle$. Если пары E (ребра) имеют направление, то граф называется направленным; если ребро имеет вес, то граф называется взвешенным.

2. *Как представляются графы в памяти?*

Существуют различные методы представления графов в программе.

Матрица смежности $B(n \times n)$ – элемент $b[i, j] = 1$, если существует ребро, связывающее вершины i и j , и $= 0$, если ребра не существует.

Список смежностей – содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с ней. Входы в списки смежностей могут храниться в отдельной таблице, либо же каждая вершина может хранить свой список смежностей.

3. *Какие операции возможны над графами?*

Основные операции над графами: обход вершин и поиск различных путей:

кратчайшего пути от вершины к вершине; кратчайшего пути от вершины ко всем остальным; кратчайших путей от каждой вершины к каждой; поиск эйлера пути и гамильтонова пути, если таковые есть в графе.

4. *Какие способы обхода графов существуют?*

Один из основных методов проектирования графовых алгоритмов – поиск в глубину.

Начиная с некоторой вершины v_0 , ищется ближайшая смежная ей вершина v , для которой в свою очередь осуществляется поиск в глубину 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 $v = v_0$. При просмотре используется стек.

Поиск в ширину – обработка вершины V осуществляется путем просмотра сразу всех «новых» соседей этой вершины, которые последовательно заносятся в очередь просмотра. Для поиска кратчайших путей используются алгоритмы Дейкстры, Беллмана-Форда, Флойда-Уоршелла.

5. *Где используются графовые структуры?*

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

Наиболее распространенным является использование графов при решении различных задач о путях, будь то построение коммуникационных линий между городами или прокладка маршрута на игровом поле.

6. *Какие пути в графе Вы знаете?*

Путь в графе, проходящий через каждое ребро ровно один раз, называется эйлеровым путём; путь может проходить по некоторым вершинам несколько раз – в этом случае он является непростым.

Путь, проходящий через каждую вершину ровно один раз, называется гамильтоновым путем.

Как эйлеров, так и гамильтонов путь могут не существовать в некоторых графах.

7. *Что такое каркасы графа?*

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (не обязательно все) его рёбра.

Для построения каркасов графа используются алгоритмы Крускала и Прима.