МГТУ им. Н. Э. Баумана

ИУ 7 – 32

Отчет о лабораторной работе №9

# «Графы»

Вариант №8

Исаев Д.С.

# Цель работы

Реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка 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

**Описание**

**Граф**

Граф – это конечное множество вершин и ребер, соединяющих их

G = < V,E >,

где V – конечное непустое множество вершин; Е – множество ребер (пар вершин).

Если пары Е (ребра) имеют направление, то граф называется ориентированным (орграф), если иначе - неориентированный (неорграф). Если в пары Е входят только различные вершины, то в графе нет петель. Если ребро графа имеет вес, то граф называется взвешенным. Степень вершины графа равна числу ребер, входящих и выходящих из нее (инцидентных ей). Неорграф называется связным, если существует путь из каждой вершины в любую другую.

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).

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). 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**.

Путь, полученный методом поиска в глубину, в общем случае не является кратчайшим путем из вершины **v** в вершину **u**. Это общий недостаток поиска в глубину [1,2]. На рис. 4 показан путь, полученный обходом графа в глубину.

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 [1].

На рис. показан путь, найденных методом поиска в ширину

Произвольный путь в графе, проходящий через каждое ребро графа точно один раз, называется эйлеровым путем. При этом, если по некоторым вершинам путь проходит неоднократно, то он является непростым. Если путь замкнут, то имеем эйлеров цикл. Для существования эйлерова пути в связном графе необходимо и достаточно, чтобы граф содержал не более двух вершин нечетной степени [2,3]

Путь в графе, проходящий в точности один раз через каждую вершину графа (а не каждое ребро) и соответствующий цикл называются гамильтоновыми и существуют не для каждого графа, как и эйлеров путь. В отличие от эйлеровых путей неизвестно ни одного простого необходимого и достаточного условия для существования гамильтоновых путей. Неизвестен даже алгоритм полиномиальной сложности, проверяющий существование гамильтонова пути в произвольном графе. Проблема существования гамильтонова пути принадлежит к классу так называемых NP-полных задач.

Каркас графа – такой его подграф, который сожержит все вершины графа и является деревом.

Графы используются достаточно обширно, например, для визуализации зависимостей между данными, или, для навигации (карты, навигаторы).

**Реализация**

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.

Во многих случаях удобнее представлять граф в виде так называемого списка смежностей. 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]).

Граф был описан так:

    public class MyGraph : BidirectionalGraph<PocVertex, PocEdge>

    {

        public int[,] Lengths;

        public int[,] Taxes;

        private int VertexCount;

…

}

Для хранения графа используются 2 матрицы меток.

**Результаты**

Для реализации поиска кратчайшего пути был выбран алгоритм Дейкстра, так он быстро работает, неотрицательные метки ребер позволяют его использовать и он достаточно прост в реализации. Алгоритм отлично работает и наглядно визуализируется в данной программе в соответствии с ТЗ.

**Вывод:**

Алгоритм Дейкстра отлично подходит для поиска кратчайшего пути, если метки ребер неотрицательные.