*Национальный технический университет Украины «КПИ»*

*Факультет информатики и вычислительной техники*

*Кафедра вычислительной техники*

***Лабораторная работа №8***

***По курсу Дискретной математики***

***Вариант №4***

*Выполнила студентка*

*1 курса ФИВТ гр. ИВ-92*

*ГлуШко Ольга*

*Проверил Флеров А.И.*

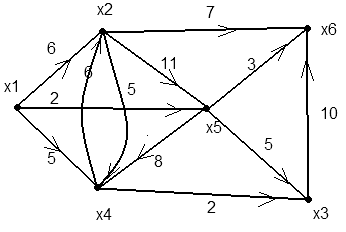
Киев 2010

**Вариант №4**

**Цель работы**: определить кратчайший путь на орграфе на основе алгоритма Форда Л.Р.,Беллмана Р.Е. между начальной вершиной 1 и конечной. Сформировать орграф D1, взяв за основу граф G2, матрицу весовых коєффициентов. Разработать алгоритм и программу выполнения задания в соответствии с вариантом. Результаты представить графически или таблично.

**Выполнение работы**

Изобразим орграф, заданный вариантом.



*Алгоритм Беллмана — Форда* — алгоритм поиска кратчайшего пути во взвешенном графе. За время

O(|V| × |E|) алгоритм находит кратчайшие пути от одной вершины графа до всех остальных. В отличие от алгоритма Дейкстры, алгоритм Беллмана — Форда допускает рёбра с отрицательным весом.

Так как в данном примере вес каждого из ребер – число положительное, то и сумма масс будет положительной.

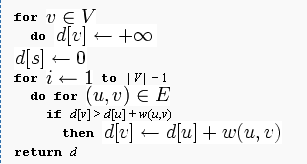
*Решение задачи на графе без отрицательных циклов*

Для нахождения кратчайших путей от одной вершины до всех остальных, воспользуемся методом динамического программирования. Построим матрицу Aij, элементы которой будут обозначать следующее: Aij — это длина кратчайшего пути из s в i, содержащего ровно j рёбер.

Путь, содержащий 0 рёбер, существует только до вершины s. Таким образом, Ai0 равно 0 при i = s, и  в противном случае.

Теперь рассмотрим все пути из s в i, содержащие ровно j рёбер. Каждый такой путь есть путь из j − 1 ребра, к которому добавлено последнее ребро. Если про пути длины j − 1 все данные уже подсчитаны, то определить j-й столбец матрицы не составляет труда.

Так выглядит **алгоритм** поиска длин кратчайших путей в графе без отрицательных циклов:



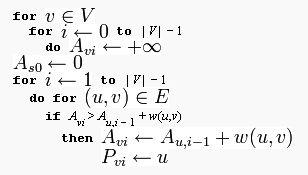
Здесь V — множество вершин графа G, E — множество его рёбер, а w — весовая функция, заданная на ребрах графа.

Внешний цикл выполняется |V| - 1 раз, поскольку кратчайший путь не может содержать большее число ребер, иначе он будет содержать цикл, который точно можно выкинуть.

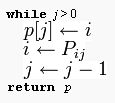
Вместо массива d можно хранить всю матрицу A, но это требует O(V²) памяти. Зато при этом можно вычислить и сами кратчайшие пути, а не только их длины. Для этого заведем матрицу P[i,j].

Если элемент A[i,j] содержит длину кратчайшего пути из s в i, содержащего j рёбер, то P[i,j] содержит предыдущую вершину до i в одном из таких кратчайших путей (ведь их может быть несколько).

Теперь алгоритм Беллмана — Форда выглядит так:



После выполнения этого алгоритма элементы A[i,j] содержат длины кратчайших путей от s до i с количеством ребер j, и из всех таких путей следует выбрать самый короткий. А сам кратчайший путь до вершины i с j ребрами восстанавливается так:



**Листинг программы**

program Ford\_Bellman1;

const

FIN = 'fordbell.in';

FOUT = 'fordbell.out';

NMAX = 200;

INF = high(integer) div 2 - 1;

type

PEdge = ^TEdge;

TEdge = record

u, v, w: integer;

next: PEdge;

end;

procedure addEdge(var l: PEdge; const u, v, w: integer);

var t: PEdge;

begin

new(t);

t.next := l;

t.u := u;

t.v := v;

t.w := w;

l := t;

end;

var

n: integer;

m: integer;

e: PEdge;

procedure Ford\_Bellman(const s, t: integer);

var i, k: integer;

cur: PEdge;

dist: array[1..NMAX] of integer;

prev: array[1..NMAX] of integer;

begin

for i := 1 to n do begin

dist[i] := INF;

prev[i] := -1;

end;

prev[s] := 0;

dist[s] := 0;

for k := 1 to n - 1 do begin

cur := e;

for i := 1 to m do begin

if dist[cur.v] > dist[cur.u] + cur.w then begin

dist[cur.v] := dist[cur.u] + cur.w;

prev[cur.v] := cur.u;

end;

cur := cur.next;

end;

end;

for i := 1 to n - 1 do write(dist[i], ' ');

writeln(dist[n]);

end;

var i, u, v, w: integer;

begin

AssignFile(input, FIN); reset(input);

AssignFile(output, FOUT); rewrite(output);

read(n, m);

for i := 1 to m do begin

read(u, v, w);

addEdge(e, u, v, w);

end;

Ford\_Bellman(1, 0);

close(input);

close(output);

end.

Используя данную программу мы можем не только вычислить длину и стоимость пути из первой в последнюю точку, но и между двумя другими произвольными точками. Приведем примеры результатов(содержатся в файле вывода):

|  |  |  |  |
| --- | --- | --- | --- |
| Начальная вершина | Конечная вершина | Длина пути | Стоимость |
| 1 | 6 | 2 | 5 |
| 2 | 4 | 1 | 5 |
| 4 | 2 | 1 | 6 |

**Вывод:** с помощью данной лабораторной работы была изучена широко распространенная задача нахождения минимального пути и стоимости. Решение проводится по алгоритму Беллмана — Форда — алгоритму поиска кратчайшего пути во взвешенном графе. На основе данного алгоритма разработана программа, выполняющая необходимые вычисления и сохраняющая результат в отдельный файл вывода.