***Алгоритм Уоршелла нахождения путевой матрицы***

Если нас интересует, достижима ли вершина vj, из вершины vi, и не интересует число путей из вершины vi в vj, то достаточно в матрице M все ненулевые элементы поменять на 1. Тогда получим матрицу достижимости, или путевую матрицу Р элементы которой Pij = 1, если существует путь из vi, в vj, и Pij = 0 в противном случае. Матрицу P называют также транзитивным замыканием матрицы смежности.

Путевая матрица только показывает, имеется или отсутствует путь между парой вершин (или цикл в любой вершине), но не определяет все пути. Рассмотренный способ определения путевой матрицы P из матрицы M громоздок. Более рациональный способ получения путевой матрицы предложен Уоршаллом, в котором используются реализуемые просто булевы операции. В этом алгоритме вначале путевая матрица P устанавливается равной матрице смежности, затем за k проходов, k=1,2,...,n, в циклах по ni и nj выполняются булевы операции:   
M[i][j] || (M[i][k] && M[k][j]).

Алгоритм выполняется за k проходов. После k-го прохода путевая матрица содержит 1 или 0 в зависимости от того, имеются ли пути между вершинами i и j, которые не проходят через вершины с номерами, большими или равными к, т.е. между вершинами i и j могут быть вершины только с номерами, меньшими или равными к. Это представлено на рис. ниже.

|  |  |  |
| --- | --- | --- |
| **Граф** | **Матрица смежности** | **Путевая матрица графа** |
| undefined | 0, 1, 1, 0  0, 0, 0, 0  0, 1, 0, 1  0, 0, 1, 0 | Проход k=0:  0, 1, 1, 0  0, 0, 0, 0  0, 1, 0, 1  0, 0, 1, 0  Проход k=1:  0, 1, 1, 0  0, 0, 0, 0  0, 1, 0, 1  0, 0, 1, 0  Проход k=2:  0, 1, 1, **1**  0, 0, 0, 0  0, 1, 0, 1  0, **1**, 1, **1**  Проход k=3:  0, 1, 1, 1  0, 0, 0, 0  0, 1, **1**, 1  0, 1, 1, 1 |

Таким образом, после k-го прохода путь между вершинами i и j есть тогда и только тогда, если:

*1) он уже был или   
2) есть пути между вершинами i и к и одновременно между вершинами к и j, проходящие через вершины с номерами, меньшими к.*

Алгоритм реализуется посредством трех вложенных циклов по к, i и j, имеет временную сложность О(n3). Ниже демонстрируется получение путевой матрицы из матрицы смежности.

for (int k=0; k<n; k++)

for (int j=0; j<n; j++)

for (int i=0; i<n; i++)

M[i][j]= M[i][j] || (M[i][k] && M[k][j]);

***Алгоритм Флойда-Уоршелла нахождения путевой матрицы***

Иногда требуется решать общую задачу нахождения кратчайших путей, т.е. нахождения для каждой пары вершин пути, это длина (расстояние) которого минимальна среди всех возможных путей.

Существует прямой способ решения этой задачи, использующий алгоритм Флойда-Уоршелла. Граф G=(V,E) с n вершинами представляется взвешенной матрицей смежности S(n\*n). В результате решения задачи формируется матрица А(n\*n) в которой элемент будет содержать значение кратчайшего пути от вершины vi до вершины vj.

Сначала исходная матрица S копируется в матрицу А, причем если дуга (ij) отсутствует, то элемент матрицы A равен \infty, а диагональные элементы равны нулю. Алгоритм выполняется за k проходов. После k-то прохода А содержит значение наименьшей длины путей из вершины i в j, которые не проходят через вершины с номером, большим k, т.е. между вершинами i и j могут быть вершины только с номерами, меньшими или равными k. При k-м проходе элемент матрицы A вычисляется как A[i][j] = min (A[i][j], A[i][k] + A[k][j]).

Алгоритм Флойда- Уоршелла реализуется посредством трех вложенных циклов и имеет сложность О(n3).

for (int k=0; k<n; ++k)

for (int i=0; i<n; ++i)

for (int j=0; j<n; ++j)

A[i][j] = min (A[i][j], A[i][k] + A[k][j]);

Часто необходимо знать не только кратчайшее расстояние между вершинами v и w, но и вершины, через которые пролегает кратчайший путь. Для этого дополнительно используется матрица М в которой элемент содержит номер вершины k, полученный при нахождении наименьшего значения в k-м проходе. Если элемент матрицы равен нулю, то вершины i и j смежные. Ниже приводится программа нахождения кратчайших расстояний по алгоритму Флойда.

for (int k=0; k<n; k++)

for (int i=0; i<n; i++)

for (int j=0; j<n; j++)

if (A[i][k] != INF && d[k][j] != INF && (A[i][k]+d[k][j]) < A[i][j])

{

A[i][j] = A[i][k]+A[k][j];

M[i][j] = M[k][j]; // вычисление матрицы предшествования

}

Восстановление путей. Пример рекурсивной функция печати кратчайшего пути между вершинами:

int Path(int \*\*P,int i,int j)

{

if(P[i][j]==i)

{

cout << i; //стартовая вершина

}

else

{

PathRec(P,i,P[i][j]);

cout << " -> " << P[i][j]; //промежуточные вершины

}

return i;

}

void printPath(int \*\*P,int n, int inf)

{

for(int i=0; i<n; i++)

for(int j=0; j<n; j++)

if(i!=j && P[i][j] != inf)

{

Path(P,i,j);

cout << " -> " << j << endl; // последняя вершина

}

}