Generate Path Matrix from the given Adjacency Matrix

Divija Palleti
IHM2016005[1]

Nidhish Kumar Reddy
ICM2016006[2]

Nanda Kishore
IWM2016002[3]

***Abstract*— The objective of this report is to describe how a Path Matrix can generated by the given Adjacency Matrix. This report also describes different approaches which can be used to construct a path matrix from the Adjacency Matrix.**

***Index Terms*— Path Matrix, Adjacency Matrix, Warshall's Algorithm, Edges, Vertices, Un-directed graph, Directed graph, Graph Representation, Transitive Closure, Depth first Search.**

I. INTRODUCTION

Graph is a collection of finite sets of vertices or nodes **(V)** and edges **(E)**. In order to describe a graph into any representation, relationship between edges and vertices should be preserved. There are variety of ways to represent a graph. One of them is Adjacency Matrix.

A path matrix P of a simple directed graph **(V,E)** with n vertices is a Boolean matrix where **P[i][j] =1** if there is a path in the graph, i.e. a finite contiguous sequence of edges, from the vertex i vertex j and **P[i][j] =0**, if there is no such path.

Transitive Closure is the reachability matrix to reach vertex u from vertex v of a graph. The path matrix is the matrix associated with the transitive closure of the adjacency relation in the vertex set **V** of the given digraph.

II. MOTIVATION

Graphs are one of the ancient and powerful tools that visually represent sets of information and therefore the relations among them. They have a really long history of serving to individuals communicate, perceive the important world, and solve scientific issues. Graph visualization helps users gain insight into data by turning the information components and their internal relationships into graphs. Graph visual image leverages the human visual system to support data discovery. it's been wide employed in several applications, like social networks and net communications. It is shown how to generate the path matrix or corresponding lists which arise in various applications involving graphs and their spanning trees.

III. ALGORITHM DESIGN

A. Background

A Adjacency matrix of a graph gives the Binary-matrix which indicate whether pairs of vertices are adjacent or not in the graph.

In an Adjacency matrix **Adj** of a digraph **G - (V, E)** :

- **Adj[i][j] == 1** if vertex **i** is adjacent to vertex **j** (i.e. there is an edge from i to j)
- **Adj[i][j] == 0** if vertex **i** is not adjacent to vertex **j** (i.e. there is no edge from i to j).
- **Adj2[i][j] == 1** if and only if we can get from vertex **i** to vertex **j** in two steps or less.
- Similarly we can obtain Adjacency matrix **Adj3**, **Adj4**, **Adj5**, and so on.
- So, **Adjn** matrix represents a path of any length from vertex i to vertex j where **n** is the number of vertices. This would also the graph's path matrix: **Adjn[i][j] == P[i][j]** for all i, j.

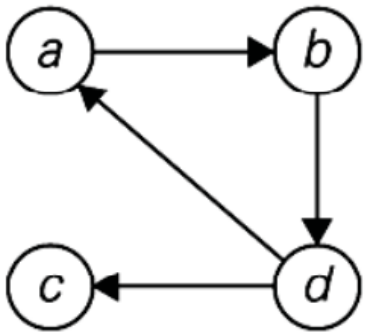

Fig. 1. Directed graph with 4 vertices

B. Methodology

We have approached this particular problem of graph using Warshall's Algorithmn. The main idea of this algorithm is that a path exists between two vertices i, j, if and only if :

- There is an edge from i to j
- There is a path from i to j going through vertex 1
- There is a path from i to j going through vertex 1 and/or 2
- There is a path from i to j going through any of the other vertices.

1) Method - 1: We make use of the function *path1* to extract the path matrix from the given matrix :

- Number of vertices will be the number of rows and columns of the given Adjacency array **Adj**.
- Recurrence relating elements Adj^k to elements of Adj^{k-1} is: $Adj^k[i][j]$ = $Adj^k[i][j]$ or [$Adj^k[i][k]$ and $Adj^k[k][j]$].

- So, on the k^{th} iteration, the algorithm determine if a path exists between two vertices i, j using just vertices among [1 - k] allowed as intermediate vertices.
- In this the new matrices can be written over their predecessors.

2) Method - 2: We can find all the vertices reachable for a vertex V by calling DFS on the vertex. So, the above idea is implemented to all the vertices of a given digraph(matrix) to obtain the Transitive closure of the graph. Thus, resulting in the path matrix.

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

Fig. 2. Adjacency Matrix of the graph in Fig.1

C. Algorithm

The Algorithm finds the path matrix P(Boolean Matrix) of the graph

Algorithm 1 Path_Matrix

```
for i=1 to (no_vert) do
    for j=1 to (no_vert) do
        if A[j][i]==0 then
            P[i][j] ← 0
        if matrix[j][i]==1 then
            P[i][j] ← 1
for k=1 to (no_vert) do
    for i=1 to (no_vert) do
        for j=1 to (no_vert) do
            P[i][j] ← P[i][j]or(P[i][k]andP[k][j])
```

Algorithm 2 Path_Matrix

```
procedure DFS(s, v, tc[100][100], A[100][100], n)
    tc[s][v] ← true
    for i=1 to (no_vert) do
        if A[v][i]==1  tc[s][i]==false then
            DFS(s, i, tc, A, n)
procedure TRANSITIVE(A[100][100], TC[100][100], n)
    for i=1 to (no_vert) do
        DFS(i, i, tc, A, n)
    for i=1 to (no_vert) do
        for j=1 to (no_vert) do
            printtc[i][j]
```

D. Output Format

- The output of the above algorithm is a n x n binary matrix(P) where n is the number of vertices V.
- If $P[i][j] = 1$ there exists path from i to j.
- If $P[i][j] = 0$ there is no path from i to j.

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	0	0	0	0
d	1	1	1	1

Fig. 3. Resultant path matrix

IV. ANALYSIS

A. Analysis

1) Method - 1: In the method 1 to find the path between 2 vertices we consider all the remaining vertices to be the intermediate vertices. So the time complexity will be directly proportional to, i.e $O(V^3)$.

$$time_{method-1} \propto O(V^3)$$

$$\implies O(V^3) \text{ - A Cubic Time Computation}$$

2) Method - 2: In here we find all vertices reachable from a vertex by applying DFS on a vertex thus resulting in the transitive closure of a graph. So the time complexity reduces to $O(V * (V + E))$. So the time complexity will be directly proportional to $O(V * (V + E))$.

$$time_{method-2} \propto O(V * (V + E))$$

But we know that $E \propto V^2$(worst case of a graph - dense graph).

$$\implies time_{method-2(worst)} \propto O(V * (V + V^2))$$

$$\implies O(V^3) \text{ - A Cubic Time Computation}$$

We also know that $E \propto V$(best case of a graph).

$$\implies time_{method-2(best)} \propto O(V * (V + V))$$

$$\implies O(V^2) \text{ - A Quadratic Time Computation}$$

3) Space Complexity: The present algorithm of $method-1method-2$ has a space complexity of $O(1)$. We don't need any additional space to store the matrices but we can just replace the previous matrix with the present matrix.

V. EXPERIMENTAL STUDY

The best way to study an algorithm is by graphs and profiling.

1) Graphs-Time-Complexity: We have seen that $t_{Method1} >= t_{Method2}$. So,the graphs for Method2 lies below Method1.We can clearly see that Method2 is of order $V(V+E)$ and Method1 is of order V^3 for connected graphs.

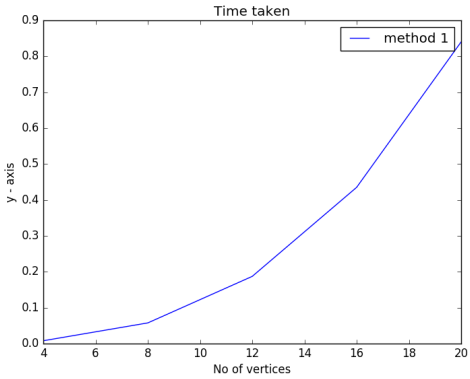

Fig. 4. Method 1 Complexity

2) Profiling:

$Table1 : NumberOfComputations$

V	$t_{method1}$	$t_{method2}$
4	0.008	0.0056
8	0.0576	0.0208
12	0.1872	0.0456
16	0.4352	0.0802
20	0.84	0.1202

- We can seen that $t_{Method1} >= t_{Method2}$
- We can also see that time increases as the value of n increases and $time_{Method-1} \propto V^3$, $time_{Method-2} \propto V(V+E)$.

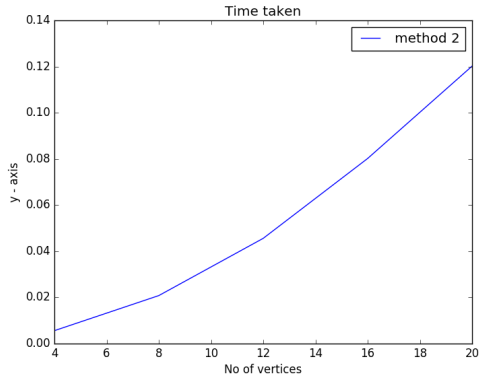

Fig. 5. Method 2 complexity.

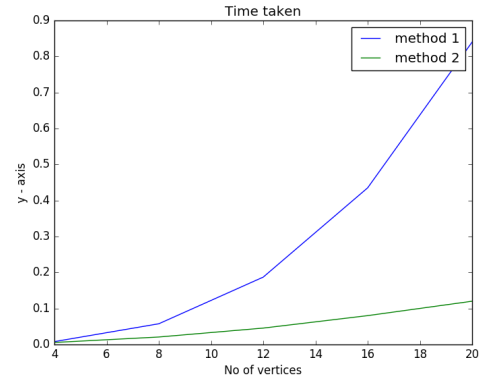

Fig. 6. Comparison between the Method 1 and 2.

VI. DISCUSSIONS

In is clearly shown that the Algorithms which uses Method2 is more efficient when compared to Method1.(i.e)Method2 has a time complexity of $O(V(V+E)$ (where V is the number of vertices and E is number of edges).

VII. CONCLUSIONS

- We can see that the Complexity of Path Matrix generation in above Method1 is $O(V^3)$ Where V is the number of rows in the given adjacency matrix.
- We can see that the Complexity of Path Matrix generation in above Method2 is $O(V(V+E))$ Where V is the number of vertices.

VIII. APPLICATIONS

Practical implementations of the presented algorithm and the information obtained through the present algorithm can be used in finding the shortest path and can also as intermediate information in generating the spanning tree. It can be used to gain knowledge about various molecules, their structure and properties.

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Introduction to Algorithms English By Thomas H. Cormen , By Charles E. Leiserson , Ronald L. Rivest ,Clifford Stein(3rd edition)

[3] $https : //www.techiedelight.com/transitive - closure - graph/$

[4] $https : //en.wikipedia.org/wiki/Adjacency\_matrix$