Transform a given adjacency matrix to an incidence matrix and vice-versa

Ankit Petkar IRM2014002, Surendra Pal Uikey IWM2014006, and Varun Kumar ICM2014008

Abstract—This document explains the transformation of Adjacency matrix to Incidence matrix and vice-versa. It begins by defining the properties of Adjacency matrix and Incidence matrix. Further we discuss on simple transformation technique to solve both problems. Complexities of the applied algorithms are defined.

Index Terms—

1 Motivation

Adjacency matrix and Incidence matrix are the matrices to represent graphs. Thus the transformation between two of them helps explore the characteristics of the graph.

2 Introduction

Graph (G) [1] (Figure 1) is 1a pictorial representation of some set of elements where some of themare connected through links. It is represented as G(V , E) , where V and E denotes set of vertices and edges respectively.

Incidence Matrix (I) (Figure 2) is rectangular matrix of n x m where n is the number of vertices and m is the number of edges, such that $I_{i,j} = 1$ if the vertex v_i and edge e_j are incident and 0 otherwise.

Adjacency Matrix (A) (Figure 3) is a square matrix of n x m where n and m is number of vertices and the elements represents whether the vertices in the graph are adjacent or not. Its a 0,1 matrix with 0 indicating no edge and 1 indicating edge present and also diagonal values are all zeroes.

3 Algorithms

3.1 Transformation from Adjacency Matrix To Incidence Matrix.

Input :- Given the Adjacency_Matrix with no self loops.

Algorithm :-

Step 1: Vertex = Adjacency_Matrix.vertex;

Step 2: Edges = Adjacency_Matrix.edges;

Step 3: Incidence_Matrix[Vertex][Edges];

Step 4: Initialize Incidence_Matrix[Vertex][Edges] to zeros so that there is no entry will be there.

Step 5: Make a counter which takes care of the Edges set it to the 1.

Step 6: Traverse through Adjacency_Matrix

Step 7:

Loop i from 1 to Vertex

Loop j from i+1 to Vertex

no. Of edges between the vertex i and j is stored in edge.

edge = Adjacency_Matrix[i][j];

loop until number of edges not equal to 0

Incidence_Matrix[i][counter] = Incidence_Matrix[i][counter] + 1;

Incidence_Matrix[j][counter] = Incidence_Matrix[j][counter] + 1;

edge = edge - 1;

End loop

End Loop

End Loop

Output :- Incidence_Matrix.

3.2 Transformation From Incidence Matrix To Adjacency Matrix.

Input :- given Incidence Matrix with no self loops.

Algorithm :-

Step 1: Vertex = Incidence_Matrix.vertex;

Step 2: Edges = Incidence_Matrix.edges;

Step 3: create adjacency matrix Adjacency_Matrix[Vertex][Vertex];

Step 4: initialize Adjacency_Matrix[Vertex][Vertex] with the zeros so that no entry will be there.

Step 5: Traverse through Incidence_Matrix

Step 6:

Loop i from 1 to Edges

first vertex involve in the ith edge is stored in first_One

And second vertex involve in the ith edge is stored second_One

Adjacency_Matrix[first_One][second_One] = Adjacency_Matrix[first_One[second_One] + 1;

Adjacency_Matrix[second_One][first_One] = Adjacency_Matrix[second_One][first_One] + 1;

End Loop

Output :- Adjacency_Matrix.

4 Implementation and Result

4.1 Algorithm 1 :- We have given the Adjacency Matrix, we need to construct the equivalent Incidence Matrix to it.

First we have to find the vertex and edges from the given Adjacency Matrix.

vertex can be determine from the order of the matrix and edges will the sum of all the non-zero entries in the adjacency matrix.

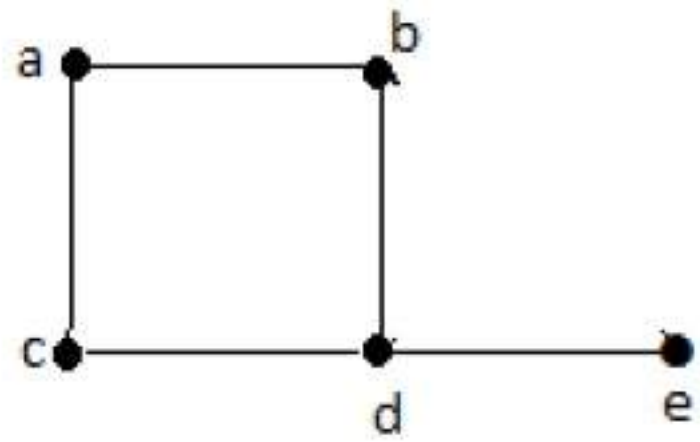

Fig. 1. **Graph(Vertex,Edge) - Vertex = {a, b, c, d, e}, Edges = {ab, ac, bd, cd, de} Source (https://www.tutorialspoint.com/graph_theory/graph_theory_introduction.htm)**

	ab	bc	bd	cd	de
a	1	1	0	0	0
b	1	0	1	0	0
c	1	0	0	1	0
d	0	0	1	1	1
e	0	0	0	0	1

Fig. 2. **Incidence Matrix**

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

Fig. 3. **Adjacency Matrix**

Now create the matrix which is order of vertex and edges. And initialise with the zeros. Which show no entries is there.

Create a counter which point the current edges that is being processed. Initially set this counter to the first edges.

Traverse through every entries of the Adjacency Matrix and take the non-zero entries from it and make entries in the column of the incidence matrix corresponds to the counter and marks the 1's entries in those vertex rows through which the edges has gone. Keep traversing until the whole matrix is traversed. After processing whole Adjacency matrix we get the incidence matrix.

Complexity Of this This Algorithm :

N = Number of the Vertex.

1. Best Case - O(N*N).
2. Worst Case - O(N*N).

4.2 Algorithm 2 :- We have given the Incidence Matrix, we need to construct the equivalent Adjacency Matrix to it.

Vertex for the Adjacency Matrix we can get from the order of the Incidence Matrix.

Create the Adjacency Matrix which is order of Vertex and Vertex. And Initialise with the zeros which shows initially there was no entries.

Now traverse through the Incidence Matrix.

For each column of the Incidence Matrix we get the two vertex that part in the edges that is corresponds to the column in the Incidence Matrix. Update the Adjacency Matrix for the above the two vertex. Keep traversing until we traverse through whole matrix. After processing whole Incidence matrix we get the Adjacency Matrix.

Complexity Of this Algorithm :

N = Number of Vertex.

M = Number of Edges.

1. Best Case - O(N*M).
2. Worst Case - O(N*M).

5 Conclusion

Thus applying the above algorithms we get the desired transformed Adjacency and Incidence matrices. Complexities of both the algorithms are thus calculated.

Analysing the time taken vs the number of vertices for the transformation of Incidence to Adjacency matrix can be viewed in Figure 4. And the time taken vs the number of vertices for the transformation of Adjacency to Incidence matrix can be viewed in Figure 5. Thus we conclude that as the number of vertices increasees the execution time increases in both the cases.

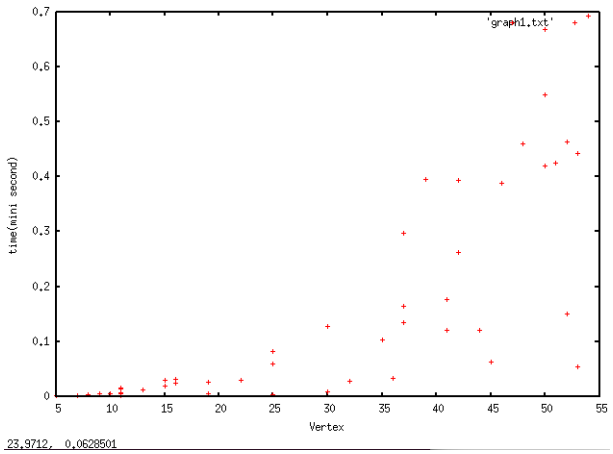

Figure 4 Analysis of time taken vs number of vertices for transformation of Incidence to Adjacency matrix.

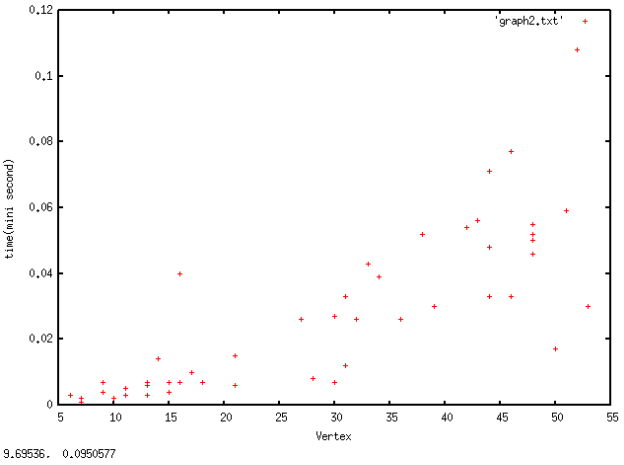

Figure 5 Analysis of time taken vs number of vertices for transformation Adjacency matrix to Incidence matrix.

References

[1] N. Deo, *Graph theory with applications to engineering and computer science.*