Demonstration of Intersection Operation on two Graphs

Anshul Agarwal *(IWM2017008)*, Vardhan Malik *(IWM2017007)*, Aman Gupta *(IWM2017006)*

***Abstract*—This paper demonstrates the Intersection operation on two independent graphs being provided in two formats i.e. Adjacency Matrix and Incidence Matrix. The operation is valid on all kind of graphs, which are Empty, Disconnected, Connected and original graphs.**

I. KEYWORDS

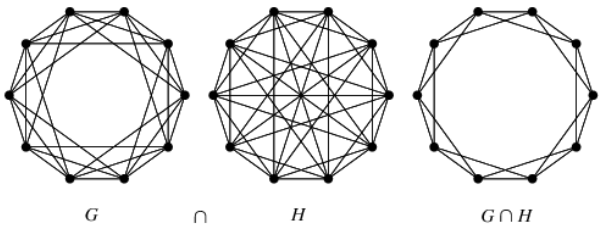
Matrix, Incidence Matrix, Adjacency Matrix, Intersection, Connected and Disconnect Graphs.

II. INTRODUCTION

IN Mathematics or Set theory, Intersection of two sets A and B, denoted by A B,is the set of all objects that are members of both the sets A and B.In symbols, $A \cap B = \{x : x \in A \text{ and } x \in B\}$.

In case of graphs, the graph intersection can be defined as follows: Formally, an intersection graph G is an un-directed graph formed from a family of sets Si, i = 0, 1, 2, ... by creating one vertex vi for each set Si, and connecting two vertices vi and vj by an edge whenever the corresponding two sets have a nonempty intersection. Only those edges and nodes are considered which are present n both the graphs.

Graph can formally be represented in two formats i.e. Adjacency matrix and Incidence matrix. An adjacency matrix is a square matrix used to represent a finite graph. Its rows and columns are labeled by graph vertices, with a 1 or 0 in position (i,j) according to whether i and j are adjacent or not. An incidence matrix is the matrix where rows are the vertices and columns are the edges.A matrix element is 1 if that edge is assigned to that particular vertex or not.Below image describes the graph intersection operation.



III. ALGORITHM DESIGN AND EXPLANATION

Our approach to given problem for Adjacency Matrix input includes the following steps:

1. We are taking the number of vertices and number of edges as an input.

2. Then we are creating an adjacency matrix with the help of the input.

3. Same procedure repeats to input second adjacency matrix.

4. Take minimum of the row numbers of the two matrices. This will be the new number of rows for the intersection matrix.

5. Take minimum of the columns numbers of the two matrices. This will be the new number of columns for the intersection matrix.

6. Loop over the matrices and if both consist of 1 at same position then add the edge to graph 7. Plot graph by using the plot function.

The pseudo-code for this algorithm is:-

Algorithm 1 Input for 1st adjacency matrix

$InputNumberofvertices$	▷ t ← t+1
$InputNumberofedges$	▷ t ← t+1
while $i < edges$ **do**	▷ t ← t+1
$InputVertex(u)$	▷ t ← t+1
$InputVertex(v)$	▷ t ← t+1
$Adjacentmatrix1[u][v] = 1$	▷ t ← t+1
end while	
$Rows1 \leftarrow Numberofvertices$	▷ t ← t+1
$Columns1 \leftarrow Numberofvertices$	▷ t ← t+1

Algorithm 2 Input for 2nd adjacency matrix

$InputNumberofvertices$	▷ t ← t+1
$InputNumberofedges$	▷ t ← t+1
while $i < edges$ **do**	▷ t ← t+1
$InputVertex(u)$	▷ t ← t+1
$InputVertex(v)$	▷ t ← t+1
$Adjacentmatrix2[u][v] = 1$	▷ t ← t+1
end while	
$Rows2 \leftarrow Numberofvertices$	▷ t ← t+1
$Columns2 \leftarrow Numberofvertices$	▷ t ← t+1

Algorithm 3 Intersection for Adjacency Matrix

$Rows = call\ min(Rows1, Rows2)\ function$ ▷ t ← t+1
$Column = call\ min(Column1, Column2)\ function$ ▷ t ← t+1
$call\ Graph()\ function$ ▷ t ← t+1
while $i < Rows$ **do** ▷ t ← t+1
  while $j < Columns$ **do** ▷ t ← t+1
    if $Adjacent1[i,j] == 1 and Adjacent2[i][j] == 1$ **then** ▷ t ← t+1
      $call\ AddEgde()\ function$ ▷ t ← t+1
    end if
  end while
end while
$call\ drawnodes()\ function$ ▷ t ← t+1
$call\ drawedges()\ function$ ▷ t ← t+1
$call\ drawlabels()\ function$ ▷ t ← t+1
$call\ plotgraph()\ function$ ▷ t ← t+1

Our approach to given problem for Incidence Matrix input includes the following steps:

1. We are taking the number of vertices and number of edges as an input.

2. Create two incidence matrices with the help of inputs.

3. Traverse the first incidence matrix column wise and for all the rows check which nodes are valued 1 and store them in some variables.

4. Without looping out of that column loop traverse the second incidence matrix column wise and for all the rows store the nodes that are valued 1 in some other variables.

5. If the respective variables match then add it as an edge in the graph otherwise jump to other column of first incidence matrix.

The pseudo code for this algorithm is on right side:

Algorithm 4 Input for 1st Incidence matrix

$InputNumberofvertices$ ▷ t ← t+1
$InputNumberofedges$ ▷ t ← t+1
while $i < edges$ **do** ▷ t ← t+1
  $InputVertex(u)$ ▷ t ← t+1
  $InputVertex(v)$ ▷ t ← t+1
  $Incidencematrix1[u][i] = 1$ ▷ t ← t+1
  $Incidencematrix1[v][i] = 1$ ▷ t ← t+1
end while
$Rows1 \leftarrow Numberofvertices$ ▷ t ← t+1
$Columns1 \leftarrow Numberofedges$ ▷ t ← t+1

Algorithm 5 Input for 2nd Incidence matrix

$InputNumberofvertices$ ▷ t ← t+1
$InputNumberofedges$ ▷ t ← t+1
while $i < edges$ **do** ▷ t ← t+1
  $InputVertex(u)$ ▷ t ← t+1
  $InputVertex(v)$ ▷ t ← t+1
  $Incidencematrix2[u][i] = 1$ ▷ t ← t+1
  $Incedencematrix2[v][i] = 1$ ▷ t ← t+1
end while
$Rows2 \leftarrow Numberofvertices$ ▷ t ← t+1
$Columns2 \leftarrow Numberofedges$ ▷ t ← t+1

Algorithm 6 Intersection for Incidence Matrix

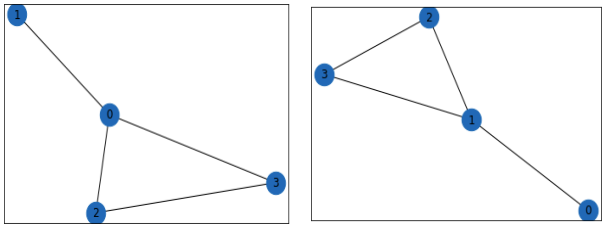
$call\ Graph()\ function$ ▷ t ← t+1
while $i < columns1$ **do** ▷ t ← t+1
  while $j < rows1$ **do** ▷ t ← t+1
    if $incident1[j,i] == 1$ **then** ▷ t ← t+1
      $store\ two\ nodes\ in\ variables\ f and s$ ▷ t ← t+2
    end if
  end while
  while $k < columns2$ **do** ▷ t ← t+1
    while $l < rows2$ **do** ▷ t ← t+1
      if $incident2[l,k] == 1$ **then** ▷ t ← t+1
        $store\ two\ nodes\ in\ variables\ a and b$ ▷ t ← t+2
      end if
    end while
    if $a == f\ and\ b == s$ **then** ▷ t ← t+2
      $call\ AddEgde()\ function$ ▷ t ← t+1
    end if
  end while
end while
$call\ drawnodes()\ function$ ▷ t ← t+1
$call\ drawedges()\ function$ ▷ t ← t+1
$call\ drawlabels()\ function$ ▷ t ← t+1
$call\ plotgraph()\ function$ ▷ t ← t+1

IV. Algorithm Analysis

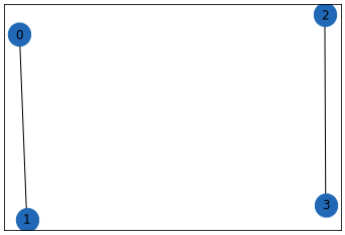
The above algorithm is designed to reduce the time complexity for this question as much as possible.The algorithm for Adjacency matrix is straightforward just finding the common node points but in incidence matrix representation the algorithm needs to be worked column wise as each edge needs to be looked over rather than nodes as the numbering of the edges could be different in two graphs.The algorithm that we proposed has kept in mind the origin of question that is not to modify incidence matrix. For decreasing the time complexity in case of incidence matrix , it could be converted into adjacency matrix but then it will have no meaning to the question.The graph here is plotted using the Networkx library in python.

V. Experimental Study

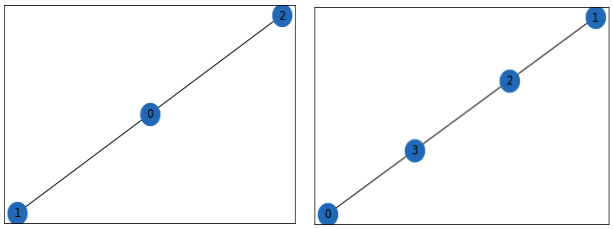
Disconnected graph :



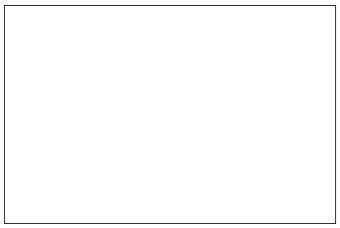
After applying the algorithm on the above input graph , the intersection operation comes out to be disconnected graph as edge 2-3 and 0-1 common in both.



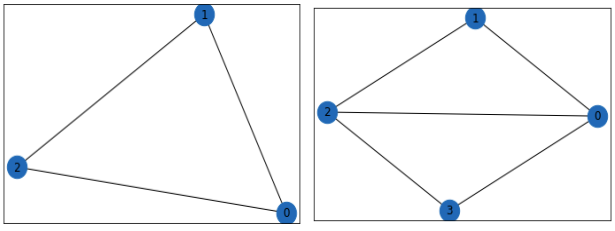
Empty/Null graph :



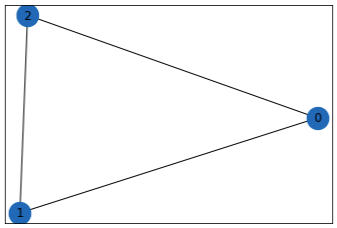
After applying the algorithm on the above input graph, the intersection operation comes out be Null/Empty Graph as no common edges in both graph .



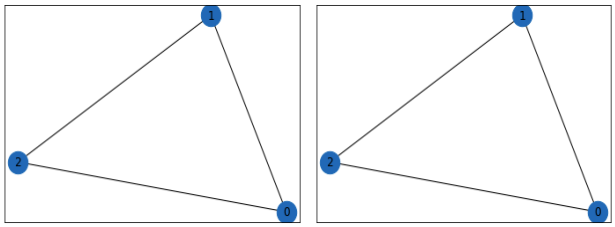
Connected graph :



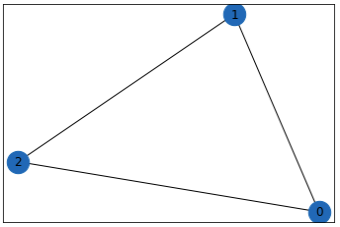
After applying the algorithm, the intersection operation of above both graph comes out be connected graph.



Original graph:



After applying the algorithm on the above input graph, the intersection operation comes out be Original Graph as our both input graph is same so their intersection is also same.



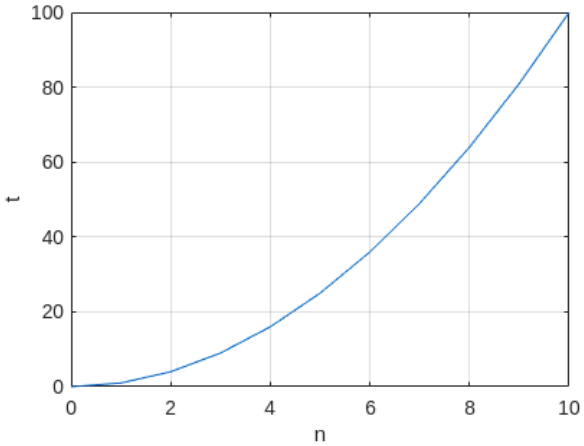
A. Time Complexity

The time complexity analysis of an Algorithm is quite necessary to learn about it's efficiency and optimization.

The time complexity of an algorithm(1) for adjacency matrix is-

$$T = O(max(n1^2, n2^2)).$$

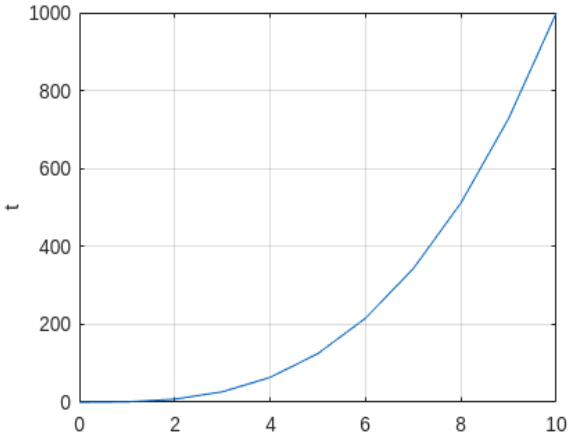
Where n1 and n2 is the number of nodes in the graph1 and graph2.



The time complexity of an algorithm(2) for incidence matrix is-

$$T = O(e1 * n1 + e1 * e2 * n2)).$$

Where e1 and e2 is the number of edges in graph1 and graph2. n1 and n2 is the number of node in graph 1 and graph2.



B. Space Complexity

The Space Complexity of algorithm(1) for adjacency matrix is-

$$O(max(n1^2, n2^2)).$$

where n1 and n2 is number of node in the graph1 and graph2. The Space Complexity of algorithm(2) for incidence matrix is-

$$O(max(n1 * e1, n2 * e2)).$$

where n1 and n2 is number of node in the graph1 and graph2, e1 and e2 id number of edges in the graph1 and graph2.

VI. Conclusion

In this paper, we demonstrated the intersection operation on two graphs using Adjacency Matrix and Incidence Matrix both. The operation is valid on all kind of graphs, which are Empty, Disconnected, Connected and original graphs.The order of time complexity is $O(n^2)$ and $O(n^3)$ respectively, where n is the number of edges(nodes). The pruning in the code has been performed quite efficiently in case of fully connected as well as disjoint graphs.

References

[1] Clyde LMonma and Victor K Wei, ”Intersection Graphs of Paths in a Tree” Bell Communications Research, Morristown, New Jersey 07960 USA,Received 8 February 1985

[2] Tom Brylawski, ”Intersection theory for graphs” JOURNAL OF COMBINATORIAL THEORY, Series B 30. 233-246 (1981)

[3] Online ”https://en.wikipedia.org/wiki/Intersection graph”

[4] Introduction to Algorithm 3rd Edition By Thomas H.Cormen

[5] Online ”https://en.wikipedia.org/wiki/Intersection(set theory)”

[6] Online ”https://www.overleaf.com”