Detection of Pendent Edges and Pendent Vertices in Incidence Matrix

Swarnima Dixit
IWM2014003
IIIT Allahabad

Akhila Jetty
IRM2014006
IIIT Allahabad

Arun Kumar Reddy
IRM2014005
IIIT Allahabad

Abstract- The objective of this work is to find accurate and optimal solutions to the problem of finding the pendent vertices and pendent edges given the incidence matrix.

I. Introduction

A graph is a representation of a set of objects called vertices where some pairs of the vertices are connected together by links called edges. A simple graph consists of a set of vertices V and a set of edges E [G = (V, E)] can be represented by unordered pairs of elements of V .

Vertices are the nodes present in the graph.For example,in figure 1 nodes v_1, v_2, v_3, v_4 and v_5 are the vertices of the graph and edges are defined as the path between two vertices of a graph or a line between two vertices of a graph. In figure 1 for example, the links between e_1 ,e_2, e_3 , e_4, e_5, e_6, e_7 and e_8.

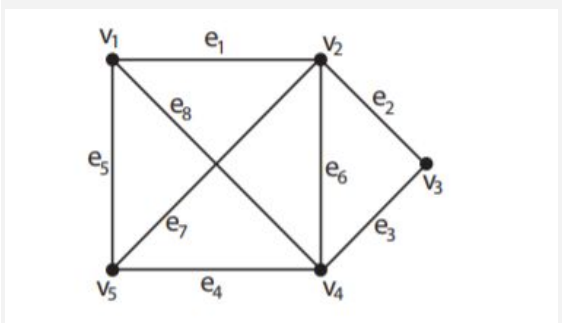

Figure 1: A simple graph.

II. Motivation

The goal is to design different approaches to the problem of finding pendent vertices and edges and analyse them and find the most optimal solution.

Incidence Matrix : Incidence matrix is a form of a representation of a graph G (V, E) where there is an edge j from the vertex i if A_{ij} is equal to 1 and 0 otherwise. The size of an incidence matrix is given by n*m where n is number of vertices and m is the number of edges.

Pendent vertices[2] are defined as those vertices in the graph which are connected to only one edge and these corresponding edges are called as pendent edges[1].

Here we are trying to find the number of pendent vertices and edges when the incidence matrix is given.

For example-1, in the given graph

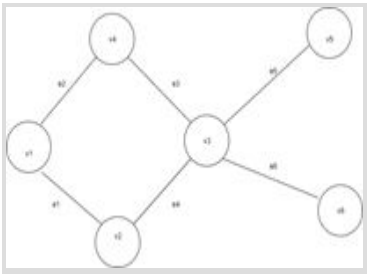

Figure 2: Representation of Graph

The incidence matrix for the above graph is :

	e1	e2	e3	e4	e5	e6
v1	1	1	0	0	0	0
v2	1	0	0	1	0	0
v3	0	0	1	1	1	1
v4	0	1	1	0	0	0
v5	0	0	0	0	1	0
v6	0	0	0	0	0	1

Figure 3: Incidence Matrix of given Graph

III. Implementation

We came up with three techniques to identify the pendent vertices and the pendent edges from the incidence matrix. The first two approaches involve finding the pendent vertices first

and then deriving their corresponding edges whereas the third approach first finds the pendent edges and then derives their corresponding pendent vertices.

Approach - 1:

Let the graph G (V, E) be represented by the incidence matrix A with size n∗m where n is the number of vertices and m is the number of edges.

- Consider a boolean array of size n to store whether the corresponding vertex is pendent or not and initialise it to false.
- The number of pendent vertices is vertex_count and the number of pendent edges is edge_count , both are initially zero.
- For each vertex in the graph , find the number of edges it is connected to by traversing its corresponding row. If this number is 1, then its a pendent vertex. Mark it and increment the vertex_count.
- For each edge , if any of its two vertices is pendent then increment the edge_count.

Time complexity: O(n∗m)

Space complexity: O(n)

Approach - 2:

Let the graph G (V, E) be represented by the incidence matrix A with size n∗m where n is the number of vertices and m is the number of edges.

- Consider a boolean array of size m to store whether the corresponding edge is pendent or not and initialise it to false.
- The number of pendent vertices is vertex_count and the number of pendent edges is edge_count , both are initially zero.
- For each vertex in the graph , find the number of edges it is connected to by traversing its corresponding row and store it in a temporary variable. If this number is 1, then its a pendent vertex and increment the vertex_count.
- If the corresponding edge was not marked, mark the edge and increment edge_count.
- This is more efficient compared to the previous approach as it iterates the incidence matrix only once instead of twice.

Time complexity: O(n∗m)

Space complexity: O(m)

The above two approaches identified the pendent vertices first while the next approach will identify the pendent edges first.

Approach - 3:

Let the graph G (V, E) be represented by the incidence matrix A with size n∗m where n is the number of vertices and m is the number of edges.

- Consider a boolean array of size n to store whether the corresponding vertex is pendent or not and initialise it to false.
- The number of pendent vertices is vertex_count and the number of pendent edges is edge_count , both are initially zero.
- For each edge in the graph , find its corresponding vertices and check if both these vertices are connected to other edges.
- If not then the edge is a pendent edge , so increment the edge_count. if either of the vertices of the edge are not marked, then mark the vertex and increment the vertex_count.

Time complexity: O(m∗(n+m))

Space complexity: O(n)

IV. COMPARISON AND RESULTS

A. T VS V (time complexity vs number of vertices)

a. For approach 1 :

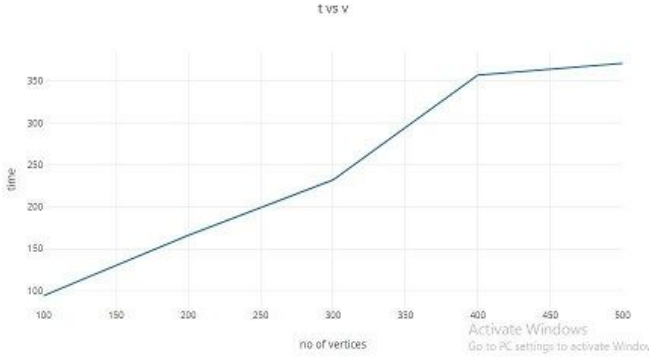

Figure4: Graph between Time vs Vertices where edges are fixed for approach 1

b. For approach 2:

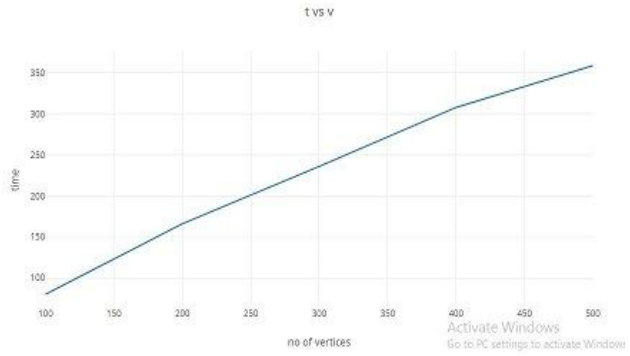

Figure 5:Graph between Time vs Vertices for fixed number of edges for approach 2.

c. For approach 3:

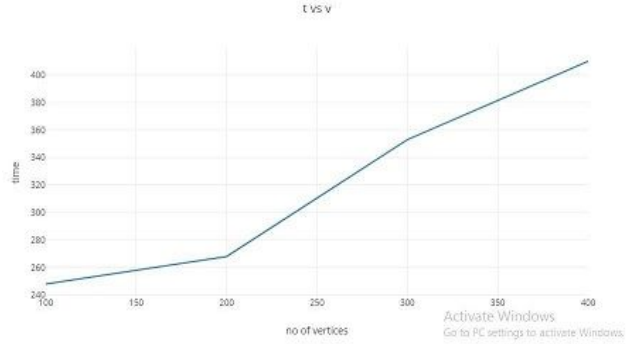

Figure 6:Graph between Time vs Vertices for fixed number of edges for approach 3.

B. T VS E (time complexity vs number of edges)

a. For approach 1 :

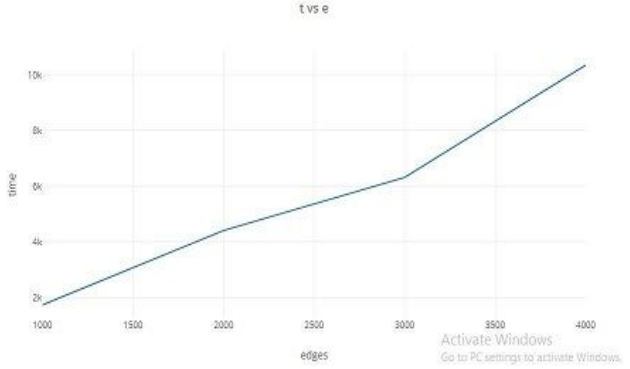

Figure 7: Graph between time complexity vs edges for fixed number of vertices for approach 1.

b. For approach 2:

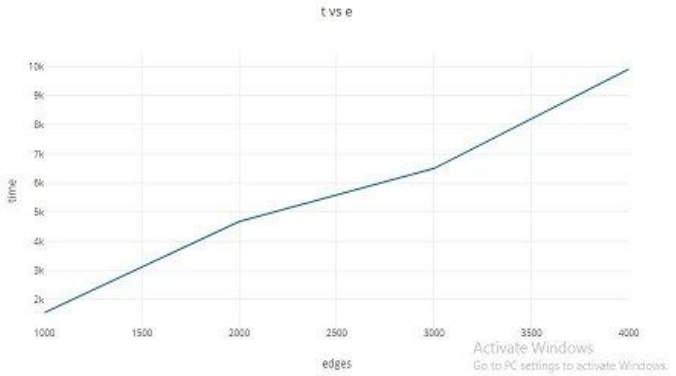

Figure 8:Graph between Time vs Edges for fixed number of vertices for approach 2.

c. For approach 3:

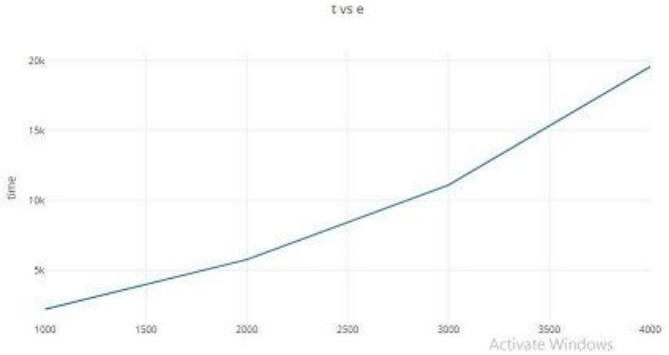

Figure 9:Graph between Time vs Edges for fixed number of vertices for approach 3.

We proposed three approaches for the detection of pendent vertices and pendent edges. Analysing their time complexities, it turns out that they are of fairly the same order. The comparison of runtimes though shows us that approach 2 is the fastest as it involves just a single iteration of the matrix and approach 3 as the slowest. If the graph is very condense, approach 2 can be expensive in memory because the auxiliary space used is in the order of the number of edges. In such cases, the other approaches may be preferred.

CONCLUSION

We can use all these approaches in different scenarios for getting good results.Theoretically all approaches have same time complexity but in real time problems we can use them according to the scenario.

REFERENCES

1. http://mathworld.wolfram.com/PendantEdge.html
2. http://mathworld.wolfram.com/PendantVertex.html