Given an adjacency matrix and find out pendant vertices and edges for directed and undirected graph.

Shiv Pratap Singh[1], Nazish Tabassum[2] and Arqum Ahmad[3]

Abstract— **An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.The adjacency matrix is a (0,1) matrix with zeros on its diagonal.**
Pendant vertex: A vertex of a graph is said to be pendant if its neighborhood contains exactly one vertex.
Pendant edge: An edge of graph is said to be pendant if one of its vertices is a pendant vertex.

I. MOTIVATION

Pendant vertex is basically a leaf node of a graph with constraint that it has degree exactly equal to one.It is important to find Pendant vertex and Pendant edge to know the end of a graph. Pendant vertex is basically a leaf node of a graph with constraint that it has degree exactly equal to one.It is important to find Pendant vertex and Pendant edge to know the end of a graph.

II. ALGORITHM

A. For Undirected Graph

FindPendantVertices(M)
Input : An adjacency matrix M.
Output : A set of pendant Vertex V.

```
for i := 1 upto N
    count := 0
    for j := 1 upto N
        if M[i][j] = 1
            count := count + 1;
    if count = 1
        V.insert(i)
```

FindPendantVertices(M)
Input : An adjacency matrix M.
Output : A set of pendant Vertex V.

```
for i := 1 upto N
    for j := 1 upto N
        if M[i][j] = 1
            degree[i] = degree[i] + 1
            degree[j] = degree[j] + 1
for i := 1 upto N
    If degree[i] = 2
        V.insert(i)
```

FindPendantEdges(M, V)
Input : An adjacency matrix M and set of pendant Vertex V.
Output : A set of pendant edges E

```
for i := AllElementOf(V)
    for j := 1 upto N
        If M[i][j] = 1
            E.insert(i, j) , Break
```

B. For Directed Graph

FindPendantVertices(M) Input : An adjacency matrix M.
Output : A set of pendant Vertex V.

```
for i := 1 upto N
    for j := 1 upto N
        if M[i][j] = 1
            outdegree[i] = outdegree[i] + 1
            Indegree[j] = indegree[j] + 1
For i := 1 upto N
    If outdegree[i] + indegree[i] = 1
        V.insert(i)
```

FindPendantEdges(M, V)
Input : An adjacency matrix M and set of pendant Vertex V.
Output : A set of pendant edges E

```
for i := 1 upto N
    for j := 1 upto N If M[i][j] = 1 and ( i  V or j  V)
        E.insert(i, j)
```

III. DESCRIPTION

To find the pendant vertex in directed as well as undirected graph , every vertex needs to be visited. If the adjacent neighbour has one and only vertex, insert it into set V.
To find pendant edges, check the adjacency of every pendant vertex(i) in set V with other vertex(j) in graph. If there is an edge, insert the pair(i,j) in the set E.
To find pendant vertex in directed graph, Indegree as well as out degree of every vertex need to be calculated. Now, if for every vertex(i) if the sum of indegree and outdegree is equals to 1, insert the vertex(i) to set V.
For pendant edges in directed graph, we check edges from every vertex. If there is an edge and any of the vertex lies in Set V, then the edge is called Pendant edge.

IV. IMPLEMENTATION AND RESULTS

The algorithm has been implemented in c++ language. Test case are generated manually using c++ library. For particular input size running time of input is calculated. The tool used for plotting graph is gnuplot. We have 2 sets for pendant vertices (directed and undirected) and 2 sets for pendant edges(directed and undirected) .

A. *Plot of time taken vs input size*

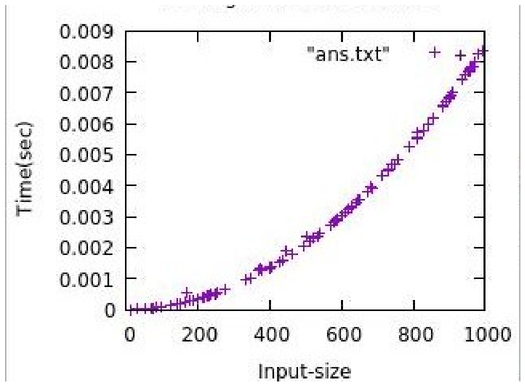

Fig. 1. Finding Pendant Vertex Time

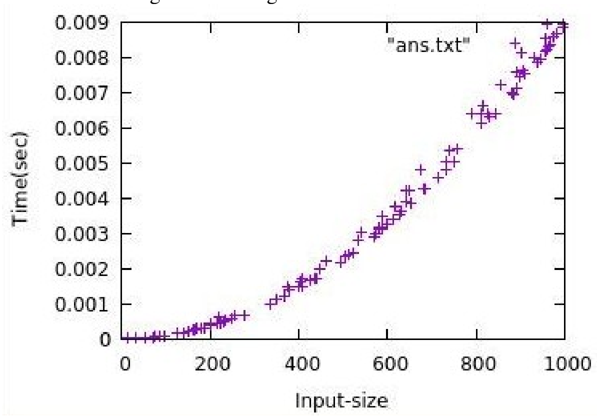

Fig. 2. Finding Pendant Vertex Time for Directed Graph

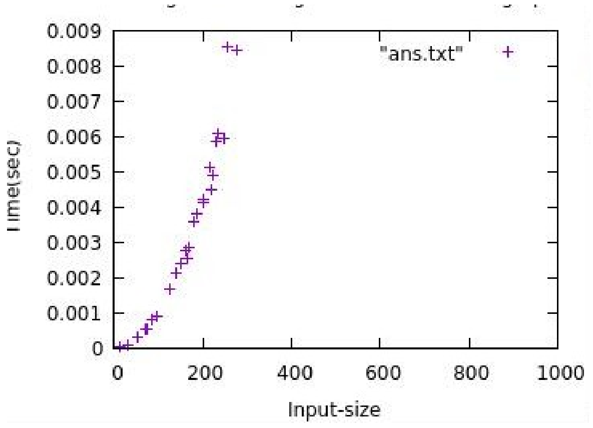

Fig. 3. Finding Pendant Edge Time

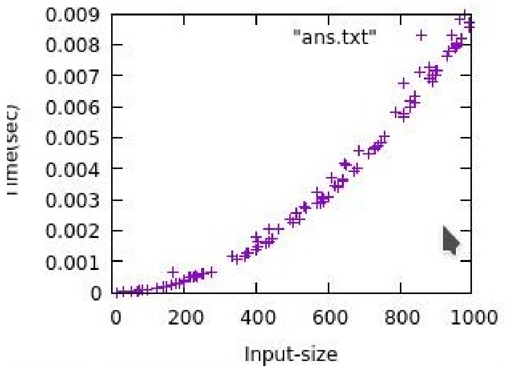

Fig. 4. Finding Pendant Edge Time for Directed Graph

V. Conclusion

N = Total number of vertices in a graph.
(For Undirected graph)
The time complexity to find Pendant vertex is O(N^2)
Time complexity to find pendant edges is O(nxN)
where n = number of Pendant vertices.

(For Directed graph)

The time complexity to find Pendant vertex is O(N^2)
Time complexity to find pendant edges is O(nxN)
where n = number of Pendant vertices.

References

[1] GNU Plot http://people.duke.edu/ hpgavin/gnuplot.html
[2] Definition of pendant edges and vertices https://www.wolframalpha.com/