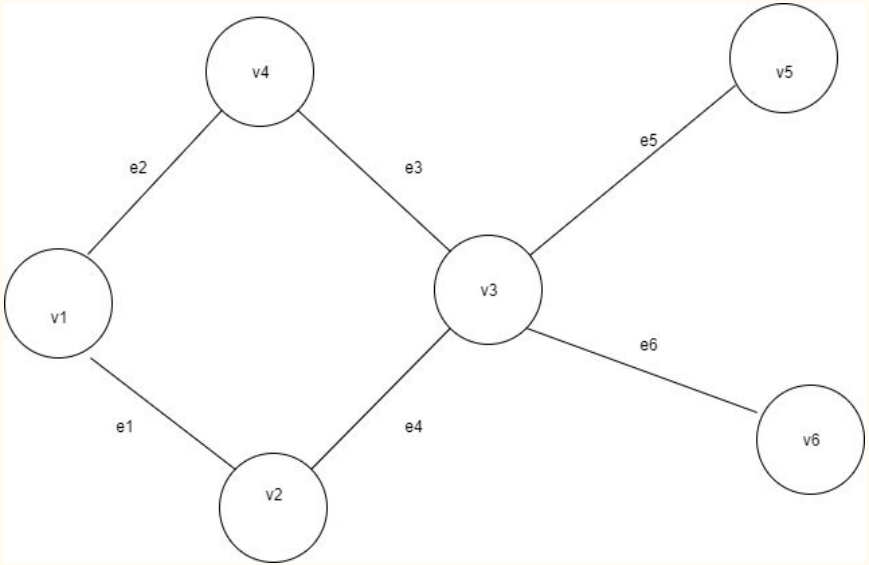
Detecting Pendent Vertices and Pendent Edges in Incidence Matrix

Akhila Jetty (IRM2014006)
Arun Kumar Reddy (IRM2014005)
Swarnima (IWM2014003)

Problem Definition and Objective

- Given an incidence matrix, the objective is to try and design an algorithm to find the pendent vertices and edges present in the graph. The idea is to find such vertices and edges independently.

	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



Methodologies

★ Approach 1:

```
n: vertices m: edges g(n,m): graph
Pendent(n,m)
        pendent[n] = false
        vertex = 0 // number of pendent vertices
        edge = 0    // number of pendent edges
        for(i: 0 to n)
                count = 0
                for(j:0 to m)
                        if(g(i,j) := 1)
                                count = count+1
                if(count := 1)
                        vertex = vertex+1
                        pendent[i] = true;
        for(i: 0 to m)
                for(j: 0 to n)
                        if(g(j,i) := 1 && pendent[j] := true){
                                edge = edge+1
                                break
```

Time Complexity : O(nxm)
Space Complexity : O(n)

★ **Approach 2:**

```
n: vertices ,m: edges ,g(n,m): graph
Pendent(n,m)
        pendent[m] = {false}
        vertex := 0// number of pendent vertices
        edge := 0   // number of pendent edges
        for(i: 0 to n)
                count := 0
                index := 0
                for(j:0 to m)
                        if(g(i,j) := 1)
                                count := count+1
                                index :=  j
                if(count := 1 )
                        vertex = vertex + 1
                        if(pendent[index] := false)
                                pendent[index] := true
                                edge := edge + 1
```

Time Complexity : O(nxm)
Space Complexity : O(m)

★ Approach 3:

```
n: vertices m: edges g(n,m): graph
Pendent(n,m)
        pendent[n] = false
        vertex = 0 // number of pendent vertices
        edge = 0    // number of pendent edges
        for(i: 0 to m)
                u,v //vertices of edge i
                for(j: 0 to m)
                        if(j!=i)
                                if(g(u,j) := 1)
                                        flag1 = true
                                if(g(u,j) := 1)
                                        flag2 - true
                if(!flag1||!flag2)
                        edge := edge +  1
                if(!flag1 && !pendent[u])
                        vertex = vertex + 1
                        pendent[u] = true
                if(!flag2 && !pendent[v])
                        vertex= vertex + 1
                        pendent[v] = true
```

Time Complexity : O(mx(n+m))
Space Complexity : O(n)

Result & Analysis