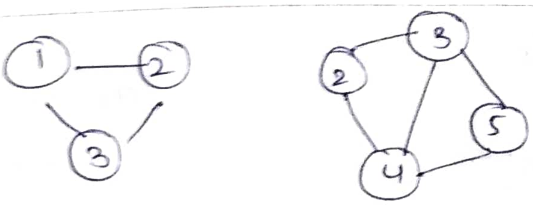
A connected component is a subgraph in which any two Vertices are connected to each other by paths & which is not connected to additional vertices of the super graph. def

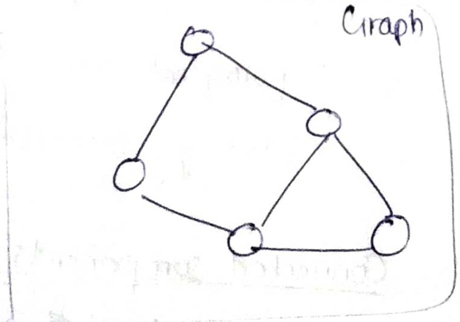
eg:-



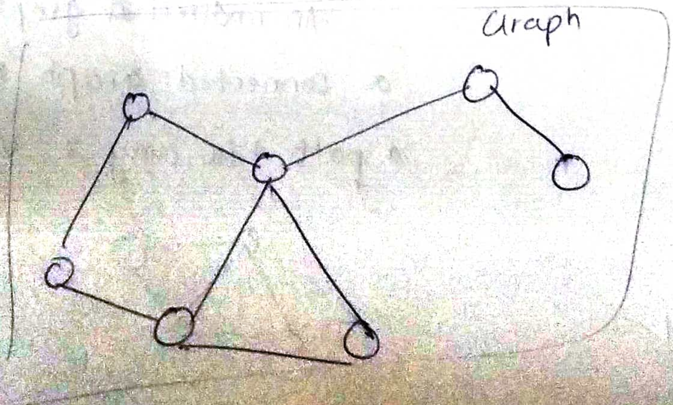
Algorithm:-

```
                      graph
Connected_Components ( G )
  1. for Each Vertex V ∈ G(V)
  2.    Make-set (v)              unvisited vertex
                                 visited vertex
  3. for each edge (U,V) ∈ G (E)
  4.    ~~for~~ if find_set(u) ≠ find_set (v)
              union (U,V)

        if   find_set (u) == find_set (v)
              return true

        else
              return false
```

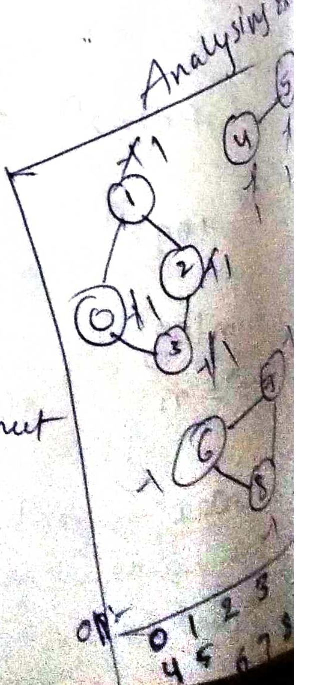
By jenny.

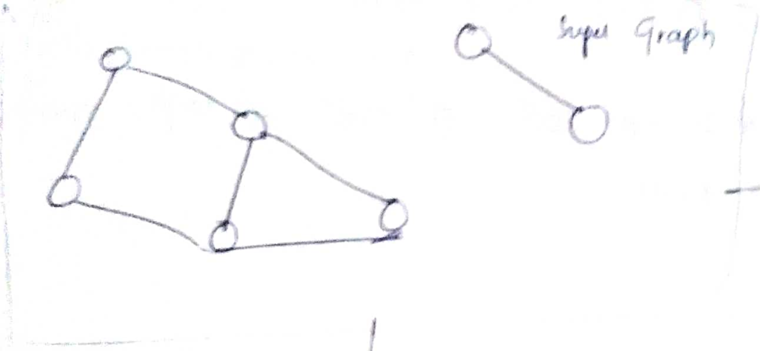


1 Connected component



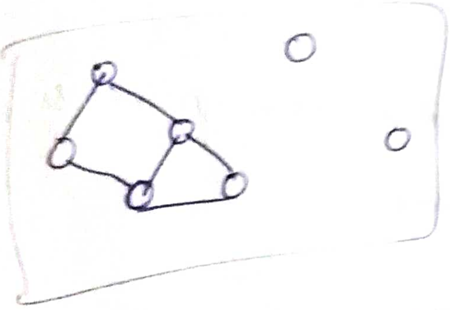
1 Connect componet



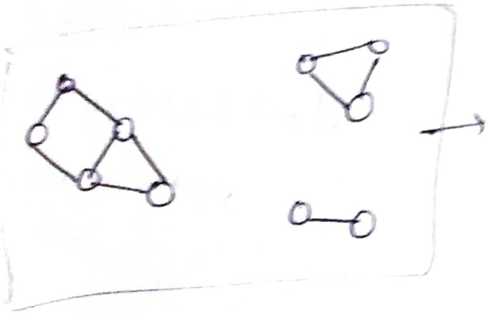


→ 2 connected components

↳ everything is not connected



→ 3 Connected Components



→ 3 connected components

pictorial representations

To find connected Components (Algo) either DFS or BFS is used
↓
no.of DFS traversals will gives us no.of connected components

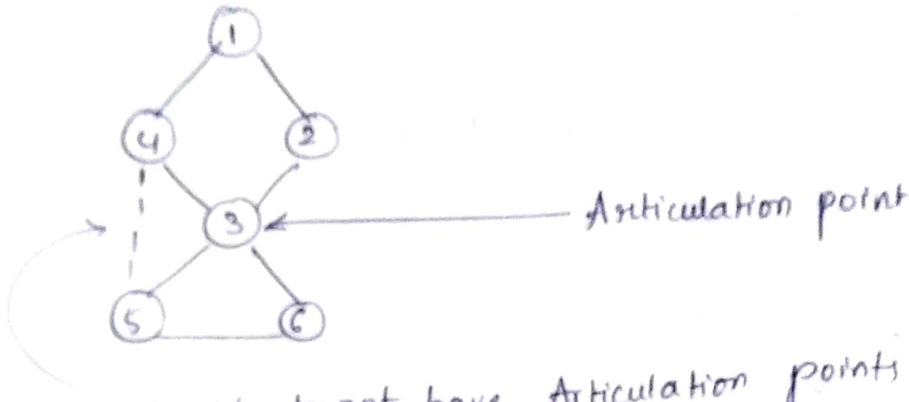
Algo.

```
Connected_components (G)
{
  for each vertex v∈N        8 in Analysing ex
      flag [v]=-1
  Count=0                     v=0
                              v=4
                              v=6
  for( int v=0; v<N ; v++ )v=9✗
  {
      if ( flag[v] == -1 )
      { DFs (v, flag)
        count++               1    2    3
      } cout<<"\n";           ↓    ↓    ↓
                            with  with  with
  }                          v=3   v=5  v=8
  ~~Count~~ cout<< Count

DFs (int v, int flag)
  {
     flag[v]=1
     Cout <<v
     for each adjacent node u of v
     if(flag[u]==-1)
          DFs (u, flag)
  }
```

=) A graph is Biconnected if it contains no 'articulation' point.

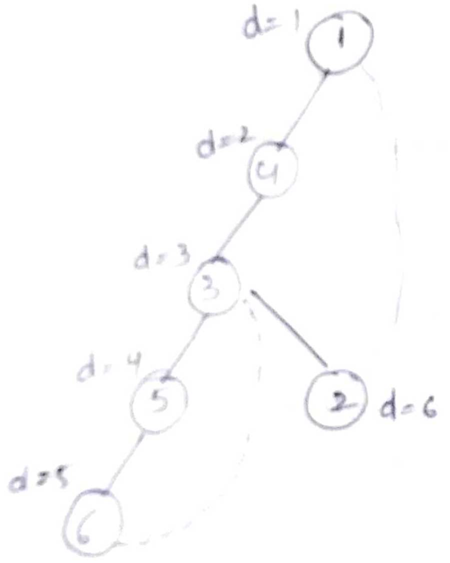
Articulation point:

The vertex whose removal will generate multiple components is called Articulation point



→ It's always preferable to not have Articulation points so always try to remove it, by connecting edges

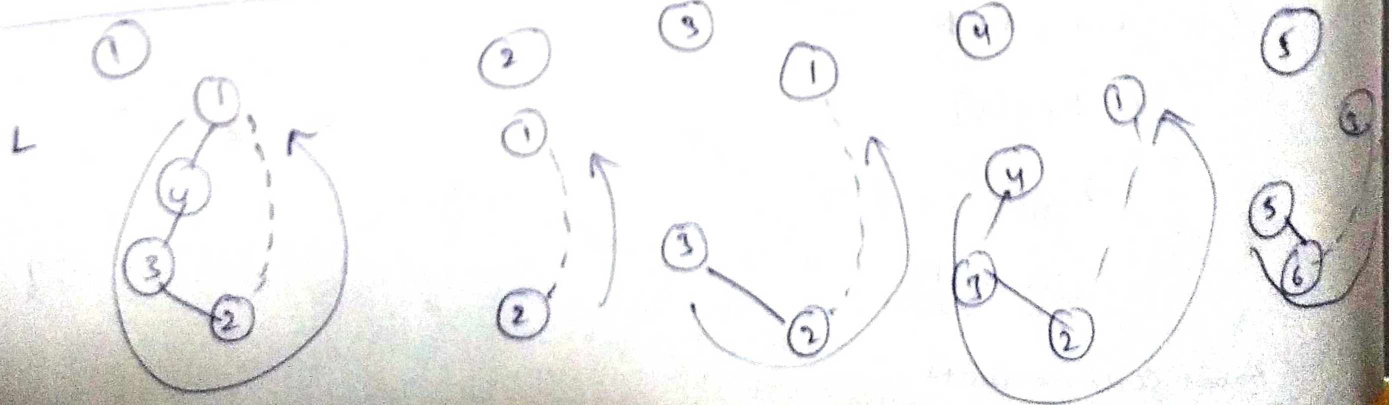
DFS :- for above graph



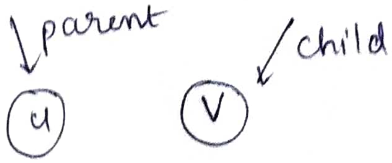
d → discovery time
order in which vertices are visited

L → Tracing back and reaching possible parent
(lowest dfs no)

	①	②	③	④	⑤	⑥
d	1	6	3	2	4	5
L	1	1	1	1	3	3



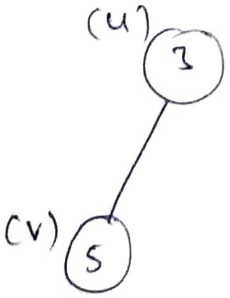
Biconnected graphs



if (L[V] ≥ d[u])

then u is an Articulation point

this Condition don't work for root



L[5] d[3]

3 ≥ 3 → so 3 has an articulation point

for root:-

if root contains multiple children then root is an articulat point