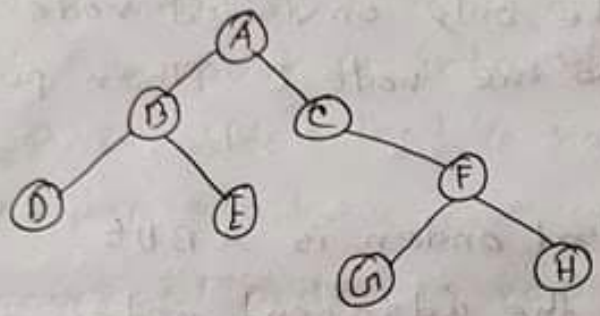
Assignment - 2

1. Full tree Traversal

Tree is a special kind of a cycle graph that contains no cycle.
Mainly we use this algorithm for full tree traversal




At first here, we will use time traversal to traversal.
Here, A is the root node.
After that, we push A in the stack. Then we will go in B and push B in the stack.
So, now the traversal order is: AB

Then, we will check the connected node of B. And go one of then those two are connected. Then we will go to D and push it into the stack.

so, the traversal order is now : ABD.

Now, there in the top of the stack is D and DFS will check ~~upon~~ where we can go by D. from this tree we can see that D is only connected with B. But we traversal D already. Thats why we pop stack And go back to node B. Then DFS will check unvisited nodes. The only unvisited node is E. Then we will go to the node E. Then push E in the stack.

so. Now the traversal order is A B D E

Then, we will check the adjacent nodes of E, E have only one adjacent node and that is B. And Here B is already visited.

so, E don't have any adjacent nodes, in left side. pop stack and go back to E. There are no visited adjacent nodes at B. so, pop stack again. we came back into order A. Now there is only one adjacent not of A and it is C that is unvisited. so, DFS will go into C. push C in stack.

Now, the Traversal is A B D E C.

Now, we will check adjacent nodes of C. C have only one adjacent node and that is F. Now, we push F in the stack.

Now, the Traversal order is ABDECF.

Then, we will check adjacent nodes of F has two unvisited node that is G and H. Then F choose G 1st. Then push G into the stack. Then we will go G. After that G has no unvisited node so we pop stack.

The traversal order is now ABDECFG.

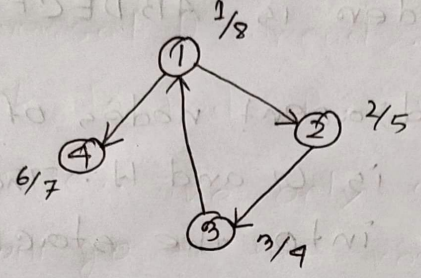
After that we came back into F. F has another unvisited node and it is H. Push H into stack. It has no unvisited node so pop the stack and came back to F.

Now, the Traversal is ABDECFGH

Here, F does not have any unvisited node, so stack pop. Then came to C. C has no unvisited node. so, again stack pop. Then we came back to A. A haven't any unvisited nodes. so, again stack pop and tree traversal has stopped.

## 2. Cycle finding

If there is a back edge in a graph than we can say that the graph has a cincle.

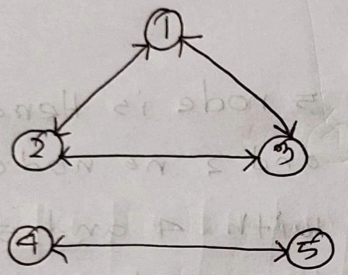



1st, we will start from 1 visited. Here in 1st node ① 1 is a starting time. Then we will go to 2 no node and the starting time is 2. Then we wil go to 3no node and here the starting time is 3. Then we can go 3 to 1 no node ~~th~~but 1 no node is already visited. so, we canit go to 1 no node. so, we will back in 3 so, now 3 no node ending time is 4. Then we will back into 2 no node. Here the ending time is 5. Then we will go to 1no node that is connected with on other node and it is 4. then we will go to 4 no none. Here the starting time is 6. Then we see that we canit go anywhere from 4 no node. so here the ending time is 7. Then we go back into node 1 and itis ending time is 8.

Here node 3 and node 1 connected with the back edge, so, the graph contains a cincle.

## 3. Component Finding

Here, we will find out how many sub graph there are in a graph.

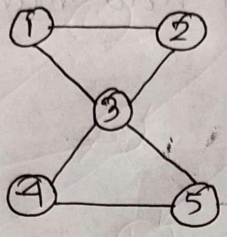



- Here at first in this graph we will check and run DFS. After 1 DFS if there find any unvisited nodes then we will run DFS from that node again. And ~~can't~~ count it.

From this, three graph we can find two components.

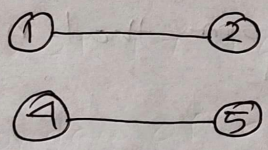
## 4. Articulation point finding

Here the main point is that How a separate graph can be found excluding any node or any edge.




Here in this graph we see that 5 node is Here And 3 no node is connected with 1 and 2 no node and also 3 no node is connected with 4 and 5 no node. so, here if we excluding 3 then we find two graph.

so, It has an articulation point. These two graph that we find from the main graph are:



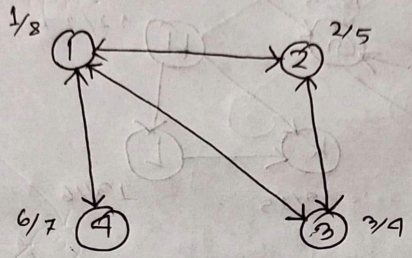

and




but in these two graph has no articulation point. Because, here we can't find any node that we can excluding.

## 5. Topological Sort

The main theme is here that sort node list by decending order of end time.




By cycle finding we can find here the sorting time and the ending time.

| Node: | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| starting time: | 1 | 2 | 3 | 6 |
| ending time: | 8 | 5 | 4 | 7 |

Now, Decending order of ending time is 8 7 5 4.

Now the node is like that by following the decending order of ending point is.

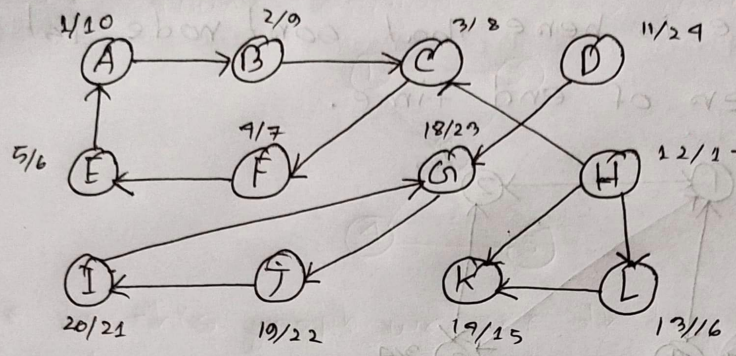
1 4 2 3

Here the topological sort is

1 4 2 3

## 6. Strogly Connected Component

At first we will run DFS and make a topology sort.




| Node: | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| starting time : | 1 | 2 | 3 | 11 | 5 | 4 | 18 | 12 | 20 | 19 | 14 | 13 |
| ending time : | 10 | 9 | 8 | 24 | 6 | 7 | 23 | 17 | 21 | 22 | 15 | 16 |

Decending order of ending time is

24. 23. 22. 21 17 16 15 10 9 8 7 6

By decending order of ending time the node is now:

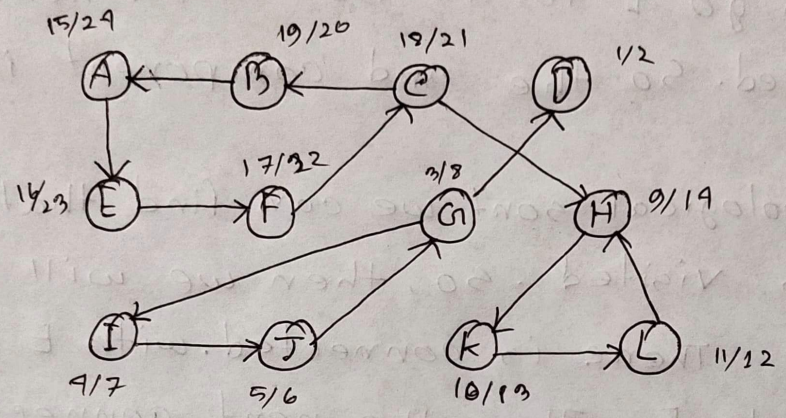
D G J I H L K A B C F E

So, the topological sort is

D G J I H L K A B C F E

Now, we will change edge direction. Then run DFS using topological sort.

Here, D is a single component.




we know that the topological sort from graph 1 is

D G J I H L K A B C F E

Now, we start it will D.

From this graph we can see that we can't go anywhere from D. so, D is the 1st component, Here.

Then the element is G that we find in topological sort.

From G we can go I, then we can go J then can go to G but G is already visited. So, the 2nd Component is G I J

Then we can find that next element in H, that is not visited. So the next starting node is H.
From H, we can go to k, that is connected with L.
Then we can go L to H. But ~~there~~ Here H is already visited. So the 3rd component is H K L.

From the topological sort we can find that D G J I H L k is visited. So, then we will start from node A. There is connected with E with direction A to E. Then the next connected direction is E to F, F to C, C to B, and B to A
But here A is already visited.

So, The 4th component is
A E F C B

Now, the strongly connected component are
D
J I G
L K H
B C F E A.