(1) ID: 191-15-12485
Ummay Habiba

1. Full TreeTraversal

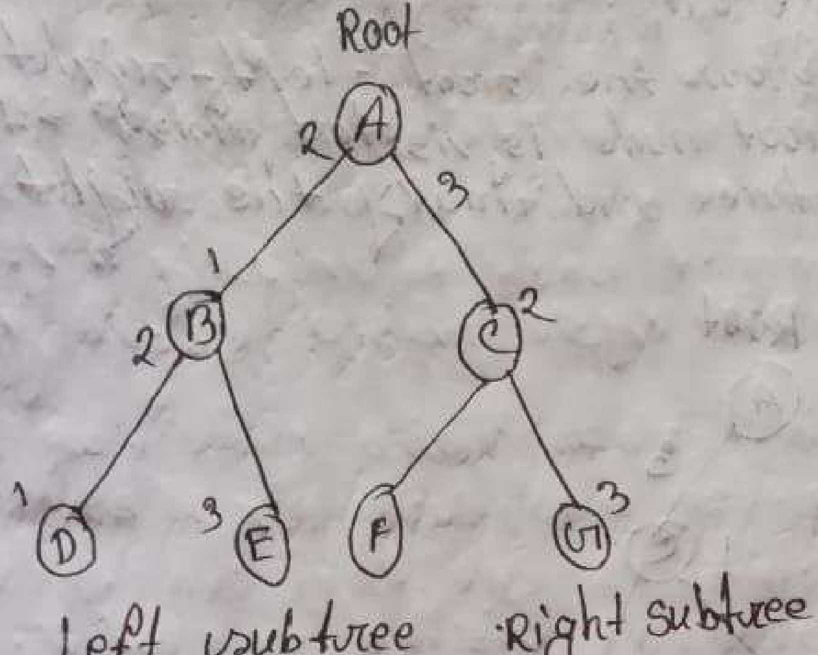
Traversal is a process to visit all the nodes of a tree and may print their values too. Because all nodes are connected via edges (links) we always start from the root (head) node. That is we can not randomly access a node in a tree. There are three ways which we use to traverse a tree-

1. In-order traversal
2. Pre- ~~ed~~order traversal
3. past-order traversal

Generally, we traverse a tree to search or locate a given item or key in the tree or to Print all the values it contains.

In-order Traversal:

In order traversal method, the left subtree is visited first, then the root and later the right sub-tree. Every node may respresent a subtree itself. In case of binary tree. It is traversed in-order. the output will Produce sorted key values in a asscending order.



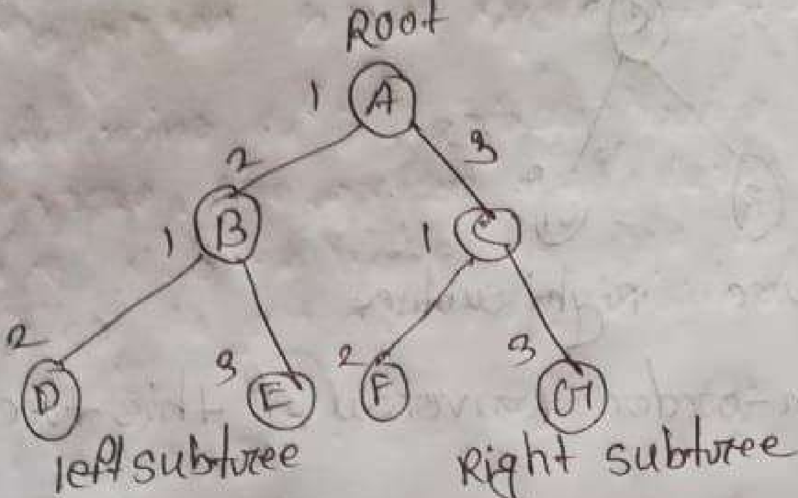
The output of in-order traversal of this tree will be -

D → B → E → A → F → C → G

we ~~start~~ form A. and following in-order traversal we move to its left subtree B. B is also traversed in order. It will continue untill all the nodes are visited.

In algorithm:
At first recursively traverse left subtree, then visit root node. Finally recursively traverse right subtree.

Pre-order traversal:
This method follows the root - left - right formud. The root node is visited first - then the left subtree and finally the right subtree.



The output of pre-order traversal of this tree will be -

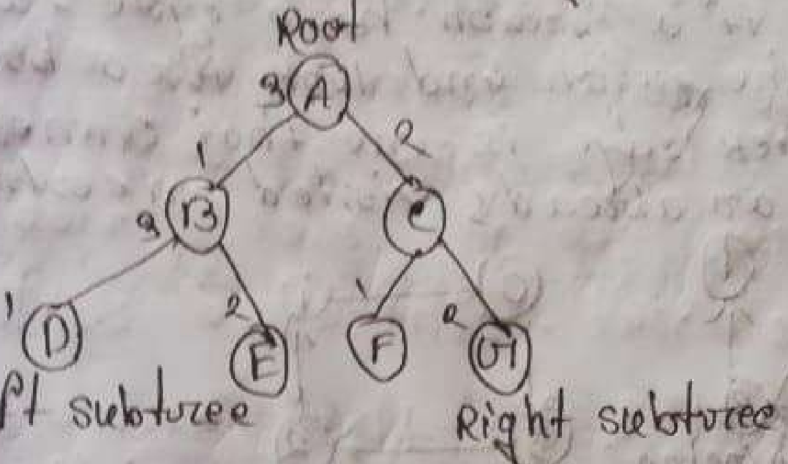
A → B → D → E → C → F → G

we start from A, and following pre order traversal, we first visit a itself and then move to its left subtree B. B is also traversed pre-order. The process goes on untill all the nodes are visited.

In Algorithm:
At first, visit the root, then recursively traversal left sub tree, finally recursively traversal right subtree.

Past-order Traversal:

This method follows the left-right-root-farmula. The root node is visited last, hence the name. First we traverse the left subtree, then the right subtree and finally the root node.



The output of past-order traversal of this tree will be -

D → E → B → F → G → C → A

We start from A, and following past order traversal, we first visit the left subtree past-order. The process goes on untill all the nodes are visited.

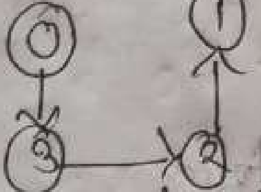
In Algorithm:

At first, recursively traverse left subtree
Then recursively traverse right subtree.
finally visit root node.

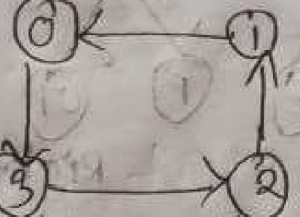
cycle finding

A graph cycle is when there is a loop or circular reference. This can be a series of edges that connect back to an origin vertex. if a graph has a cycle it is a cyclic graph.

To determine if a graph has a cycle, we can traverse the graph and look for a back edge A back edge is one that connects a vertex to an already visited ancestor.



Acyclic graph (no back edges)



cyclic graph (has two back edges)

To detect a cycle in a directed graph we can use depth-first search (with some introduction of local state to tell we if a back edge occurs):

Key differnces:

1. We no longer colour vertices / maintain buckets.

2. We have to make sure to account for the fact that edges are bidirectional so an edge to an ancestor is allowed. If that ancestor is the parent vertex.

3. We only keep track of visited vertices (similar to the grey bucket).

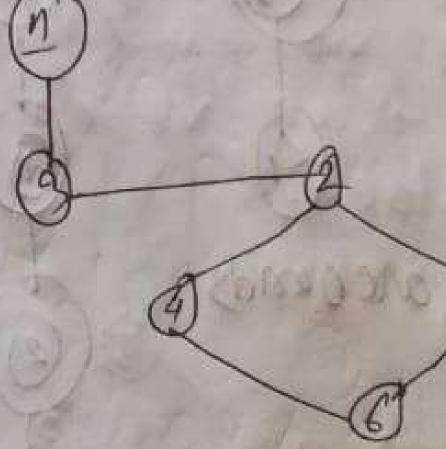
4. When exploring / visiting all descendants of a vertex, if we come across a vertex that has already been visited than we have detected a cycle.

~~ka~~ Time Complexity is O: for an adjacency list. space complexity is $O(V)$. For an adjacency matrix, the time an space

3 Component finding

Component finding can be processed by using both DFS and BFS. when all nodes are connected , we can go from one node to another node. BFS and we will run a loop outside DFS and will see for every node it is visited or not, otherwise will do BFS or DFS on that node. The result will be how many times we BFS or DFS.

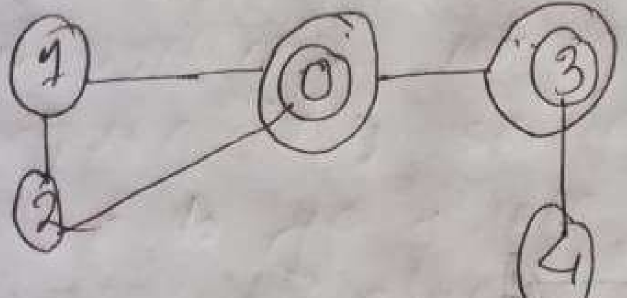
Here is an example of a tree given below. If we start BFS or DFS From 1 then 1-6 all will be visited , so we donot need to do BFS or DFS for 2, 3 - 6 , But 7 is not Connected with any node and is not visited yet. so, we will do BFS or DFS for 7. we have to visit all the node like this.



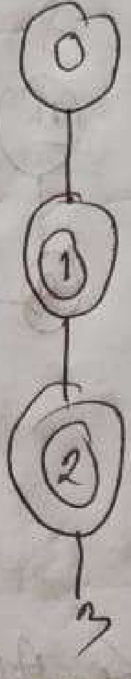
4 Articulation points in a graph

A vertex in an undirected connected graph is an articulation point (or vertex) if removing it (and edges through it) disconnects the graph Articulation points represent vlulneabities in a connected network.

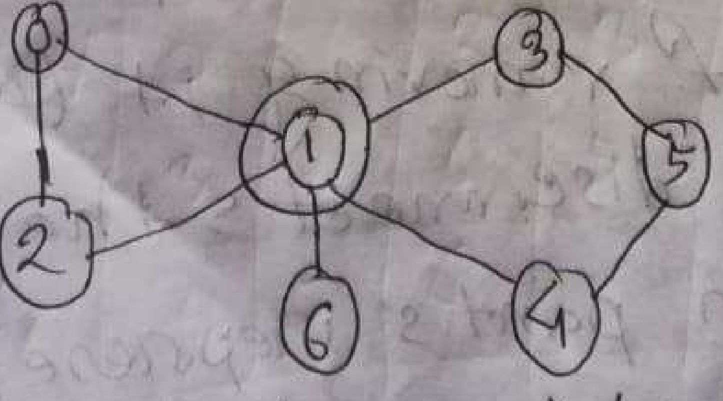
Following are some example graph with articulation points e encircled with red color



Articulation points are 0 and 3



Articulation points are 1&2



Articulation points 1

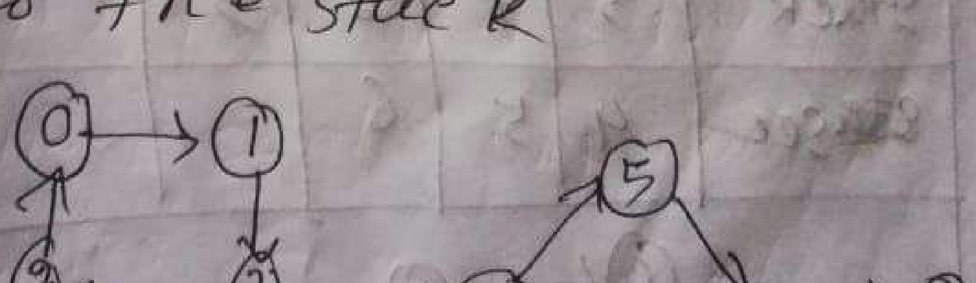
Kosarajus Algorithm:

1. Kosarajus Algorithm is based on the depth-first search algorithm implemented twice.

Three steps as involved.

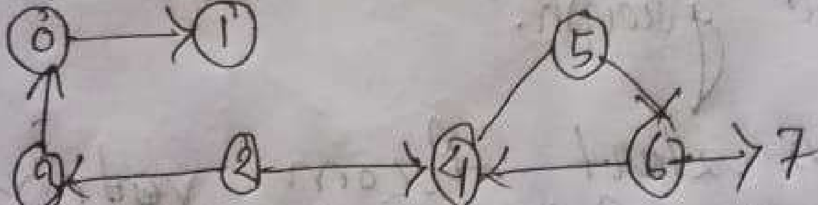
Perfom a depth first search on the whole graph.

Let us start from vertex - 0, visit all of its child veties and mark the visited verti'es as done. If a vertex leads to an already visited vertex, then push this vertex to the stack



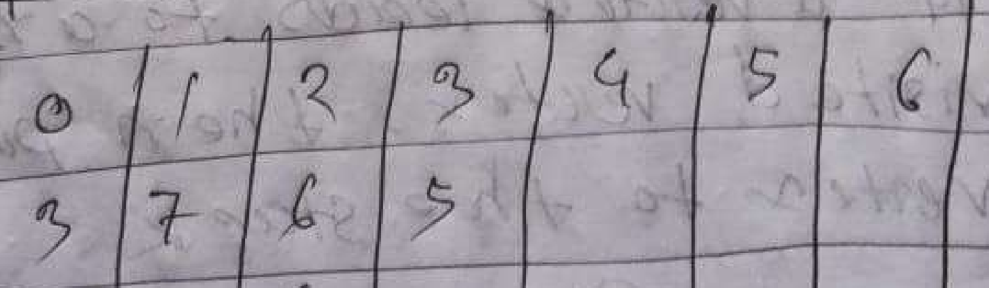
visited	0	1	2	3			
stack	3						

Go to the previous vertex (vertex-2) and visit its child vertices i.e vertex 4, vertex 5, ~~vet~~ vertex-6 and vertex-7 usequentially seinee there is now where to go from vertex-7 push it into the stack.



visited	0	1	2	3	4	5	6	7
stack	3	7						

visited	0	1	2	3	4	5	6
stack	3	7	6	5			
~~sta~~see	4	5	6				



visited	0	1	2	3	4	5	6	
stack								
see	7							

4. Thus ,the strongly Connected Components are