Assignment

Course code: CSE-214/215

Course Title: Algorithms

Submitted To

Subroto Nag Pinku

Department of CSE

Daffodil International University

Submitted by

Name: Mahmodul Hasan

ID: 191-15-12933

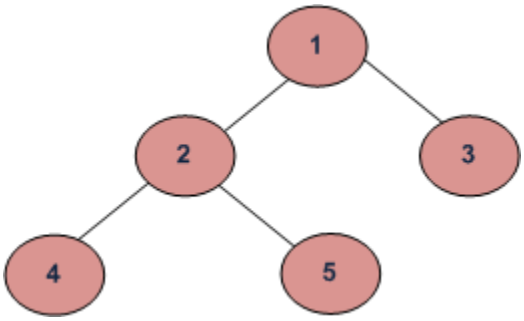
Section: O-14

Department of CSE

Daffodil International University

Date of submission: 08.11.2020

1. Full Tree Traversal



Depth First Traversals:
(a) Inorder (Left, Root, Right) : 4 2 5 1 3
(b) Preorder (Root, Left, Right) : 1 2 4 5 3
(c) Postorder (Left, Right, Root) : 4 5 2 3 1

a. Algorithm Inorder (Tree)

- First traverse the left-subtree.
- Second visit the root.
- Third traverse right-subtree.

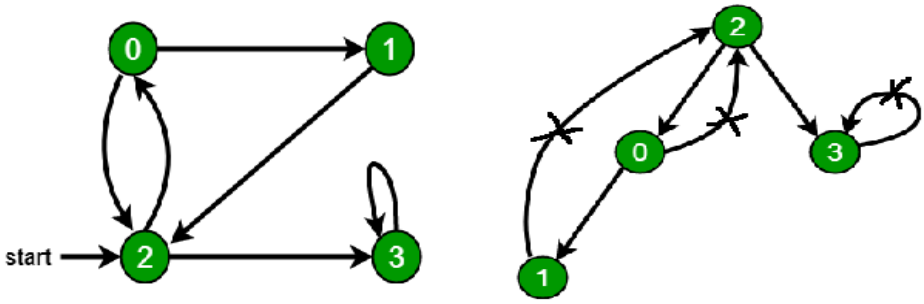
b. Algorithm Preorder (Tree)

- First visit the root.
- Second traverse left-subtree.
- Third traverse right-subtree.

c. Algorithm Postorder (Tree)

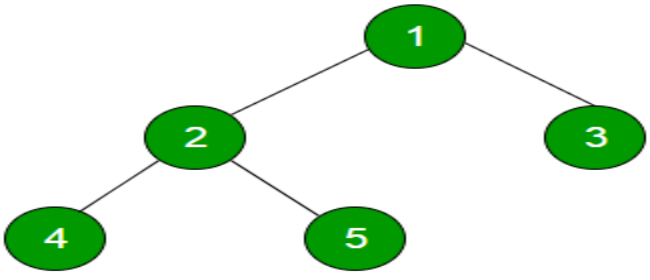
- First traverse the left-subtree.
- Second traverse the right-subtree.
- Third visit the root.

2.Cycle Finding



Depth first traversal can use to find out loop from a graph. If any back edge is present in a graph, then we can say there is a loop. Back edge is an edge that is from node to itself. Above graph has three loops and they are assigned by cross sign. Loops are 2-0-1-2, 2-0-2 and 3-3.
To detect a back edge, keep track of vertices currently in the recursion stack of function for DFS traversal. If a vertex is reached that is already in the recursion stack, then there is a cycle in the tree.

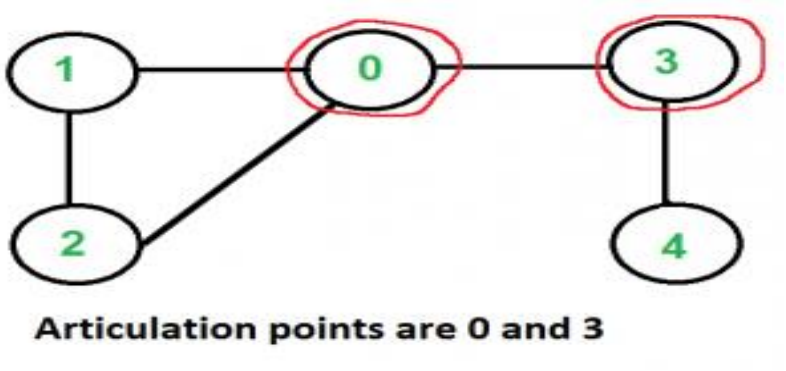
3. Component Finding



Output: 1, 2, 3, 4, 5

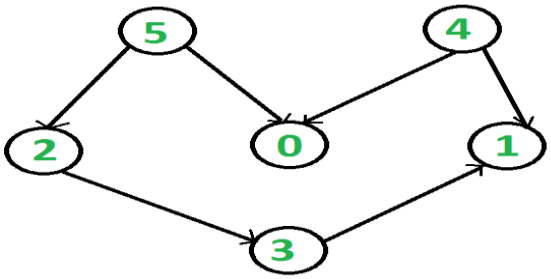
BFS stands for Breath first search is a vertex based technique for finding a shortest path in graph. It uses a Queue data structure which follows first in first out. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

4. Articulation Point Finding



In a graph, a vertex is called an articulation point if removing it and all the edges associated with it results in the increase of the number of connected components in the graph. From above figures, if we remove vertex "0" or "3" then the graph will divide in two part. For vertex "0", they are 1-2 and 3-4.

5. **Topological Sort**



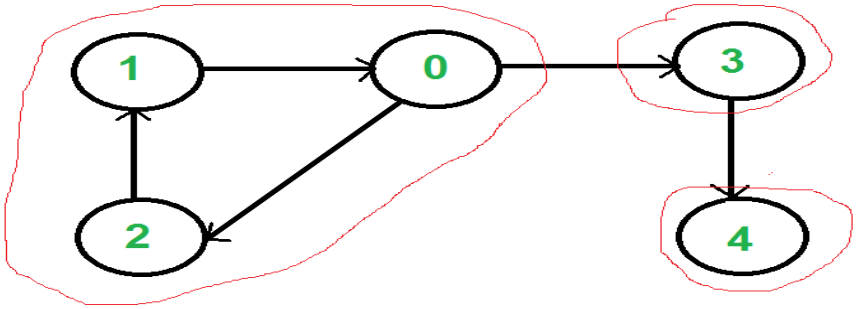
Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge uv, vertex u comes before v in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.

For example, a topological sorting of the following graph is "5 4 2 3 1 0". There can be more than one topological sorting for a graph. For example, another topological

sorting of the following graph is “4 5 2 3 1 0”. The first vertex in topological sorting is always a vertex with in-degree as 0 (a vertex with no incoming edges).

We recommend to first see implementation of DFS here. We can modify DFS to find Topological Sorting of a graph. In DFS, we start from a vertex, we first print it and then recursively call DFS for its adjacent vertices. In topological sorting, we use a temporary stack. We don’t print the vertex immediately, we first recursively call topological sorting for all its adjacent vertices, then push it to a stack. Finally, print contents of stack. Note that a vertex is pushed to stack only when all of its adjacent vertices (and their adjacent vertices and so on) are already in stack.

6. Strongly Connected Components



A directed graph is strongly connected if there is a path between all pairs of vertices. A strongly connected component (SCC) of a directed graph is a maximal strongly connected subgraph. For example, there are 3 SCCs in the following graph.

By using Kosaraju’s algorithm:

1) Create an empty stack ‘S’ and do DFS traversal of a graph. In DFS traversal, after calling recursive DFS for adjacent vertices of a vertex, push the vertex to stack. In the above graph, if we start DFS from vertex 0, we get vertices in stack as 1, 2, 4, 3, 0.
2) Reverse directions of all arcs to obtain the transpose graph.
3) One by one pop a vertex from S while S is not empty. Let the popped vertex be ‘v’. Take v as source and do DFS . The DFS starting from v prints strongly connected component of v. In the above example, we process vertices in order 0, 3, 4, 2, 1 (One by one popped from stack).