



Depth First Search



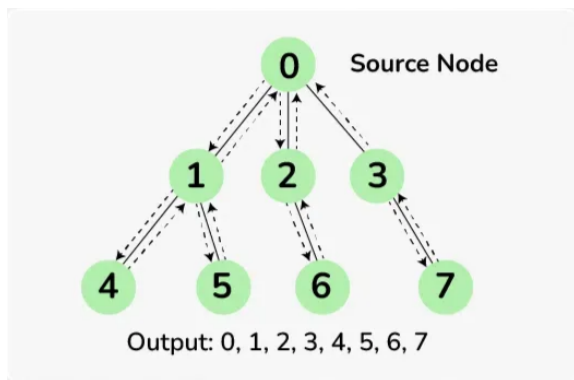
Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph

This algorithm traverses a graph in a depth-ward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

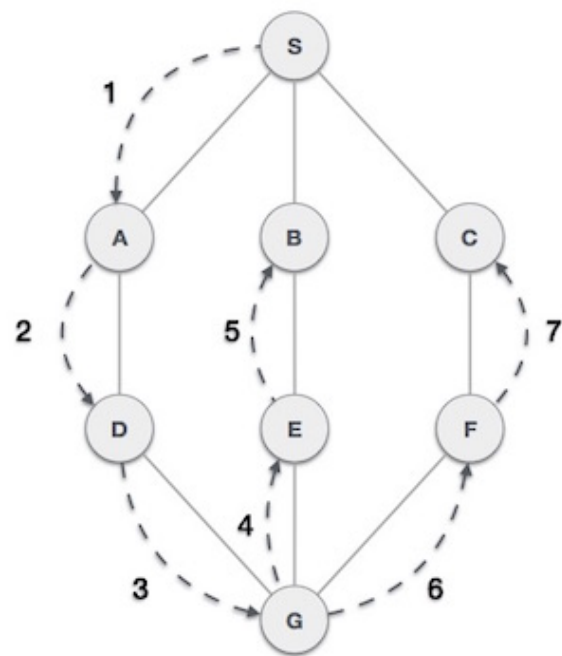
A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.



→ BFS in Tree Structure

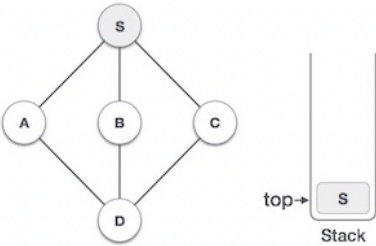
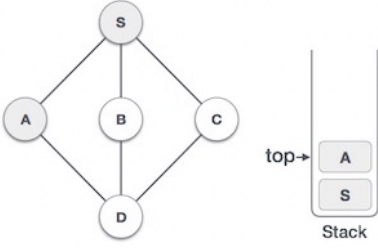
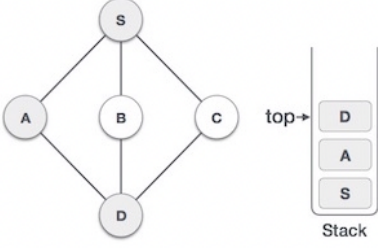
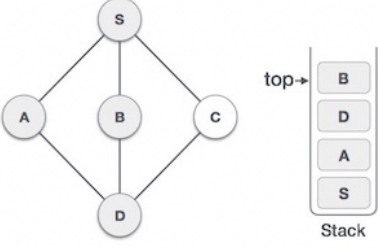
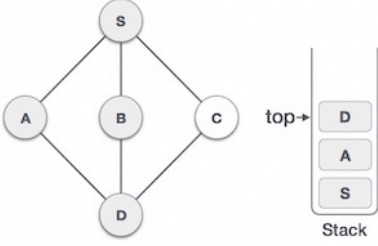


→ BFS in Queue Structure

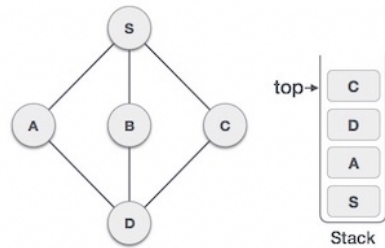
Algorithm steps

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.



2		<p>Mark S as visited and put it onto the stack.</p> <p>Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order.</p>
3		<p>Mark A as visited and put it onto the stack.</p> <p>Explore any unvisited adjacent node from A. Both S and D are adjacent to A but we are concerned for unvisited nodes only.</p>
4		<p>Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in an alphabetical order.</p>
5		<p>We choose B, mark it as visited and put onto the stack. Here B does not have any unvisited adjacent node. So, we pop B from the stack.</p>
6		<p>We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack.</p>

7



Only unvisited adjacent node is from **D** is **C** now. So we visit **C**, mark it as visited and put it onto the stack.

Depth First Search : S A D B C

Pseudocode

```

DFS(G, u)
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G, v)

init() {
    For each u ∈ G
        u.visited = false
    For each u ∈ G
        DFS(G, u)
}
  
```

Analysis

- The time complexity of the DFS algorithm is represented in the form of $O(V + E)$, where V is the number of nodes and E is the number of edges.

Real World Application of Jump Search

- For finding the path
- To test if the graph is bipartite
- For finding the strongly connected components of a graph

4. For detecting cycles in a graph



Java implementation can be found under Implementation_Java folder



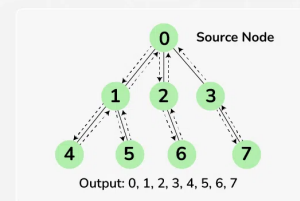
References

DFS traversal of a Tree - GeeksforGeeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive

🔗 <https://www.geeksforgeeks.org/dfs-traversal-of-a-tree-using-recursion/>

Depth First Search



Depth First Search (DFS) Algorithm

Depth First Search (DFS) Algorithm - Depth First Search (DFS) algorithm is a recursive algorithm for searching all the vertices of a graph or tree data structure. This algorithm

🌐 https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm



tutorialspoint

Depth First Search (DFS) Algorithm

Depth First Search is a recursive algorithm for searching all the vertices of a graph or tree data structure. In this tutorial, you will learn about the depth-first search with examples in Java, C, Python, and C++.

🔗 <https://www.programiz.com/dsa/graph-dfs>



Author → Serhat Kumas

<https://www.linkedin.com/in/serhatkumas/>

SerhatKumas - Overview

Computer engineering student who loves coding in different fields instead of focusing on a one spesific area. - SerhatKumas

 <https://github.com/SerhatKumas>

