# RSC TASK REPORT

# TOPIC

## Study Of Depth First Search And Breadth First Search

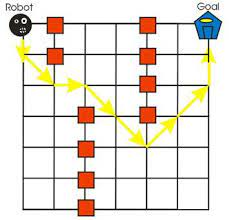
By

Himanshu Kamdi

FY Computer Science

# Introduction

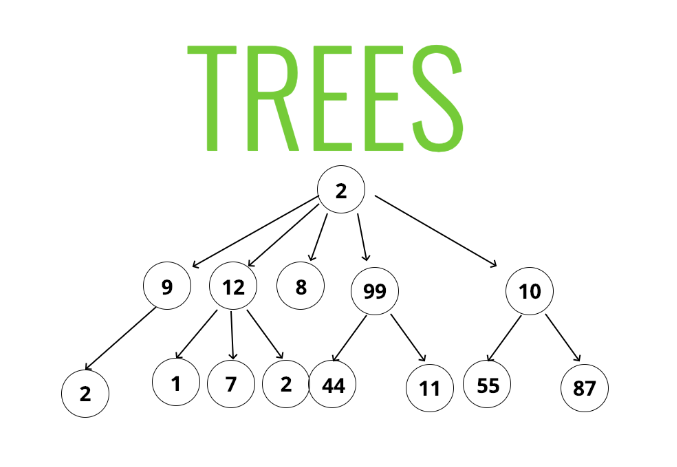
Depth First Search (DFS) and Breadth First Search (BFS) both are Tree search algorithm which are used to traverse through Tree Data structure and find point which satisfies given condition. It is used extensively in Robotics for path finding.



# 

# Tree Data Structure

A tree data structure is defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy.A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type. In the above tree structure, the node contains the name of the employee, so the type of data would be a string.Each node contains some data and the link or reference of other nodes that can be called children.



# Tree Search Algorithms

　　In Computer Science, Tree Search Algorithm also known as Tree Traversal is a form of graph traversal and refers to the process of visiting each node in a tree data structure , exactly once. From Traversing a tree involves iterating over all nodes in some manner. Because from a given node there is more than one possible next node then assuming sequential computation, some nodes must be deferred-stored in some way for later visiting. This is often done via stack(LIFO) or queue(FIFO). As a tree is a self-referential data structure, traversal can be defined by recursion in a natural and clear fashion, in these cases the deferred nodes are stored implicitly in the call stack**.Breadth First Search** (BFS) and **Depth First Search** (DFS) are the common tree search algorithms. Depth first search is easily implemented via a stack, including recursively, while Breadth First Search is easily implemented via a queue.

# Breadth First Search -BFS

　　Breadth-first search is a graph traversal algorithm that starts traversing the graph from the root node and explores all the neighboring nodes. Then, it selects the nearest node and explores all the unexplored nodes. While using BFS for traversal, any node in the graph can be considered as the root node. It is a recursive algorithm to search all the vertices of a tree or graph data structure. BFS puts every vertex of the graph into two categories - visited and non-visited. It selects a single node in a graph and, after that, visits all the nodes adjacent to the selected node.

**Traversing child nodes**

　　A graph can contain cycles, which may bring you to the same node again while traversing the graph. To avoid processing of same node again, use a boolean array which marks the node after it is processed. While visiting the nodes in the layer of a graph, store them in a manner such that you can traverse the corresponding child nodes in a similar order.To make this process easy, use a queue to store the node and mark it as 'visited' until all its neighbors (vertices that are directly connected to it) are marked. The queue follows the First In First Out (FIFO) queuing method, and therefore, the neighbor of the node will be visited in the order in which they were inserted in the node i.e. the node that was inserted first will be visited first, and so on.

**Traversing process**

　　The traversing will start from the source node and push s in queue. s will be marked as 'visited'.

　　First iteration

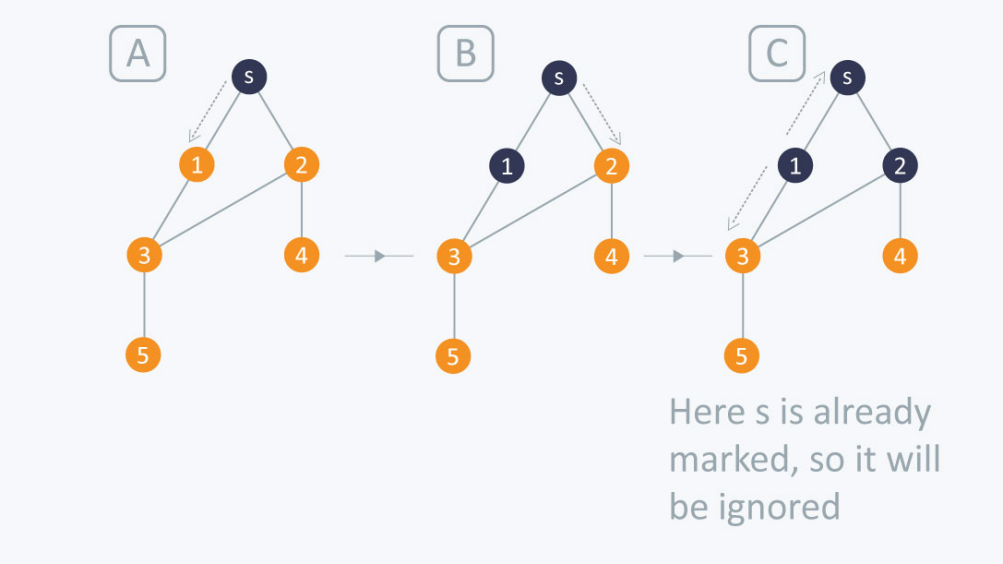
* s will be popped from the queue
* Neighbors of s i.e. 1 and 2 will be traversed
* 1 and 2, which have not been traversed earlier, are traversed. They will be:
  + Pushed in the queue
  + 1 and 2 will be marked as visited

　　Second iteration

* 1 is popped from the queue
* Neighbors of 1 i.e. s and 3 are traversed
* s is ignored because it is marked as 'visited'
* 3, which has not been traversed earlier, is traversed. It is:
  + Pushed in the queue
  + Marked as visited

　　Third iteration

* 2 is popped from the queue
* Neighbors of 2 i.e. s, 3, and 4 are traversed
* 3 and s are ignored because they are marked as 'visited'
* 4, which has not been traversed earlier, is traversed. It is:
  + Pushed in the queue
  + Marked as visited

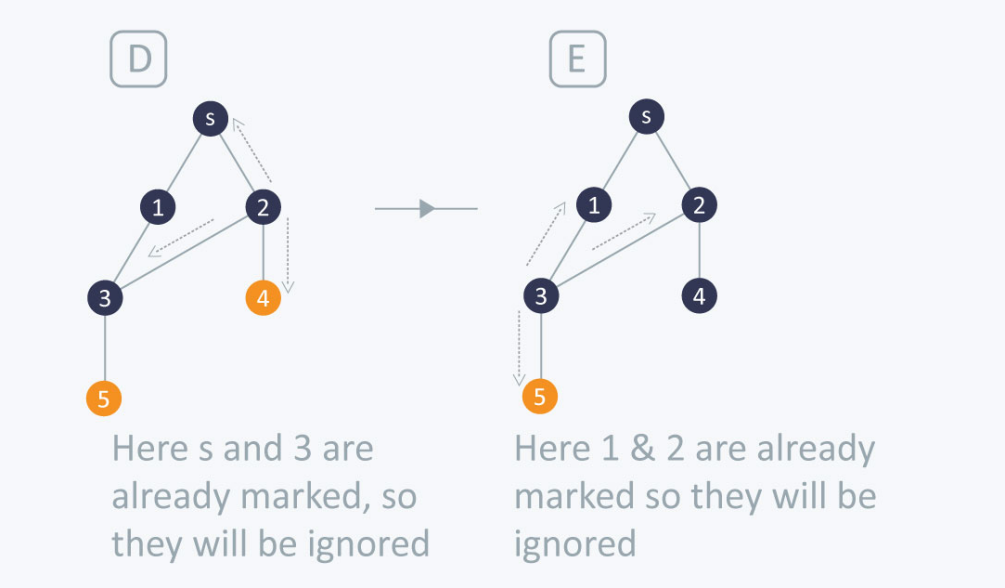


　　Fourth iteration

* 3 is popped from the queue
* Neighbors of 3 i.e. 1, 2, and 5 are traversed
* 1 and 2 are ignored because they are marked as 'visited'
* 5, which has not been traversed earlier, is traversed. It is:
  + Pushed in the queue
  + Marked as visited

　　Fifth iteration

* 4 will be popped from the queue
* Neighbors of 4 i.e. 2 is traversed
* 2 is ignored because it is already marked as 'visited'

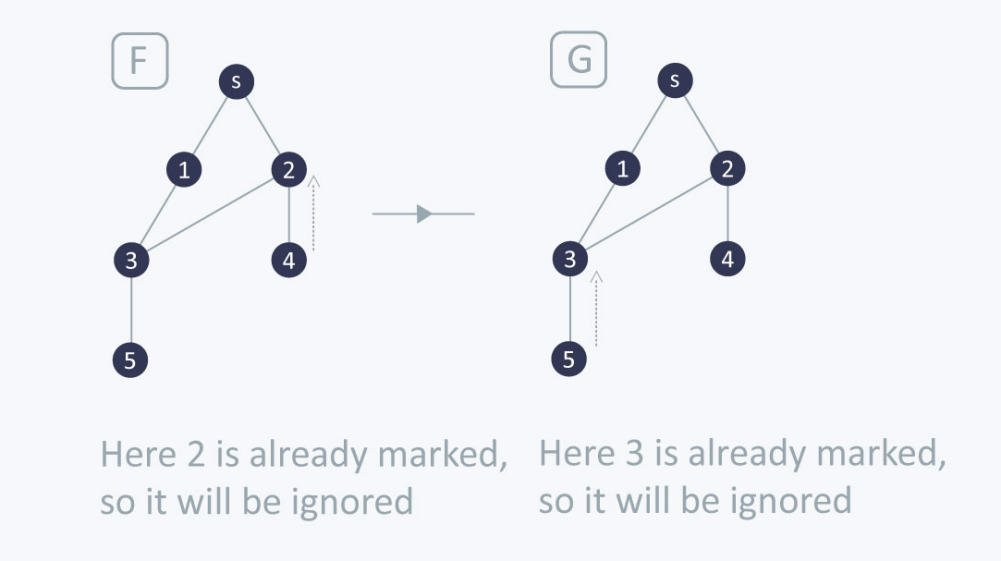


　　Sixth iteration

* 5 is popped from the queue
* Neighbors of 5 i.e. 3 is traversed
* 3 is ignored because it is already marked as 'visited'

　　The queue is empty and it comes out of the loop. All the nodes have been traversed by using BFS.

　　If all the edges in a graph are of the same weight, then BFS can also be used to find the minimum distance between the nodes in a graph.



**Complexity**

　　Time complexity of BFS depends upon the data structure used to represent the graph. The time complexity of BFS algorithm is ****O(V+E)****, since in the worst case, BFS algorithm explores every node and edge. In a graph, the number of vertices is O(V), whereas the number of edges is O(E).

　　The space complexity of BFS can be expressed as ****O(V)****, where V is the number of vertices.

**Applications of BFS algorithm**

　　The applications of breadth-first-algorithm are given as follows -

* As earlier mentioned it is used in robotics to help robots help find path to their goal on place with obstacles
* BFS can be used to find the neighboring locations from a given source location.
* In a peer-to-peer network, BFS algorithm can be used as a traversal method to find all the neighboring nodes. Most torrent clients, such as BitTorrent, uTorrent, etc. employ this process to find "seeds" and "peers" in the network.
* BFS can be used in web crawlers to create web page indexes. It is one of the main algorithms that can be used to index web pages. It starts traversing from the source page and follows the links associated with the page. Here, every web page is considered as a node in the graph.
* BFS is used to determine the shortest path and minimum spanning tree.
* BFS is also used in Cheney's technique to duplicate the garbage collection.
* It can be used in ford-Fulkerson method to compute the maximum flow in a flow network.

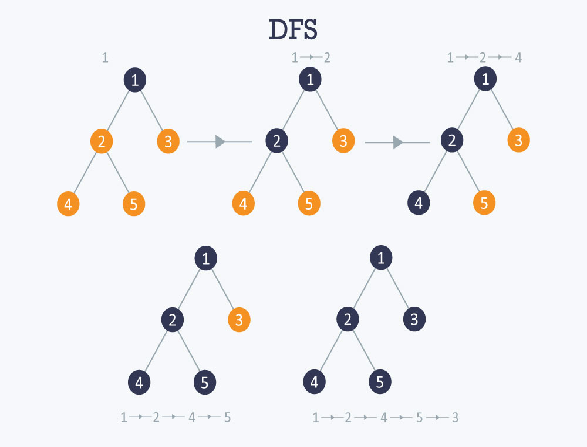
# Depth First Search -DFS

　　Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

**Traversing process**

　　This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:  
Pick a starting node and push all its adjacent nodes into a stack.  
Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.  
Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

　　The following image shows how DFS works.



**Complexity**

**The time complexity of DFS if the entire tree is traversed is O(V) where V is the number of nodes. In the case of a graph, the time complexity is O(V+E) where V is the number of vertexes and E is the number of edges.**

### **Applications of DFS algorithm**

　　The applications of depth-first-algorithm are given as follows :

* As earlier mentioned it is used in robotics to help robots help find path to their goal on place with obstacles
* If we perform DFS on unweighted graph, then it will create minimum spanning tree for all pair shortest path tree
* We can detect cycles in a graph using DFS. If we get one back-edge during BFS, then there must be one cycle.
* Using DFS we can find path between two given vertices u and v.
* We can perform topological sorting is used to scheduling jobs from given dependencies among jobs. Topological sorting can be done using DFS algorithm.
* Using DFS, we can find strongly connected components of a graph. If there is a path from each vertex to every other vertex, that is strongly connected.

# Difference Between BFS and DFS

**Following are the important differences between BFS and DFS.**

| **Sr. No.** | **Key** | **BFS** | **DFS** |
| --- | --- | --- | --- |
| 1 | Definition | BFS, stands for Breadth First Search. | DFS, stands for Depth First Search. |
| 2 | Data structure | BFS uses Queue to find the shortest path. | DFS uses Stack to find the shortest path. |
| 3 | Source | BFS is better when target is closer to Source. | DFS is better when target is far from source. |
| 4 | Suitability for decision tree | As BFS considers all neighbour so it is not suitable for decision tree used in puzzle games. | DFS is more suitable for decision tree. As with one decision, we need to traverse further to augment the decision. If we reach the conclusion, we won. |
| 5 | Speed | BFS is slower than DFS. | DFS is faster than BFS. |
| 6 | Time Complexity | Time Complexity of BFS = O(V+E) where V is vertices and E is edges. | Time Complexity of DFS is also O(V+E) where V is vertices and E is edges. |