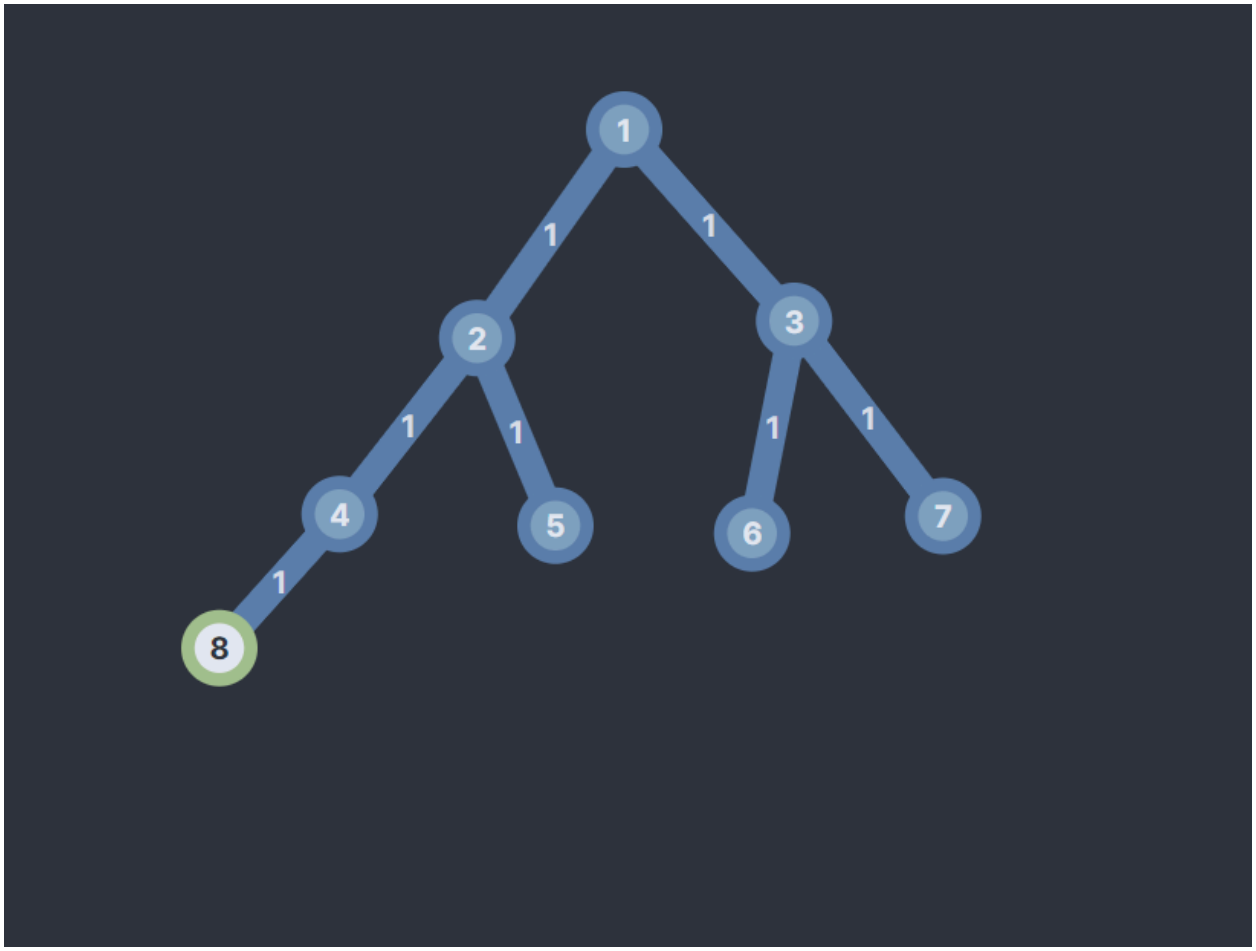
(Depth First Search)

-> Trick of DFS :- Basically you go to each branch from the root node and exhaust the searching there ; come back and do the same process with other branches till they are done.

Graph : -



DFS ORDER : JUST MAKE BOUNDARY OF THE GRAPH/TREE AND YOU WILL GET THE DFS OF GRAPH.

DFS is a way to travel the graph.

ORDER :- [1 2 4 8 5 3 6 7]

TO UNDERSTAND THE SPACE DIFFERENCES OF BFS VS DFS :- https://stackoverflow.com/questions/23477856/why-does-a-breadth-first-search-use-more-memory-than-depth-first

C++ https://ideone.com/0Zj2S8
Java https://ideone.com/DovpoW
Python https://ideone.com/5MbskC

General DFS Algorithm :->

You take the source node (1) ; you print it

-> Then you check and exhaust all the branches connected to source node ; and do the DFS there as well(assuming that again the top node of this branch is source node and do the same thing)

-> As you are repeating the same process again and again its using recursion.

-> Recursion uses secret stack space memory jii

for(auto u : G[1]){

u-> 2 | 3

}

Flow of DFS :- >

DFS(1)

DFS(2) → used[2] = 1 (it has now been visited)

DFS(4)

DFS (8)

As node 4 is fully exhausted(searched)

DFS(5)

As node 2 is fully exhausted(searched)

DFS(3)

DFS (6)

As node 6 is fully exhausted(searched)

DFS(7)

-> Secret Stack :->
All nodes which we were traveling got stored in the stack and popped out as per need.

Stack = {}

Stack = {1,2} (2 is the topmost node of the stack)

Stack = {1,2,4}

Stack = {1,2,4,8}

Stack = { 1,2,4}

Stack = { 1,2 }

Stack = {1,2,5}

Stack = {1,2}
(now it is confirmed that visited all the children of node so now u can successfully kick it out ; you can only kick out the node from stack if all its children and that node itself are visited)

Stack = {1}

Stack = {1 3}

Stack = {1 3 6}

Stack = {1 3}

Stack = { 1 3 7}

Stack = {1 3}

You can now kick out 3 because 3 and all its children have been visited.

Stack = {1 }

Stack = {}

You kick out 1 because all nodes have been visited.

Time Complexity : O(Nodes + Edges)

Space Complexity :- Maximum depth of the graph from the source node(that's how large the stack can get)

Maximum depth of our graph was : - 4

O(Max Depth of graph) - O(Diameter of graph)

→ 1—2—3—--4—--5—----------n ; depth :- N

SC :- O(N)