**BFS (Breadth First Search):**

It kind of signifies level wise traversal, here we would be provided and that would be considered as level 0.

After traversing the node at level 0 we would move ahead to the next level and continue till we reach the last node.

The image given below shoes the result for BFS where starting node was 1

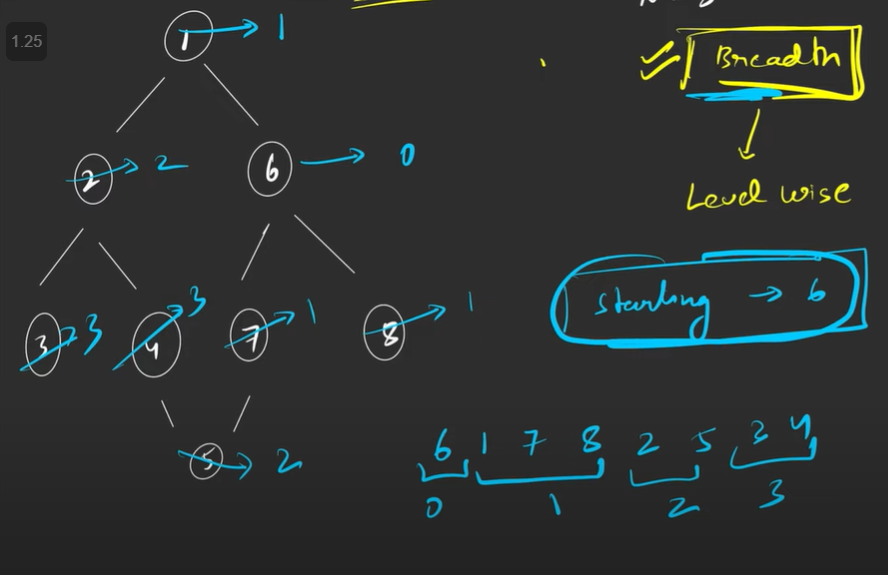
* After visiting 1
* We went to next level which is 2 and 6
* After 2 and 6 we would move on to 3 4 7 and 8

A picture containing diagram

Description automatically generated

**What if we change the starting point**

If 6 becomes our starting point, we would need to follow the steps given below.



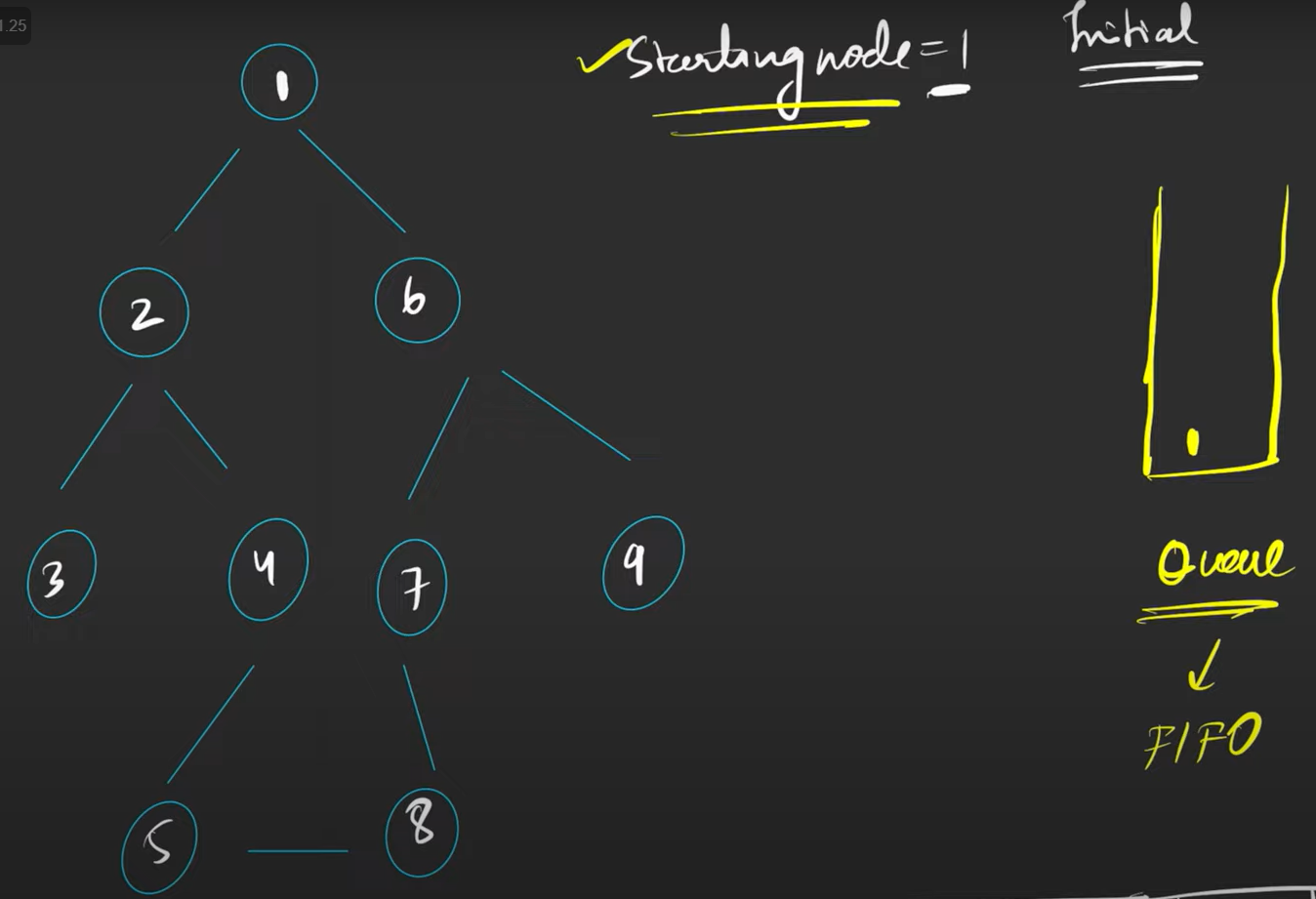
6 would be considered as level 0.

1, 7 and 8 would be considered as level 1.

2 and 5 would be considered as level 2.

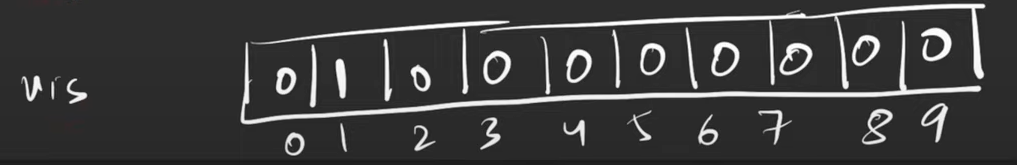
3 and 4 would be considered as level 3.

**Algorithm**



Initially we would be provided with the starting node.

We will have a Queue data structures which follows FIFO (First In First Out).



A visited array as shown above will mark the node which has been touched or we can say visited as 1.

So if a node’s index in visited array is marked as 1 it means that the element has already been into Queue.

**Note: What will do is that whatever has been into Queue, will start taking it out and we will take out elements till our queue is not empty.**

We take out 1 from the queue and we print it as shown in the picture given below.

A picture containing diagram

Description automatically generated

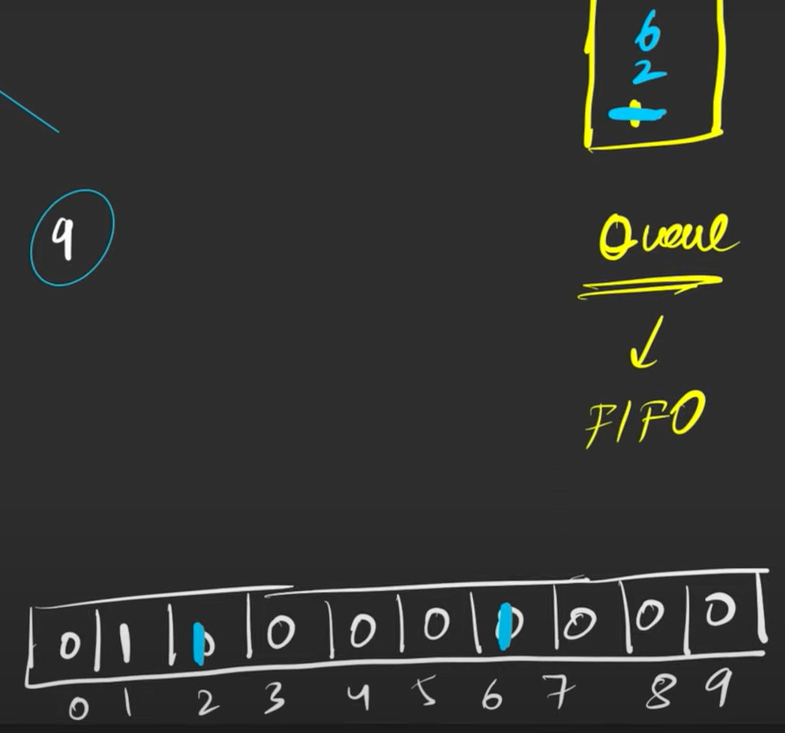
We have stored our graph data structure in form of some adjacency list and that list looks like:

A screenshot of a computer

Description automatically generated with medium confidence

Now since we took out 1 from the queue which was the starting point.

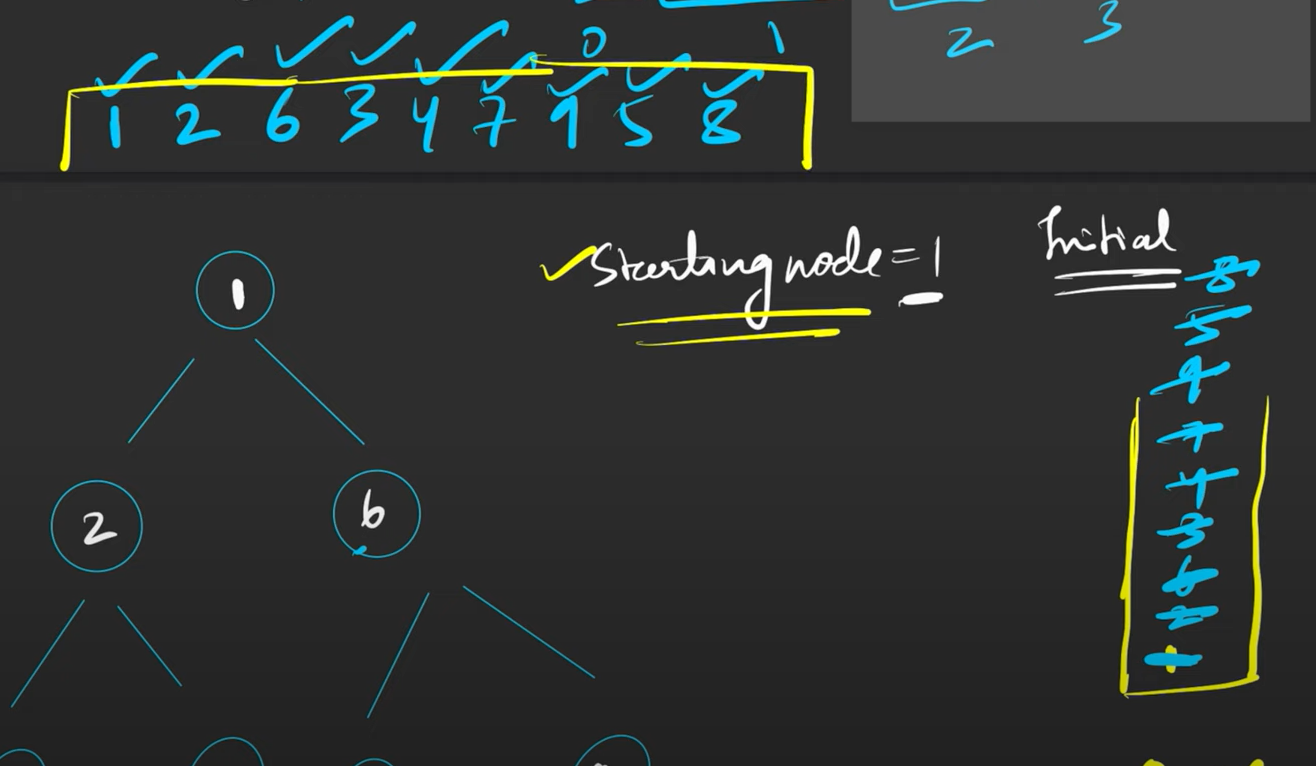
We will take the neighbours of 1 and put it into the queue and mark it as 1 which means visited for the visited array.



Now take out the next node which is 2 and store its unvisited neighbours into the queue, now two has 3 neighbours which are 1,3 and 4. Since 1 is already visited, we would not put it into the queue.

We will put 3 and 4 into the queue.

At the end we would get our BFS traversal as shown in below picture.



**Time complexity**: Now for every node, we would push it into the queue and run BFS for all its degrees that if a node is pushed into the Queue, we are going to run BFS for all its neighbours.

Therefore, our time complexity comes out to be O(N) + O(2E) where 2E is total number of degrees in the graph as Each edge connects two nodes therefore total number of degrees in a graph would be 2E.