BFS

-Graph traversal: means “visiting every vertex & edge exactly once” in a well defined order.

- During the traversal we should “track which vertices we’ve visited” by marking them. A graph can contain cycles! ==> So use a boolean array which marks the node.
- Start by traversing from a selected node(source) & traverse the graph laywise thus “exploring the neighbor nodes” <==> nodes directly connected to the source.
- Then you must move towards the next level neighbor nodes.

- **Algorithm idea**:

(1) First move horizontally and visit all the nodes of the current layer.(Distance 1 from the source node)
(2) Move to the next layer.
We must traverse all the nodes in a layer before we move to nodes of the next layer.
(3) While visiting the nodes in a layer of a graph G(V,E) ,store them in a way such that you can traverse the child nodes in a similar order.
(4) Use a **queue** to sore the node & mark as “visited” until all the neighbors are marked.
(5) Queue: FIFO logic. ==> The node which was inserted first will be visited first.

- **Pseudo-Code:**

```
BFS(G,s): // G: the graph, s: the source node
    // Let Q the queue
    Q.enqueue(s); // insert s in Q
    Mark s as visited // use a bool array
    while(Q not empty) {
        // Remove that vertex from Q, whose neighbor will be visited now
        v = Q.dequeue();

        // Processing all the neighbors of v
        for(all neighbors w of v in G) {
            if(w is not visited) {
                Q.enqueue(w); // stores w in Q to further visit it’s neighbor
                Mark w as visited // detect cycle ==> no backtracking(mainly for undirected)
            }
        }
    }
```

Application: Test if a G is connected

BFS: Traversing along (use a queue)
DFS: Traversing downwards (use a stack)