```java
for(int nbr: graph.get(node)){
    if(visited[nbr] == false){
        //if nbr is  unvisited
        dfs(graph,visited,nbr);
    }
}
```

{3, 5, 6}

int[] arr = new int[n]
String[] arr = new String[n]

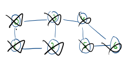
ArrayList<Integer> arr = new ...
int[][] arr = new int[n][m];
ArrayList<ArrayList<Integer>> a..
ArrayList<Integer>[] arr = new ...

[" " / " " / " " / " "]

ArrayList<Integer>[] arr = ...

ArrayList<Integer>[] arr = ... ⇒ Array of String
Array of Arraylist

[
0: { }
1: { }
2: { }
3: { }
]



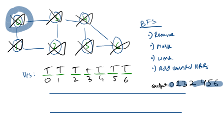
0: 1, 3
1: 0, 2
2: 1, 3
3: 0, 2, 4
4: 3, 5, 6
5: 6, 4.
6: 4, 5

T T T T T T T
0 1 2 3 4 5 6

Output: 0 1 2 3 4 5 6

4
3
2
1
0

DFS
·) Mark visited
·) Work
·) Visit unvisited nbrs




```java
public static void DFSTraversal(List<List<Integer>> edges, int n
    //Write your code here
    ArrayList<ArrayList<Integer>> graph = new ArrayList<>();
    //i have to add n empty arraylists inside graph
    for(int i = 0;i<n;i++){
        graph.add(new ArrayList<Integer>());
    }
    //its time to add edges
    for(List<Integer> edge: edges){
        int u = edge.get(0);
        int v = edge.get(1);

        graph.get(u).add(v);
        graph.get(v).add(u);
    }
}
```

4	4
1	3
0	2
2	3
1	2

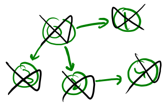
{
0: { }
1: {3 }
2: { }
3: {1 }
}

{ {1, 3}
{0, 2}
{2, 3}
{1, 2}
}

{0, 2}

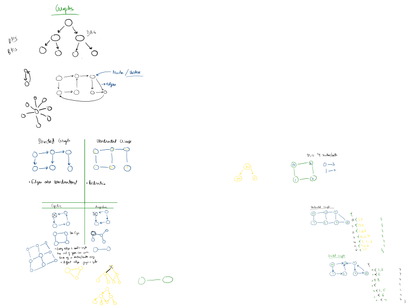
```java
// your code here
boolean[] visited = new  boolean[vertices];
Queue<Integer> q = new  LinkedList<>();
q.add(x);
//bfs starts
while(q.size()>0){
    //Remove
    int curr = q.remove();
    //mark
    if(visited[curr] == true) continue;
    visited[curr] = true;
    //work
    System.out.print(curr+" ");
    //Add unvisited nbrs
    for(int nbr: graph[curr]){//graph is an array of
        if(!visited[nbr]) q.add(nbr);
    }
}
```

0 1 2 3 4

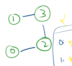


[
0: { 1, 2, 3 }
1: { }
2: { 4 }
3: { }
4: { }
]

[null null null]



n
4 4 → e
1 3
0 2
2 3
1 2
edge



{
0: {2 }
1: {3, 2 }
2: {0, 3, 1 }
3: {1, 2 }
}

{
{ }
{ }
}