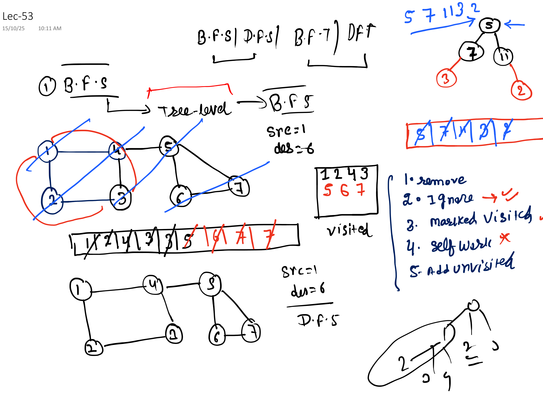
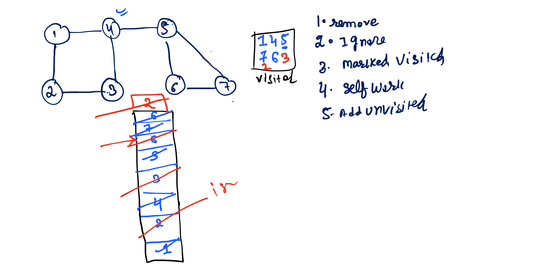
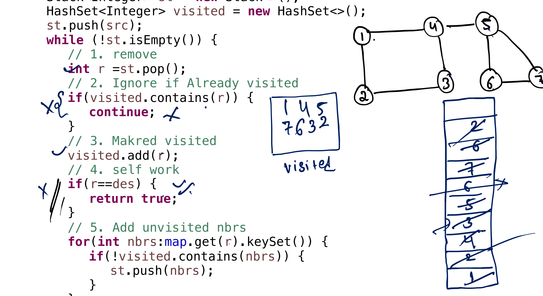
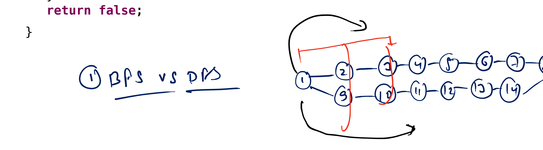
Lec-53

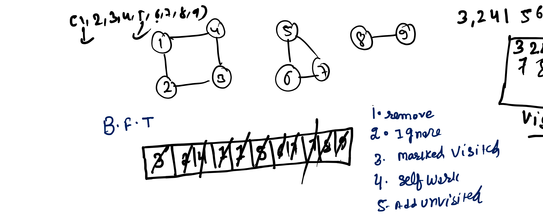




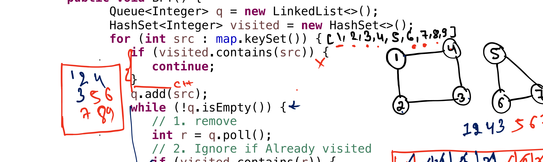

```
public boolean DFS(int src, int des) {
      Stack<Integer> st = new Stack<>();
      HashSet<Integer> visited = new HashSet<>();
      st.push(src);
      while (!st.isEmpty()) {
         // 1. remove
         int r =st.pop();
         // 2. Ignore if Already visited
         if(visited.contains(r)) {
            continue;
         }
         // 3. Makred visited
         visited.add(r);
         // 4. self work
         if(r==des) {
            return true;
         }
         // 5. Add unvisited nbrs
         for(int nbrs:map.get(r).keySet()) {
            if(!visited.contains(nbrs)) {
               st.push(nbrs);
            }
         }
      }
      return false;
   }
```







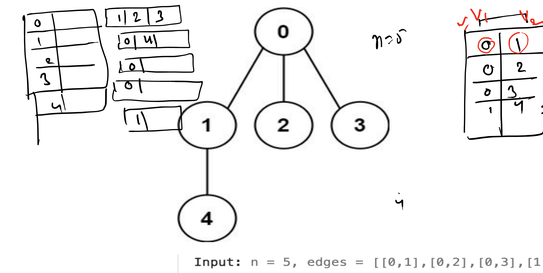

```
public void BFT() {
      Queue<Integer> q = new LinkedList<>();
      HashSet<Integer> visited = new HashSet<>();
      for (int src : map.keySet()) {
         if (visited.contains(src)) {
            continue;
         }
         q.add(src);
         while (!q.isEmpty()) {
            // 1. remove
            int r = q.poll();
            // 2. Ignore if Already visited
            if (visited.contains(r)) {
               continue;
            }
            // 3. Makred visited
            visited.add(r);
            // 4. self work
            System.out.print(r + " ");
            // 5. Add unvisited nbrs
            for (int nbrs : map.get(r).keySet()) {
               if (!visited.contains(nbrs)) {
                  q.add(nbrs);
               }
            }
         }
      }
      System.out.println();

   }
```



You have a graph of `n` nodes labeled from `0` to `n - 1`. You are given an integer n and a list of `edges` where `edges[i] = [a_i, b_i]` indicates that there is an undirected edge between nodes `a_i` and `b_i` in the graph.

Return `true` *if the edges of the given graph make up a valid tree, and* `false` *otherwise.*

Example 1:




```
Input: n = 5, edges = [[0,1],[0,2],[0,3],[1,4]]
Output: true
```