1. import java.util.\*;

public class GraphTraversal {

// BFS Traversal Function

public static List<Integer> bfsOfGraph(int V, ArrayList<ArrayList<Integer>> adj) {

List<Integer> bfs = new ArrayList<>();

boolean[] visited = new boolean[V];

Queue<Integer> q = new LinkedList<>();

q.add(0); // Start from node 0

visited[0] = true;

while (!q.isEmpty()) {

int node = q.poll();

bfs.add(node);

for (int neighbor : adj.get(node)) {

if (!visited[neighbor]) {

visited[neighbor] = true;

q.add(neighbor);

}

}

}

return bfs;

}

}

2.

import java.util.\*;

public class GraphTraversal {

public static List<Integer> dfsOfGraph(int V, ArrayList<ArrayList<Integer>> adj) {

List<Integer> dfs = new ArrayList<>();

boolean[] visited = new boolean[V];

dfsHelper(0, adj, visited, dfs);

return dfs;

}

private static void dfsHelper(int node, ArrayList<ArrayList<Integer>> adj,

boolean[] visited, List<Integer> dfs) {

visited[node] = true;

dfs.add(node);

for (int neighbor : adj.get(node)) {

if (!visited[neighbor]) {

dfsHelper(neighbor, adj, visited, dfs);

}

}

}

}