```
package Love_Babbar;

public class Graph {
    // Create a Graph, print it
    // Implement BFS algorithm
    class Solution {
        public ArrayList<Integer> bfsOfGraph(int V, ArrayList<ArrayList<Integer>> adj) {
            ArrayList<Integer> bfs = new ArrayList<>();
            boolean vis[] = new boolean[V];
            for (int i = 0; i < V; i++) {
                vis[i] = false;
            }
            Queue<Integer> q = new LinkedList<>();
            q.add(src);
            vis[src] = true;
            while (!q.isEmpty()) {
                Integer node = q.poll();
                bfs.add(node);
                for (Integer it : adj.get(node)) {
                    if (vis[it] == false) {
                        vis[it] = true;
                        q.add(it);
                    }
                }
            }
            return bfs;
        }
    }
    // Implement DFS algorithm
    class Solution {
        public void dfs(int src, boolean[] vis, ArrayList<ArrayList<Integer>> adj,
ArrayList<Integer> dfs) {
            vis[src] = true;
            dfs.add(src);
            for (Integer it : adj.get(src)) {
                if (visited[it] == false) {
                    dfs(it, vis, adj, dfs);
                }
            }
        }

        public ArrayList<Integer> dfsOfGraph(int V, ArrayList<ArrayList<Integer>> adj) {
            ArrayList<Integer> dfs = new ArrayList<>();
            boolean vis[] = new boolean[V];
            for (int i = 0; i < V; i++) {
                vis[i] = false;
            }
            visited[src] = true;
            dfs(src, vis, adj, dfs);
```

```
            return dfs;
        }
    }
    // Detect Cycle in Directed Graph using BFS/DFS Algo
    // Detect Cycle in UnDirected Graph using BFS/DFS Algo
    // Search in a Maze
    // Minimum Step by Knight
    // Flood fill algo
    // Clone a graph
    // Making wired Connections
    // Word Ladder
    // Dijkstra algo
    // Implement Topological Sort
    // Minimum time taken by each job to be completed given by a Directed Acyclic
    // Graph
    // Find whether it is possible to finish all tasks or not from given
    // dependencies
    // Find the no. of Islands
    // Given a sorted Dictionary of an Alien Language, find order of characters
    // Implement Kruksal'sAlgorithm
    // Implement Prim's Algorithm
    // Total no. of Spanning tree in a graph
    // Implement Bellman Ford Algorithm
    // Implement Floyd warshall Algorithm
    // Travelling Salesman Problem
    // Graph Colouring Problem
    // Snake and Ladders Problem
    // Find bridge in a graph
    // Count Strongly connected Components(Kosaraju Algo)
    // Check whether a graph is Bipartite or Not
    // Detect Negative cycle in a graph
    // Longest path in a Directed Acyclic Graph
    // Journey to the Moon
    // Cheapest Flights Within K Stops
    // Oliver and the Game
    // Water Jug problem using BFS
    // Find if there is a path of more thank length from a source
    // M-Colouring Problem
    // Minimum edges to reverse to make path from source to destination
    // Paths to travel each nodes using each edge(Seven Bridges)
    // Vertex Cover Problem
    // Chinese Postman or Route Inspection
    // Number of Triangles in a Directed and Undirected Graph
    // Minimise the cashflow among a given set of friends who have borrowed money
    // from each other
    // Two Clique Problem
}
```