```
import java.util.List;
import java.util.LinkedList;
import java.util.Queue;
import java.util.ArrayDeque;
import java.util.Arrays;

public class new_file {

    public static void main(String[] args) {
        List<Edge> edges = Arrays.asList(
            new Edge(2,1), new Edge(2,3), new Edge(3,4), new Edge(3,5)
        );
        Graph g = new Graph(edges);
        int size = 0;
        for (Edge edge : edges) {
            size = Integer.max(size, Integer.max(edge.src, edge.dest));
        }
        boolean[] vis = new boolean[size + 1];
        bfs_traversal(g, 2, vis);
        Arrays.fill(vis, false);
        System.out.println();
        System.out.println(get_path_length(g, 1, 5, vis));

    }


    public static int get_path_length(Graph g, int src, int dest, boolean[] vis) {
        int total_cost = 0;
        Queue<Integer> q = new ArrayDeque<>();
        q.add(src);
        vis[src] = true;
        while (!q.isEmpty()) {
            src = q.poll();
            if (src == dest)
                break;
            for (var adj_nodes : g.adj.get(src)) {
                total_cost += adj_nodes;
                if (!vis[adj_nodes]) {
                    vis[adj_nodes] = true;
                    q.add(adj_nodes);
                }
            }
        }
        return total_cost;
    }


    public static void bfs_traversal(Graph g, int src, boolean[] vis) {
        Queue<Integer> q = new ArrayDeque<>();
```

```
        q.add(src);
        vis[src] = true;
        while (!q.isEmpty()) {
            src = q.poll();
            System.out.print(src + " ");
            for (var adj_nodes : g.adj.get(src)) {
                if (!vis[adj_nodes]) {
                    vis[adj_nodes] = true;
                    q.add(adj_nodes);
                }
            }
        }
    }


    static class Edge {
        int src, dest;

        public Edge(int src, int dest) {
            this.src = src;
            this.dest = dest;
        }
    }


    static class Graph {
        List<List<Integer>> adj;

        public Graph(List<Edge> edges) {
            adj = new LinkedList<>();
            int max = 0;
            for (Edge edge : edges) {
                max = Integer.max(max, Integer.max(edge.src, edge.dest));
            }
            for (int i = 0; i < max + 1; i++)
                adj.add(new LinkedList<>());
            for (Edge edge : edges)
                adj.get(edge.src).add(edge.dest);
        }

    }


}
```