# 模版·BFS(求最短路径)

```
class Node{
    int val;
    List<Node> neighbors;
    int distance;
    Node previousNode;

    public Node(int val){
        this.val=val;
        this.neighbors=new ArrayList<>();
        this.distance=Integer.MAX_VALUE;
        this.previousNode=null;
    }

    public void addNeighbor(Node neighbor){
        this.neighbors.add(neighbor);
    }
}

class Graph{
    private Map<Integer,Node> nodes;

    public Graph(int n){
        this.nodes=new HashMap<>();
        for(int i=1;i<=n;i++){
            nodes.put(i,new Node(i));
        }
    }

    public void addEdge(int u,int v){
        Node nodeU=nodes.get(u);
```

```
        Node nodeV=nodes.get(v);
        nodeU.addNeighbor(nodeV);
        nodeV.addNeighbor(nodeU);
    }
    public void findShortestPath(int start,int target){
        Queue<Node> queue=new LinkedList<>();
        Set<Node> visited=new HashSet<>();

        Node startNode=nodes.get(start);
        startNode.distance=0;
        queue.add(startNode);
        visited.add(startNode);

        while(!queue.isEmpty()){
            Node currentNode=queue.poll();

            if(currentNode.val==target){
                printShortestPath(currentNode);
                return;
            }

            for(Node neighbor:currentNode.neighbors){
                if(!visited.contains(neighbor)){
                    neighbor.distance=currentNode.distance+1;
                    neighbor.previousNode=currentNode;
                    queue.add(neighbor);
                    visited.add(neighbor);
                }
            }
        }

        System.out.println("Pathnotfound.");
    }
```

```
    private void printShortestPath(Node node){
        if(node==null){
            return;
        }

        printShortestPath(node.previousNode);
        System.out.print(node.val+"");
    }
}

public class BreadthFirstSearch{
    public static void main(String[] args){
        int n=5;
        Graph graph=new Graph(n);

        graph.addEdge(1,2);
        graph.addEdge(1,3);
        graph.addEdge(2,4);
        graph.addEdge(3,4);
        graph.addEdge(4,5);

        graph.findShortestPath(1,5);
    }
}
```