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

/**
 * this is not an good approach to find the motherVertex
 * this problem can effeciently solved using Kosaraju's
 * Strongly Connected Component Algo.
 * Coded in Kosraju.java in { .\Compete\javaCodes }
 */

public class motherVertex {

    public static void main(String[] args) {
        List<Edge> edges = Arrays.asList(
            new Edge(1,0), new Edge(0,2), new Edge(2,1),
            new Edge(0,3), new Edge(3,4)
        );
        Graph g = new Graph(edges);
        Set<Integer> s = new HashSet<>();
        var max = 0;

        for (Edge edge : edges)
            max = Integer.max(max, Integer.max(edge.dest, edge.src));

        boolean[] vis = new boolean[max + 1];
        set_the_set(g, s, 0, vis);
        Set<Integer> ans = new HashSet<>();
        getMotherVertex(g, edges, ans, s, vis);
        System.out.println(ans);
    }


    private static void set_the_set(Graph g, Set<Integer> s, int i, boolean[] vis) {
        Queue<Integer> q = new ArrayDeque<>();
        q.add(i);
        vis[i] = true;

        while (!q.isEmpty()) {
            i = q.poll();
            s.add(i);
            for (var adj_nodes : g.adj.get(i)) {
                if (!vis[adj_nodes]) {
                    vis[adj_nodes] = true;
                    q.add(adj_nodes);
```

```
            }
        }
    }
}


    public static void getMotherVertex(Graph g,List<Edge> edges, Set<Integer> ans,
Set<Integer> s, boolean[] vis) {
        Set<Integer> tmp_set = new HashSet<>();
        for (Edge edge : edges) {
            Arrays.fill(vis, false);
            set_the_set(g, tmp_set, edge.src, vis);
            if (tmp_set.equals(s))
                ans.add(edge.src);
        }
    }



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


}
```