```java
package test;

import java.util.HashSet;
import java.util.Iterator;

class DirNodes
{
	int NodeId;
	int [] directNodes;
}
public class BFS {
	int len;
	// represent edges
	DirNodes [] ConnectedNode = new DirNodes[len];
	// whether in set R_i()  yet
	int [] PathNr = new int[len];
	HashSet<Integer> Rset;
	int result;
	public void GetPathNr(int v, int w)
	{
		if(v == w)
			result = 1;
		else{
			//**********************
			//**init ConnectedNode**
			//**********************

			for(int i = 0; i < len; i ++)
				PathNr[i] = -1;
			PathNr[v] = 1;
			Rset = new HashSet<Integer>();
			Rset.add(v);
			ReccursivelySearch(w, 0);
		}

	}
	private void ReccursivelySearch(int w, int level) {
		// TODO Auto-generated method stub
		HashSet<Integer> NextRset =
				new HashSet<Integer>();
		boolean findW = false;
		Iterator<Integer> it = Rset.iterator();
		while(it.hasNext()){
			int nodeid = it.next();
			for(int nextNodeId : ConnectedNode[nodeid].directNodes)
			{
				if(nextNodeId == w)
					findW = true;
				if(PathNr[nextNodeId] < 0){
					NextRset.add(nextNodeId);
					PathNr[nextNodeId] = 0;
				}
				if(NextRset.contains(nextNodeId))
					PathNr[nextNodeId] += PathNr[nodeid];
			}
		}
		if(findW)
			result = PathNr[w];
		else if(NextRset.size() < 1){
				result = 0;
		}else{
			Rset = NextRset;
			ReccursivelySearch(w, level + 1);
		}
	}
}
```