Java code :

package psn.learn.graphic;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Graphic {  
  
 */\*\*  
 \* 通过矩阵数据检查该矩阵代表的图是否是连通。  
 \** ***@param*** *values  
 \** ***@return*** *\*/* public boolean checkGraphicConnect(int[][] values){  
 if(values == null || values.length <1){  
 System.*out*.println(**"input values is null; not graphic"**);  
 return false;  
 }  
  
 this.printGraphic(values);  
  
 int nodeSize = values.length;  
 int[] tmp = new int[nodeSize];  
 initSetZero(tmp);  
  
  
 checkByDFS(values,0,tmp);  
  
 List<String> sigleNodes = new ArrayList<String>();  
 for(int i=0 ;i<values.length;i++){  
 if(tmp[i] < 1){  
 sigleNodes.add(String.*valueOf*(i+1));  
 }  
 }  
  
 if(!sigleNodes.isEmpty()){  
 System.*out*.println(**"No."** + Arrays.*toString*(sigleNodes.toArray(new String[sigleNodes.size()])) +**" Node is single; not graphic"**);  
 return false;  
 } else {  
 System.*out*.println(**"all node is connect;"**);  
 return true;  
 }  
 }  
  
 */\*\*  
 \* 初始化节点的连通性的值；默认所有节点的连通性都是0，表示不连通；  
 \** ***@param*** *tmp  
 \*/* public void initSetZero(int[] tmp){  
 for(int i=0;i<tmp.length;i++){  
 tmp[i] = 0;  
 }  
 }  
  
 */\*\*  
 \* 递归，判断每个节点所联通的下个节点是否有联通节点；如果，那么nodeConnect对应的参数设置为1；  
 \** ***@param*** *values  
 \** ***@param*** *node  
 \** ***@param*** *nodeConnet  
 \*/* private void checkByDFS(int[][] values , int node, int[] nodeConnet){  
 nodeConnet[node] = 1;  
 for(int i = 0; i<values.length; i++){  
 if(values[node][i] > 0 && nodeConnet[i]<1){  
 checkByDFS(values,i,nodeConnet);  
 }  
 }  
 return;  
 }  
  
 */\*\*  
 \* print graphic matrix;  
 \** ***@param*** *values  
 \*/* private void printGraphic(int[][] values){  
 for(int i=0;i<values.length;i++){  
 String tmp =**""**;  
 for(int j=0;j<values[i].length;j++){  
 tmp = tmp + values[i][j]+**" "**;  
 }  
 System.*out*.println(tmp);  
 }  
 }  
}

test code, junit

package psn.learn.graphic.test;  
  
import org.junit.**Test**;  
import psn.learn.graphic.Graphic;  
  
import java.util.Random;  
  
*/\*\*  
 \* 测试用例说明；  
 \* 1. 使用图的关联矩阵；如果两个点之间有连接，就用1表示，如果不通，则用0表示；  
 \* 2. 由于只是测试无向联通图，所以关联矩阵是个对称矩阵；  
 \*/*public class GraphicTest {  
  
 **@Test** public void testCheckGraphicConnect(){  
 int[][] values = this.initGraphicMatrix(5);  
 Graphic grp = new Graphic();  
  
 grp.checkGraphicConnect(values);  
  
 }  
  
 **@Test** public void testCheckGraphicConnctByFixedMatrix(){  
 Graphic grp = new Graphic();  
 grp.checkGraphicConnect(this.initFixGraphicmatrix());  
 }  
  
 private int[][] initFixGraphicmatrix(){  
 int[][] result = new int[5][];  
 for(int i =0 ;i < 5 ; i++){  
 result[i] = new int[5];  
 }  
  
 */\*\*  
 \* a b c d e  
 \* a 0 1 0 1 0  
 \* b 1 0 1 0 1  
 \* c 0 1 0 1 0  
 \* d 1 0 1 0 1  
 \* e 0 1 0 1 0  
 \*/* result[0][0] = 0;  
 result[0][1] = 1;  
 result[0][2] = 0;  
 result[0][3] = 1;  
 result[0][4] = 0;  
  
 result[1][0] = 1;  
 result[1][1] = 0;  
 result[1][2] = 1;  
 result[1][3] = 0;  
 result[1][4] = 1;  
  
 result[2][0] = 0;  
 result[2][1] = 1;  
 result[2][2] = 0;  
 result[2][3] = 1;  
 result[2][4] = 0;  
  
 result[3][0] = 1;  
 result[3][1] = 0;  
 result[3][2] = 1;  
 result[3][3] = 0;  
 result[3][4] = 1;  
  
 result[4][0] = 0;  
 result[4][1] = 1;  
 result[4][2] = 0;  
 result[4][3] = 1;  
 result[4][4] = 0;  
  
 return result;  
 }  
  
 */\*\*  
 \* 初始化 nodeCnt 节点数的图形矩阵；  
 \** ***@param*** *nodeCnt  
 \** ***@return*** *\*/* private int[][] initGraphicMatrix( int nodeCnt){  
  
 int[][] result = new int[nodeCnt][];  
  
 for(int i =0 ;i<nodeCnt ;i++){  
 result[i] = new int[nodeCnt];  
 for(int j = i; j<nodeCnt ; j++){  
 if(j == i){  
 result[i][j] = 0; // 自己的节点设置为0；  
 } else {  
 //对称矩阵；  
 result[j] = new int[nodeCnt];  
 int s = this.randomValue();  
 result[i][j] = s;  
 result[j][i] = s;  
 }  
 }  
 }  
  
 return result;  
 }  
  
 */\*\*  
 \* 随机值  
 \** ***@return*** *\*/* private int randomValue(){  
 Random random = new Random();  
 return random.nextInt(2);  
 }  
  
}