Java code:

package psn.learn.graphic;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Graphic {  
  
 */\*\*  
 \* through the matrix data to check the graph is connected or not;   
 \** ***@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;  
 }  
 }  
  
 */\*\*  
 \* Initialize the node's connectivity value; the default connectivity of all nodes is 0, that shows not connected；  
 \** ***@param*** *tmp  
 \*/* public void initSetZero(int[] tmp){  
 for(int i=0;i<tmp.length;i++){  
 tmp[i] = 0;  
 }  
 }  
  
 */\*\*  
 \* recursively, to determine whether each node connected to the next node has the Unicom node; if, then nodeConnet corresponding parameter is set to 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);  
 }  
 }  
}

1. test code, junit
2. package psn.learn.graphic.test;  
     
   import org.junit.**Test**;  
   import psn.learn.graphic.Graphic;  
     
   import java.util.Random;  
     
   */\*\*  
    \* test case description；  
    \* 1. Use the graph’s associative matrix; if there is a connection between two points, use 1 to indicate, if not, then use 0；  
    \* 2. because it is only test non-connected Unicom graph, so the correlation matrix is a symmetric matrix；  
    \*/*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;  
    }  
     
    */\*\*  
    \** Initialize the *nodeCnt matrix；  
    \** ***@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; // your own node is set to 0；  
    } else {  
    //symmetric matrix；  
    result[j] = new int[nodeCnt];  
    int s = this.randomValue();  
    result[i][j] = s;  
    result[j][i] = s;  
    }  
    }  
    }  
     
    return result;  
    }  
     
    */\*\*  
    \* random value  
    \** ***@return*** *\*/* private int randomValue(){  
    Random random = new Random();  
    return random.nextInt(2);  
    }  
     
   }