

实验二打印

Java |

```
1 int CreateGraph()
2 {
3     int point, edge;
4     cout << "请输入顶点和边的个数: " << endl;
5     cin >> point >> edge;
6     for (int i = 0; i < point; i++) { // 初始化边的权值
7         for (int j = 0; j < point; j++) {
8             node[i][j] = INF;
9         }
10    }
11    int weight;
12    for (int k = 0; k < edge; k++) {
13        int i, j;
14        cout << "请输入第" << k + 1 << "条边的两个顶点和权值: " << endl;
15        cin >> i >> j >> weight;
16        node[i][j] = weight;
17    }
18    return point;
19 }
20 // 求 n个顶点的多段图的最短路径
21
22 int Path(int n)
23 {
24     int i, j;
25     int cost[100], path[100]; // 存储路径长度和路径
26     for (i = 1; i < n; i++) {
27         cost[i] = INF; // 初始化
28         path[i] = -1;
29     }
30     cost[0] = 0;
31     path[0] = -1;
32     for (j = 1; j < n; j++) { // 前驱节点
33         for (i = j - 1; i >= 0; i--) {
34             if (cost[i] + node[i][j] < cost[j]) {
35                 cost[j] = cost[i] + node[i][j]; // 更新值
36                 path[j] = i; // 将i的值
```

Microsoft Visual Studio 调试控制台

请输入顶点和边的个数:
7 12
请输入第1条边的两个顶点和权值:
0 1 4
请输入第2条边的两个顶点和权值:
0 3 8
请输入第3条边的两个顶点和权值:
0 2 5
请输入第4条边的两个顶点和权值:
1 3 6
请输入第5条边的两个顶点和权值:
2 3 5
请输入第6条边的两个顶点和权值:
1 4 6
请输入第7条边的两个顶点和权值:
3 4 8
请输入第8条边的两个顶点和权值:
2 5 7
请输入第9条边的两个顶点和权值:
3 6 9
请输入第10条边的两个顶点和权值:
4 6 5
请输入第11条边的两个顶点和权值:
5 6 4
请输入第12条边的两个顶点和权值:
3 5 9
6<-4<-1<-0
最短路径长度为: 15

D:\Users\86136\source\repos\dongtaiguihua\x64\Debug\dongtaiguihua.exe (进程 22560)已退出，代码为 0 (0x0)。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .

```
1  public class test21 {
2
3      private static final int N = 4; // 棋盘大小
4      private static int[] board = new int[N + 1]; // 存储每行皇后的列位置，索引
从1开始
5
6      public static void main(String[] args) {
7          solveNQueens(1); // 从第1行开始放置皇后
8      }
9
10     // 检查当前位置是否可以放置皇后
11     private static boolean isSafe(int row, int col) {
12         for (int i = 1; i < row; i++) {
13             if (board[i] == col || Math.abs(board[i] - col) == Math.abs(i
- row)) {
14                 return false;
15             }
16         }
17         return true;
18     }
19
20     // 解决N皇后问题的核心函数
21     private static void solveNQueens(int row) {
22         for (int col = 1; col <= N; col++) {
23             if (isSafe(row, col)) {
24                 board[row] = col; // 放置皇后
25                 if (row == N) {
26                     printSolution(); // 如果已经放置了所有皇后，打印解
27                 } else {
28                     solveNQueens(row + 1); // 继续放置下一行的皇后
29                 }
30             }
31         }
32     }
33
34     // 打印当前解
35     private static void printSolution() {
36         for (int i = 1; i <= N; i++) {
37             for (int j = 1; j <= N; j++) {
38                 if (board[i] == j) {
39                     System.out.print("Q ");
40                 } else {
41                     System.out.print(".");
42                 }
43             }
44         }
45     }
46 }
```

```
43         }
44         System.out.println();
45     }
46     System.out.println();
47 }
48 }
```

The screenshot shows a Java development environment with the following details:

- File Explorer:** Shows files Main.java, test1.java, test2.java, test3.java, and test21.java.
- Run Configuration:** The configuration "test21" is selected.
- Output Window:** Displays the command "C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ... followed by the program's output:

```
. Q .
. . . Q
Q . .
. . Q .
.
. . Q .
Q . .
. . . Q
. Q . .
```
- Status Bar:** Shows the message "进程已结束, 退出代码为 0".

```

1 import java.util.Scanner;
2
3 public class test23 {
4
5     private static final int MAXN = 1005;
6     private static int[][] G = new int[MAXN][MAXN]; // 用于存储图中的边
7     private static int[] color = new int[MAXN]; // 用于存储每个节点的颜色
8     private static int n, m; // n表示图中节点的数量, m
表示可供选择的颜色数目
9
10    public static boolean ok(int u, int c) {
11        for (int i = 1; i <= n; i++) {
12            if (G[u][i] == 1 && color[i] == c) {
13                return false;
14            }
15        }
16        return true;
17    }
18
19    public static boolean dfs(int u) {
20        if (u > n) {
21            return true;
22        }
23        for (int i = 1; i <= m; i++) {
24            if (ok(u, i)) {
25                color[u] = i;
26                if (dfs(u + 1)) {
27                    return true;
28                }
29                color[u] = 0; // 回溯
30            }
31        }
32        return false;
33    }
34
35    public static void main(String[] args) {
36        Scanner scanner = new Scanner(System.in);
37
38        System.out.println("请输入顶点数和可用颜色数: ");
39        n = scanner.nextInt();
40        m = scanner.nextInt();
41
42        System.out.println("请输入边数: ");
43        int e = scanner.nextInt();

```

```
44
45     System.out.println("请输入相连接的顶点: "); // 顶点从1开始
46     for (int i = 0; i < e; i++) {
47         int u = scanner.nextInt();
48         int v = scanner.nextInt();
49         G[u][v] = G[v][u] = 1;
50     }
51
52     if (dfs(1)) {
53         for (int i = 1; i <= n; i++) {
54             System.out.println("结点 " + i + " 的颜色为: " + color[i]);
55         }
56     } else {
57         System.out.println("No solution");
58     }
59
60     scanner.close();
61 }
62 }
```

The screenshot shows a Java IDE interface with a terminal window displaying the output of a Java program named test23. The terminal window has tabs for "运行" (Run), "test21" (highlighted), and "test23". The output of the program is as follows:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java" -jar test23.jar  
请输入顶点数和可用颜色数:  
5 3  
请输入边数:  
7  
请输入相连接的顶点:  
1 2  
1 3  
2 3  
2 4  
2 5  
4 5  
3 5  
结点 1 的颜色为: 1  
结点 2 的颜色为: 2  
结点 3 的颜色为: 3  
结点 4 的颜色为: 3  
结点 5 的颜色为: 1  
进程已结束, 退出代码为 0
```

The terminal window also shows the file structure at the bottom: test > src > test23. The status bar indicates the time as 3:22, line endings as CRLF, and the file type as UTF-8.