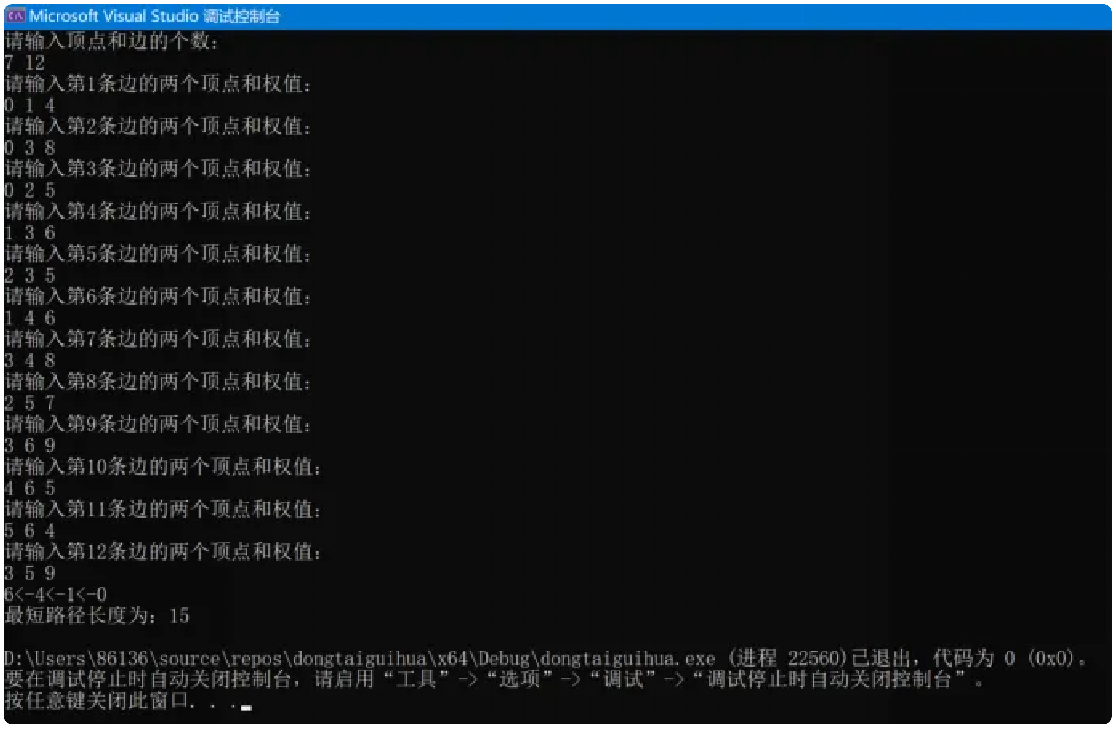
实验二打印

Java

```
int CreateGraph()
{
    int point, edge;
    cout << "请输入顶点和边的个数：" << endl;
    cin >> point >> edge;
    for (int i = 0; i < point; i++) { // 初始化边的权值
        for (int j = 0; j < point; j++) {
            node[i][j] = INF;
        }
    }
    int weight;
    for (int k = 0; k < edge; k++) {
        int i, j;
        cout << "请输入第" << k + 1 << "条边的两个顶点和权值：" << endl;
        cin >> i >> j >> weight;
        node[i][j] = weight;
    }
    return point;
}
// 求 n个顶点的多段图的最短路径

int Path(int n)
{
    int i, j;
    int cost[100], path[100]; // 存储路径长度和路径
    for (i = 1; i < n; i++) {
        cost[i] = INF; // 初始化
        path[i] = -1;
    }
    cost[0] = 0;
    path[0] = -1;
    for (j = 1; j < n; j++) { // 前驱节点
        for (i = j - 1; i >= 0; i--) {
            if (cost[i] + node[i][j] < cost[j]) {
                cost[j] = cost[i] + node[i][j]; // 更新值
                path[j] = i; // 将i的值
```

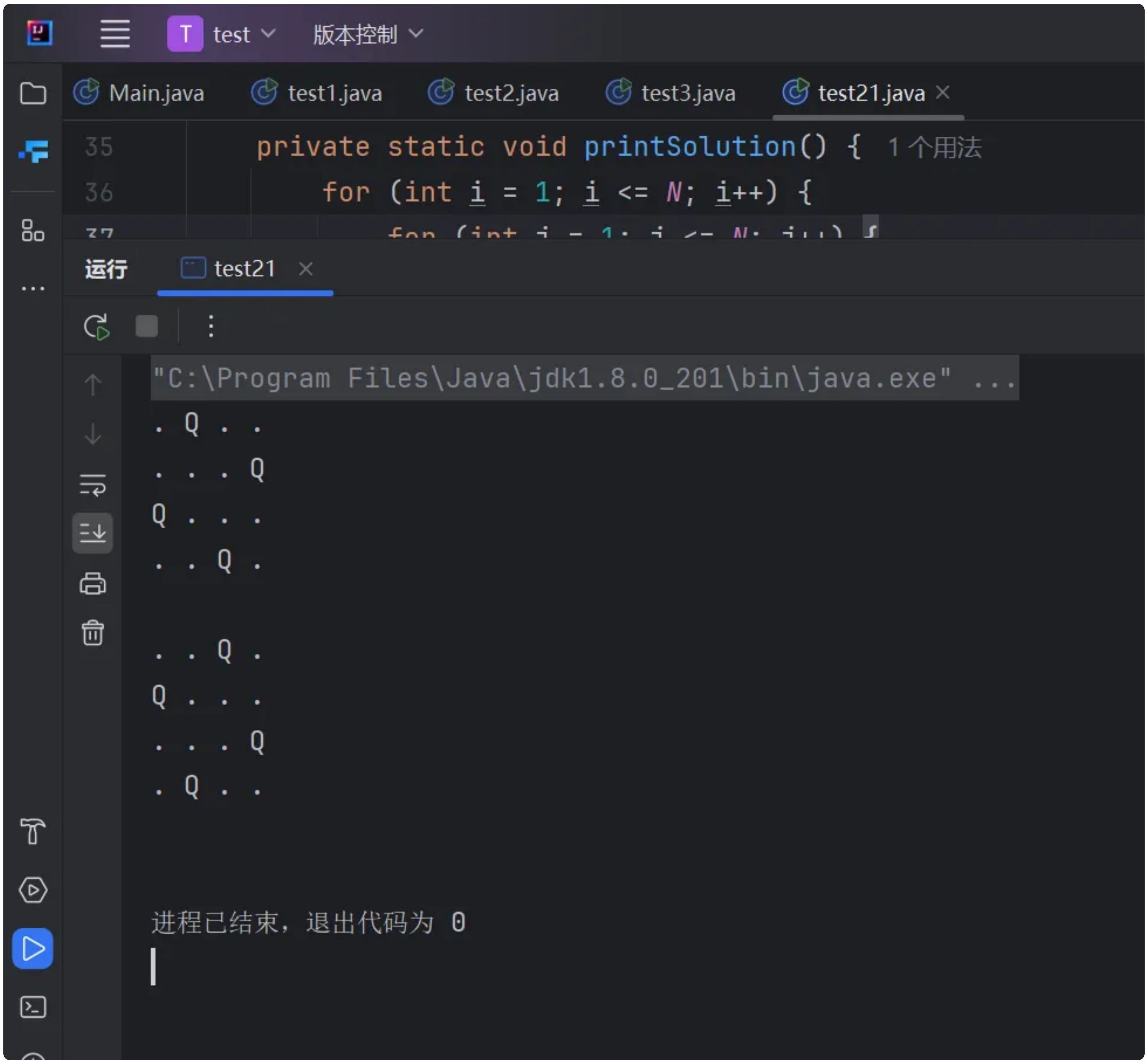


Microsoft Visual Studio 调试控制台

```
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. . .
```

Java

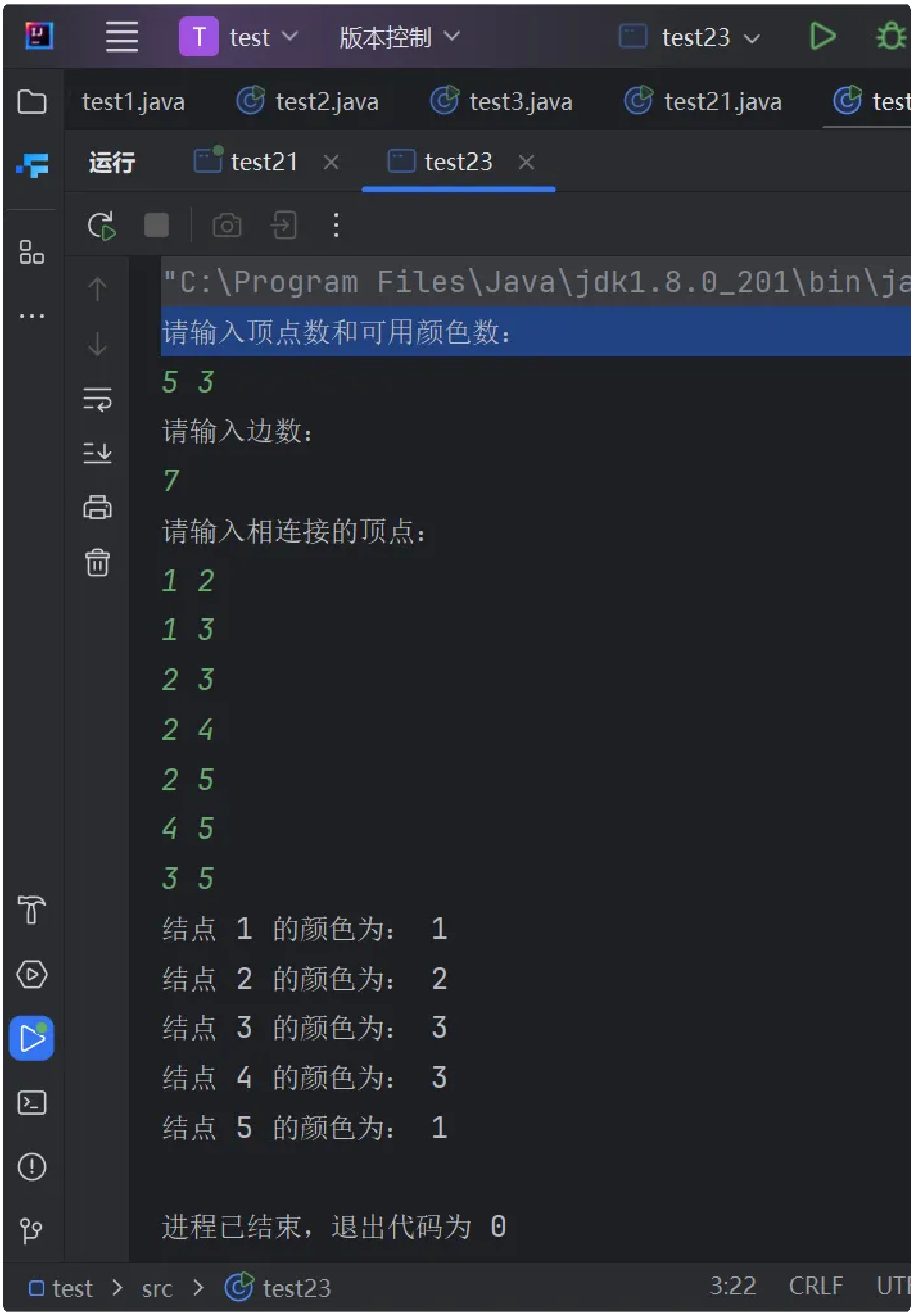
```java
public class test21 {

    private static final int N = 4; // 棋盘大小
    private static int[] board = new int[N + 1]; // 存储每行皇后的列位置，索引从1开始

    public static void main(String[] args) {
        solveNQueens(1); // 从第1行开始放置皇后
    }

    // 检查当前位置是否可以放置皇后
    private static boolean isSafe(int row, int col) {
        for (int i = 1; i < row; i++) {
            if (board[i] == col || Math.abs(board[i] - col) == Math.abs(i - row)) {
                return false;
            }
        }
        return true;
    }

    // 解决N皇后问题的核心函数
    private static void solveNQueens(int row) {
        for (int col = 1; col <= N; col++) {
            if (isSafe(row, col)) {
                board[row] = col; // 放置皇后
                if (row == N) {
                    printSolution(); // 如果已经放置了所有皇后，打印解
                } else {
                    solveNQueens(row + 1); // 继续放置下一行的皇后
                }
            }
        }
    }

    // 打印当前解
    private static void printSolution() {
        for (int i = 1; i <= N; i++) {
            for (int j = 1; j <= N; j++) {
                if (board[i] == j) {
                    System.out.print("Q ");
                } else {
                    System.out.print(". ");
                }
                }
                System.out.println();
            }
            System.out.println();
        }
    }
```



```java
    private static void printSolution() {  1 个用法
        for (int i = 1; i <= N; i++) {
```

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
. Q . .
. . . Q
Q . . .
. . Q .

. . Q .
Q . . .
. . . Q
. Q . .


进程已结束，退出代码为 0
```

```java
import java.util.Scanner;

public class test23 {

    private static final int MAXN = 1005;
    private static int[][] G = new int[MAXN][MAXN]; // 用于存储图中的边
    private static int[] color = new int[MAXN];     // 用于存储每个节点的颜色
    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

    public static boolean ok(int u, int c) {
        for (int i = 1; i <= n; i++) {
            if (G[u][i] == 1 && color[i] == c) {
                return false;
            }
        }
        return true;
    }

    public static boolean dfs(int u) {
        if (u > n) {
            return true;
        }
        for (int i = 1; i <= m; i++) {
            if (ok(u, i)) {
                color[u] = i;
                if (dfs(u + 1)) {
                    return true;
                }
                color[u] = 0; // 回溯
            }
        }
        return false;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("请输入顶点数和可用颜色数：");
        n = scanner.nextInt();
        m = scanner.nextInt();

        System.out.println("请输入边数：");
        int e = scanner.nextInt();

        System.out.println("请输入相连接的顶点: "); // 顶点从1开始
        for (int i = 0; i < e; i++) {
            int u = scanner.nextInt();
            int v = scanner.nextInt();
            G[u][v] = G[v][u] = 1;
        }

        if (dfs(1)) {
            for (int i = 1; i <= n; i++) {
                System.out.println("结点 " + i + " 的颜色为:  " + color[i]);
            }
        } else {
            System.out.println("No solution");
        }

        scanner.close();
    }
}
```



```
"C:\Program Files\Java\jdk1.8.0_201\bin\ja
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
```