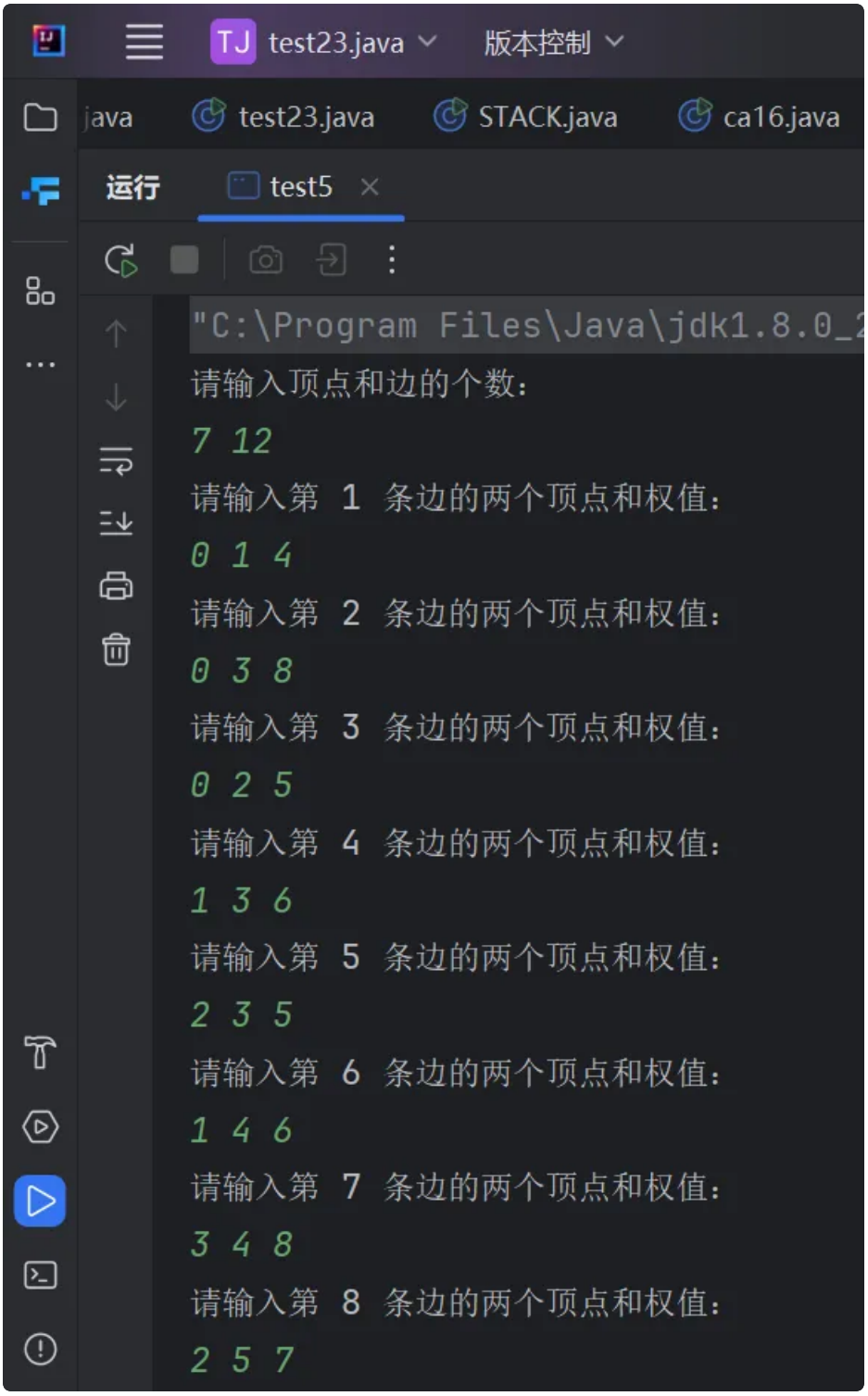
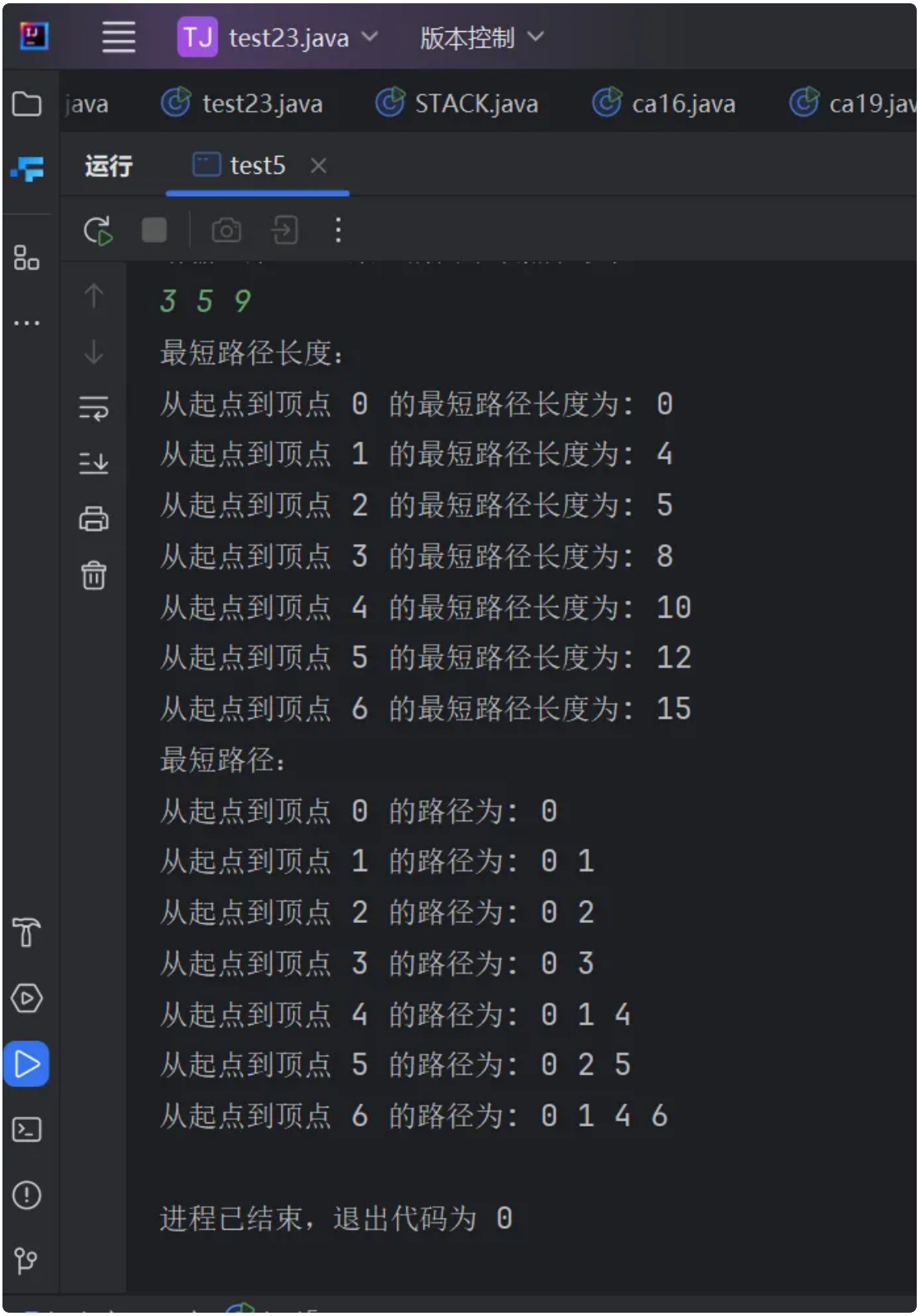
# 实验二打印

DP多段图最短路径问题 Java

```java
import java.util.Scanner;

public class test5 {

    private static final int INF = Integer.MAX_VALUE;

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

        int[][] node = createGraph(scanner);
        int n = node.length;

        int[] cost = new int[n];
        int[] path = new int[n];

        computeShortestPath(node, cost, path);

        printResults(n, cost, path);

        scanner.close();
    }

    private static int[][] createGraph(Scanner scanner) {
        System.out.println("请输入顶点和边的个数：");
        int point = scanner.nextInt();
        int edge = scanner.nextInt();

        int[][] node = new int[point][point];

        // 初始化边的权值
        for (int i = 0; i < point; i++) {
            for (int j = 0; j < point; j++) {
                node[i][j] = INF;
            }
        }

        // 输入每条边的信息
        for (int k = 0; k < edge; k++) {
```


test23.java
版本控制
java
test23.java
STACK.java
ca16.java
运行
test5
"C:\Program Files\Java\jdk1.8.0_2
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

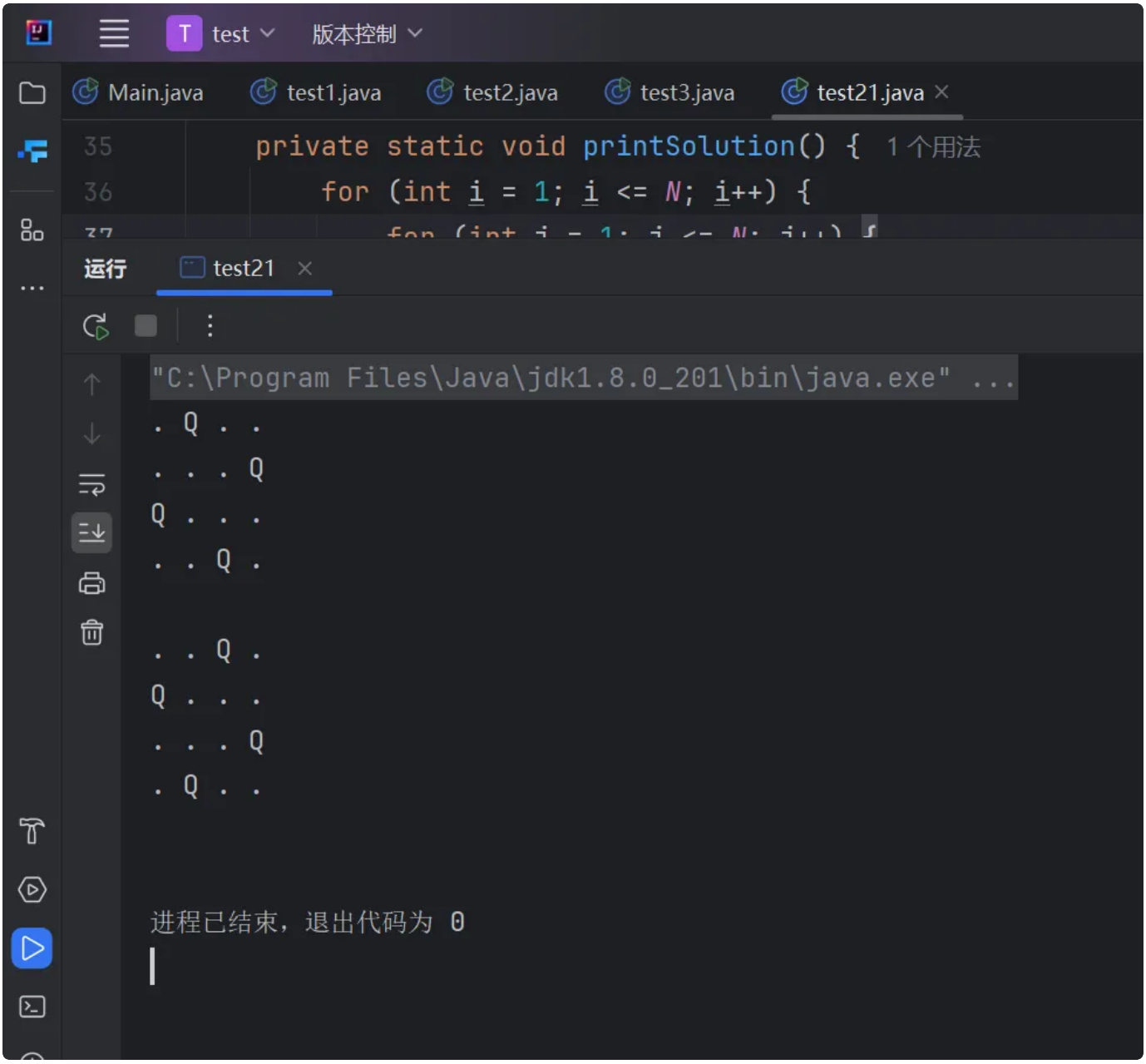
```
请输入第 8 条边的两个顶点和权值：
2 5 7
请输入第 9 条边的两个顶点和权值：
3 6 9
请输入第 10 条边的两个顶点和权值：
4 6 5
请输入第 11 条边的两个顶点和权值：
5 6 4
请输入第 12 条边的两个顶点和权值：
3 5 9
最短路径长度：
从起点到顶点 0 的最短路径长度为: 0
从起点到顶点 1 的最短路径长度为: 4
从起点到顶点 2 的最短路径长度为: 5
从起点到顶点 3 的最短路径长度为: 8
从起点到顶点 4 的最短路径长度为: 10
从起点到顶点 5 的最短路径长度为: 12
从起点到顶点 6 的最短路径长度为: 15
```


test23.java
版本控制
java
test23.java
STACK.java
ca16.java
ca19.jav
运行
test5
3 5 9
最短路径长度:
从起点到顶点 0 的最短路径长度为: 0
从起点到顶点 1 的最短路径长度为: 4
从起点到顶点 2 的最短路径长度为: 5
从起点到顶点 3 的最短路径长度为: 8
从起点到顶点 4 的最短路径长度为: 10
从起点到顶点 5 的最短路径长度为: 12
从起点到顶点 6 的最短路径长度为: 15
最短路径:
从起点到顶点 0 的路径为: 0
从起点到顶点 1 的路径为: 0 1
从起点到顶点 2 的路径为: 0 2
从起点到顶点 3 的路径为: 0 3
从起点到顶点 4 的路径为: 0 1 4
从起点到顶点 5 的路径为: 0 2 5
从起点到顶点 6 的路径为: 0 1 4 6
进程已结束，退出代码为 0

四皇后问题 Java

```java
public class QueenSolver {

    private static final int BOARD_SIZE = 4; // 定义棋盘大小
    private static int[] positions = new int[BOARD_SIZE + 1]; // 记录每行皇后的位置，索引从1开始

    public static void main(String[] args) {
        findQueenPositions(1); // 从第1行开始尝试放置皇后
    }

    /**
     * 检查在指定行和列位置放置皇后是否安全。
     *
     * @param currentRow 当前行号
     * @param column 列号
     * @return 如果位置安全返回true，否则返回false
     */
    private static boolean checkSafety(int currentRow, int column) {
        for (int prevRow = 1; prevRow < currentRow; prevRow++) {
            // 检查列冲突和对角线冲突
            if (positions[prevRow] == column || Math.abs(positions[prevRow] - column) == Math.abs(prevRow - currentRow)) {
                return false;
            }
        }
        return true;
    }

    /**
     * 使用回溯法解决N皇后问题。
     *
     * @param currentRow 当前正在处理的行号
     */
    private static void findQueenPositions(int currentRow) {
        for (int column = 1; column <= BOARD_SIZE; column++) {
            if (checkSafety(currentRow, column)) { // 如果当前位置安全
                positions[currentRow] = column; // 在当前行放置皇后
                if (currentRow == BOARD_SIZE) { // 如果所有皇后都已经放置
                    displaySolution(); // 输出当前解
                } else {
                    findQueenPositions(currentRow + 1); // 尝试放置下一行的皇后
                }
            }
        }
```

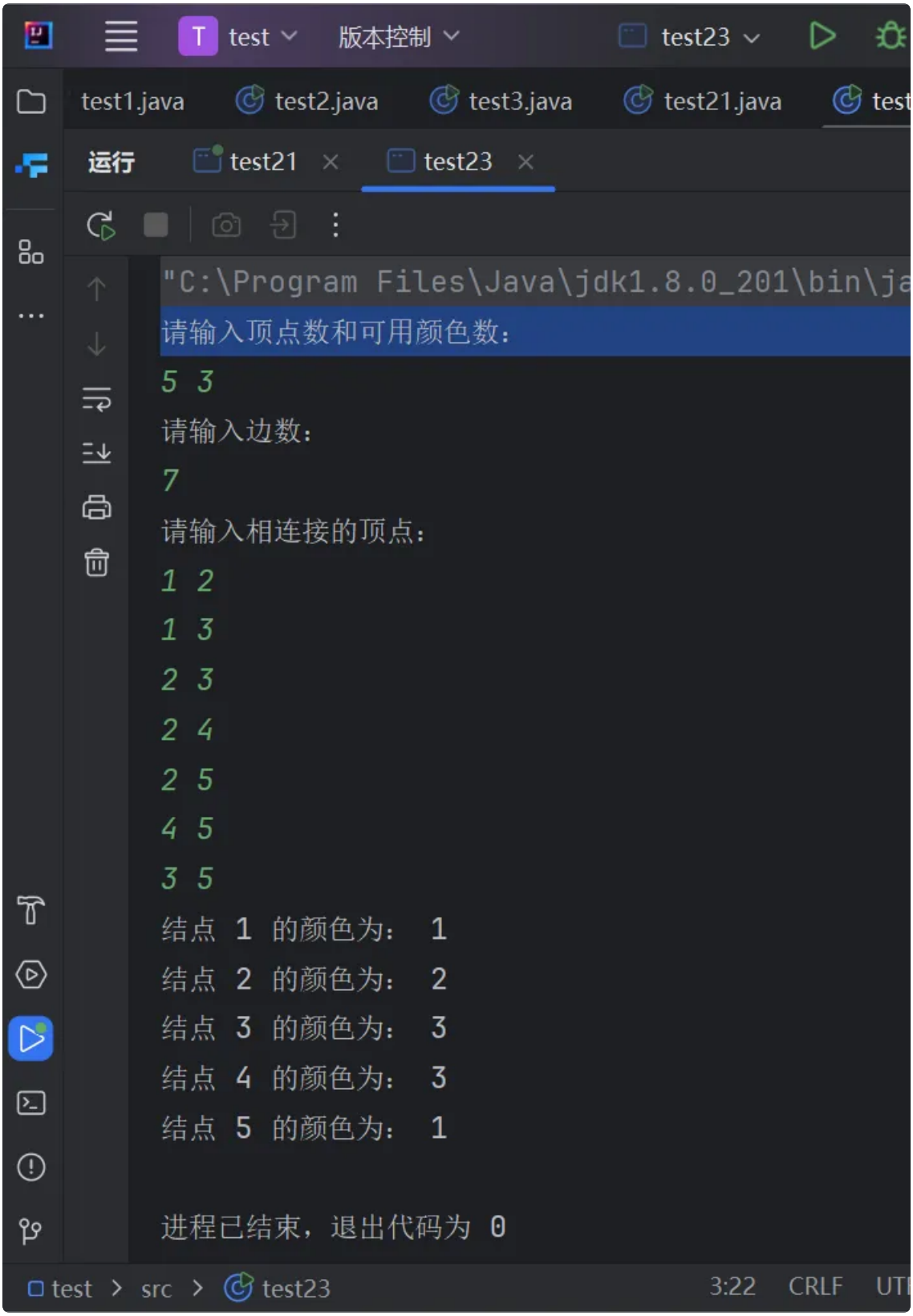
```
        }

        /**
         * 输出当前找到的皇后布局方案。
         */
        private static void displaySolution() {
            for (int row = 1; row <= BOARD_SIZE; row++) {
                for (int column = 1; column <= BOARD_SIZE; column++) {
                    // 如果当前位置有皇后，则输出'Q'，否则输出'.'
                    if (positions[row] == column) {
                        System.out.print("Q ");
                    } else {
                        System.out.print(". ");
                    }
                }
                System.out.println(); // 换行，准备输出下一行
            }
            System.out.println(); // 输出完一个解后，空一行以便区分不同的解
        }
    }
```


test
版本控制
Main.java
test1.java
test2.java
test3.java
test21.java
35 private static void printSolution() { 1 个用法
36 for (int i = 1; i <= N; i++) {
运行
test21
"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

图着色问题 Java

```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();
    }
}
```


test
版本控制
test23
test1.java
test2.java
test3.java
test21.java
test
运行
test21
test23
"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
test
src
test23
3:22
CRLF