

深圳大学实验报告

课程名称： 算法设计与分析

实验项目名称： 八皇后问题

学院： 计算机与软件学院

专业： 软件工程

指导教师： 李荣华

报告人： 洪继耀 学号： 2014150120 班级： 02

实验时间： 2016.10.19

实验报告提交时间： 2016.10.19

实验目的与要求：

一、实验目的：

1. 掌握回溯法设计思想。
2. 掌握八皇后问题的回溯法解法。

八皇后问题是一个以国际象棋为背景的问题：如何能够在 8×8 的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后？为了达到此目的，任两个皇后都不能处于同一条横行、纵行或斜线上。八皇后问题可以推广为更一般的 n 皇后摆放问题：这时棋盘的大小变为 $n \times n$ ，而皇后个数也变成 n 。当且仅当 $n = 1$ 或 $n \geq 4$ 时问题有解。

二、内容：

1. 编写测试代码。

测试数组 (M, N) ，其中 M 代表皇后所在的行， N 代表皇后所在的列。

例如，

第一组测试数据 $(1, 4) (2, 7) (3, 3) (4, 8) (5, 2) (6, 5) (7, 1) (8, 6)$

第二组测试数据 $(1, 5) (2, 2) (3, 4) (4, 7) (5, 3) (6, 8) (7, 6) (8, 1)$

第三组测试数据 $(1, 4) (2, 2) (3, 7) (4, 3) (5, 6) (6, 8) (7, 5) (8, 1)$

判断测试数据（可能得到的解）是否满足八皇后问题要求。

2. 对于八皇后问题，整个程序中应该包括主函数模块，摆放皇后的函数模块，以及判断皇后的位置是否摆放正确的判断模块。对于模块间的关系，在运行主函数的过程中会调用摆放皇后的函数模块，在摆放皇后的函数模块中，又会调用判断皇后位置是否摆放正确的判断模块。

方法、步骤：

三、算法与代码

1. 测试八皇后棋盘

算法与代码

1. 首先我们生成一个棋盘，棋盘用一个一维整型数组表示，并把数组每一项初始化为各不相同数值相差巨大的数。
2. 然后我们开始根据输入摆放棋子。
3. 每摆放一个，就调用 `judge()` 函数来判断当前棋盘是否合法，不合法直接中断。
4. 如果全部摆放完还能没有中断，我们就认为棋盘合法。
5. 核心代码如下

```
/**
 * 根据输入生成棋盘 每摆放一次棋子 检查是否合法
 *
 * QUEENS_NUM 皇后数量
 * checkerboard[] 棋盘数组
 * in 输入读取器
 * @return 摆放是否成功
 */
private static boolean initCheckerboard() {
    // 初始化棋盘
```

```

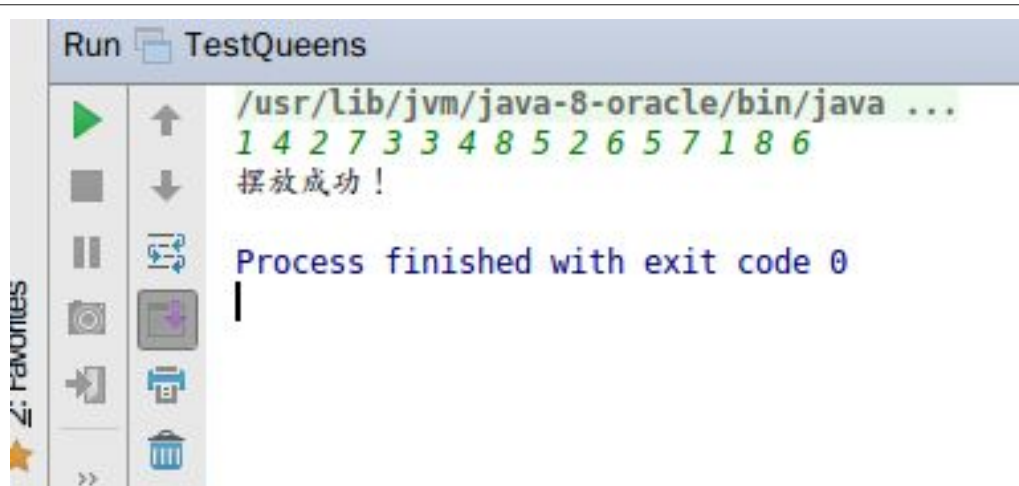
    for (int i = 0; i < QUEENS_NUM; i++) {
        checkerboard[i] = -QUEENS_NUM * i;
    }
    for (int i = 0; i < QUEENS_NUM; i++) {
        int row = in.nextInt(); // 行
        int col = in.nextInt(); // 列
        checkerboard[row - 1] = col - 1; // 摆放棋子
        if (!judge(col - 1)) {
            return false;
        }
    }
    return true;
}

/**
 * 判断皇后的位置是否摆放正确的判断模块
 *
 * checkerboard[] 棋盘数组
 * @param col 摆的位置 哪一列
 * @return 是否合法
 */
private static boolean judge(int col) {
    for (int i = 0; i < col; i++) {
        if (checkerboard[i] == checkerboard[col]
            || Math.abs(col - i) ==
                Math.abs(checkerboard[col] - checkerboard[i])) {
            return false;
        }
    }
    return true;
}

```

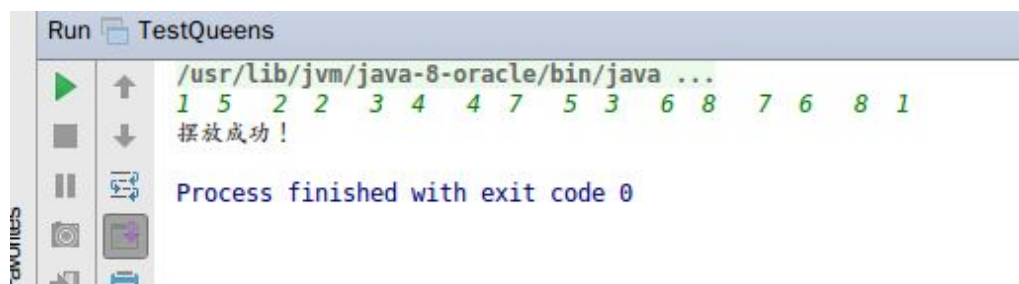
测试结果

1. 第一组数据



```
Run TestQueens
/usr/lib/jvm/java-8-oracle/bin/java ...
1 4 2 7 3 3 4 8 5 2 6 5 7 1 8 6
摆放成功!
Process finished with exit code 0
```

2. 第二组数据



```
Run TestQueens
/usr/lib/jvm/java-8-oracle/bin/java ...
1 5 2 2 3 4 4 7 5 3 6 8 7 6 8 1
摆放成功!
Process finished with exit code 0
```

3. 第三组数据



```
Run TestQueens
/usr/lib/jvm/java-8-oracle/bin/java ...
1 4 2 2 3 7 4 3 5 6 6 8 7 5 8
摆放成功!
Process finished with exit code 0
```

2. 解决八皇后问题

算法与代码

1. 首先我们准备一个棋盘。
2. 然后我们开始往棋盘上摆皇后。

3. 每摆放一个，就调用 `judge()` 函数来判断当前棋盘是否合法，不合法则后退。
4. 每次摆完八个，就认为这是一个可行解，输出这个解，回退到上一步。
5. 每次回退到上一步开始尝试下一步的摆法，如果尝试完了所有可行的下一步，也回退到上一步。
6. 当程序结束，也就输出了所有可行解。
7. 核心代码如下

```
/**
 * 摆放皇后的函数模块 回溯地去尝试所有的可能解
 * queens[] 棋盘数组
 * QUEEN_NUM 皇后数量
 * display() 输出解的函数
 * @param n 刚开始要摆的位置
 */
private static void check(int n) { // 当前是准备摆第 n 个皇后
    if (n == QUEEN_NUM) { // 摆完了
        numOfAns++;
        display();
        return; // 返回上一个可行情形，以便寻找下一个解
    }
    for (int i = 0; i < QUEEN_NUM; i++) {
        queens[n] = i; // 尝试放皇后
        if (judge(n)) {
            check(n + 1); // 尝试下一个位置的可能 即使尝试成功后还是会退回来 因此可遍历所有可能解
        }
    }
}

/**
 * 判断皇后的位置是否摆放正确的判断模块
 * queens[] 棋盘数组
 * @param n 摆的位置
 * @return 是否合法
 */
private static boolean judge(int n) {
    for (int i = 0; i < n; i++) {
        if (queens[i] == queens[n]
            || Math.abs(n - i) ==
```

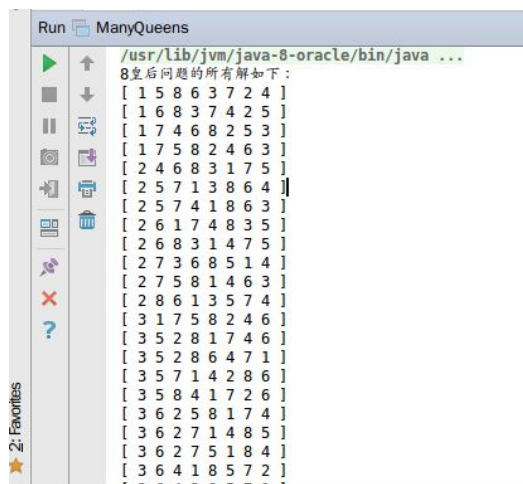
```

        Math.abs(queens[n] - queens[i])) {
            return false; // 只需要检查新放进来的 queens[n]
        }
    }
    return true;
}

```

测试结果

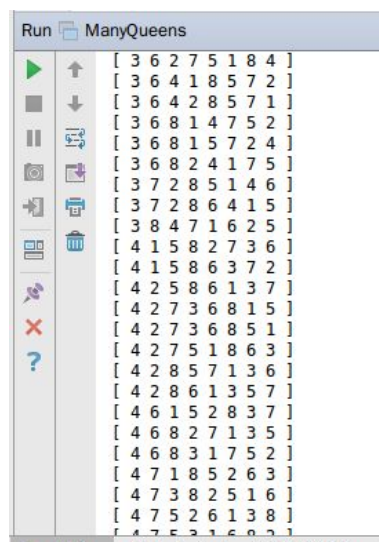
测试结果较长，分多个图来表示



```

Run ManyQueens
/usr/lib/jvm/java-8-oracle/bin/java ...
8皇后问题的所有解如下:
[ 1 5 8 6 3 7 2 4 ]
[ 1 6 8 3 7 4 2 5 ]
[ 1 7 4 6 8 2 5 3 ]
[ 1 7 5 8 2 4 6 3 ]
[ 2 4 6 8 3 1 7 5 ]
[ 2 5 7 1 3 8 6 4 ]
[ 2 5 7 4 1 8 6 3 ]
[ 2 6 1 7 4 8 3 5 ]
[ 2 6 8 3 1 4 7 5 ]
[ 2 7 3 6 8 5 1 4 ]
[ 2 7 5 8 1 4 6 3 ]
[ 2 8 6 1 3 5 7 4 ]
[ 3 1 7 5 8 2 4 6 ]
[ 3 5 2 8 1 7 4 6 ]
[ 3 5 2 8 6 4 7 1 ]
[ 3 5 7 1 4 2 8 6 ]
[ 3 5 8 4 1 7 2 6 ]
[ 3 6 2 5 8 1 7 4 ]
[ 3 6 2 7 1 4 8 5 ]
[ 3 6 2 7 5 1 8 4 ]
[ 3 6 4 1 8 5 7 2 ]

```



```

Run ManyQueens
[ 3 6 2 7 5 1 8 4 ]
[ 3 6 4 1 8 5 7 2 ]
[ 3 6 4 2 8 5 7 1 ]
[ 3 6 8 1 4 7 5 2 ]
[ 3 6 8 1 5 7 2 4 ]
[ 3 6 8 2 4 1 7 5 ]
[ 3 7 2 8 5 1 4 6 ]
[ 3 7 2 8 6 4 1 5 ]
[ 3 8 4 7 1 6 2 5 ]
[ 4 1 5 8 2 7 3 6 ]
[ 4 1 5 8 6 3 7 2 ]
[ 4 2 5 8 6 1 3 7 ]
[ 4 2 7 3 6 8 1 5 ]
[ 4 2 7 3 6 8 5 1 ]
[ 4 2 7 5 1 8 6 3 ]
[ 4 2 8 5 7 1 3 6 ]
[ 4 2 8 6 1 3 5 7 ]
[ 4 6 1 5 2 8 3 7 ]
[ 4 6 8 2 7 1 3 5 ]
[ 4 6 8 3 1 7 5 2 ]
[ 4 7 1 8 5 2 6 3 ]
[ 4 7 3 8 2 5 1 6 ]
[ 4 7 5 2 6 1 3 8 ]
[ 4 7 5 3 1 6 8 2 ]

```

```
Run ManyQueens
[ 4 7 3 8 2 5 1 6 ]
[ 4 7 5 2 6 1 3 8 ]
[ 4 7 5 3 1 6 8 2 ]
[ 4 8 1 3 6 2 7 5 ]
[ 4 8 1 5 7 2 6 3 ]
[ 4 8 5 3 1 7 2 6 ]
[ 5 1 4 6 8 2 7 3 ]
[ 5 1 8 4 2 7 3 6 ]
[ 5 1 8 6 3 7 2 4 ]
[ 5 2 4 6 8 3 1 7 ]
[ 5 2 4 7 3 8 6 1 ]
[ 5 2 6 1 7 4 8 3 ]
[ 5 2 8 1 4 7 3 6 ]
[ 5 3 1 6 8 2 4 7 ]
[ 5 3 1 7 2 8 6 4 ]
[ 5 3 8 4 7 1 6 2 ]
[ 5 7 1 3 8 6 4 2 ]
[ 5 7 1 4 2 8 6 3 ]
[ 5 7 2 4 8 1 3 6 ]
[ 5 7 2 6 3 1 4 8 ]
[ 5 7 2 6 3 1 8 4 ]
[ 5 7 4 1 3 8 6 2 ]
[ 5 8 4 1 3 6 2 7 ]
[ 5 8 4 1 7 2 6 3 ]
```

```
[ 7 2 6 3 1 4 8 5 ]
[ 7 3 1 6 8 5 2 4 ]
[ 7 3 8 2 5 1 6 4 ]
[ 7 4 2 5 8 1 3 6 ]
[ 7 4 2 8 6 1 3 5 ]
[ 7 5 3 1 6 8 2 4 ]
[ 8 2 4 1 7 5 3 6 ]
[ 8 2 5 3 1 7 4 6 ]
[ 8 3 1 6 2 5 7 4 ]
[ 8 4 1 3 6 2 7 5 ]
共有解92个
Process finished with exit code 0
```

四、数据分析

共有 92 个解。

五、感想

(1)遇到的问题

没遇到什么问题。

(2)心得

算法真奇妙，回溯法真好用。

深圳大学学生实验报告用纸

实验结论：

<p>指导教师批阅意见：</p>
<p>成绩评定：</p>
<p>指导教师签字：</p> <p>年 月 日</p>
<p>备注：</p>

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。