项目说明文档

数据结构课程设计

——N皇后问题

作 者 姓 名： 曹晓慈

学 号： 2052844

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

## 分析

### 1.背景分析

八皇后问题（英文：Eight queens），是由国际西洋棋棋手马克斯·贝瑟尔于1848年提出的问题，是回溯算法的典型案例。

问题表述为：在8×8格的国际象棋上摆放8个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。高斯认为有76种方案。1854年在柏林的象棋杂志上不同的作者发表了40种不同的解，后来有人用图论的方法解出92种结果。如果经过±90度、±180度旋转，和对角线对称变换的摆法看成一类，共有42类。计算机发明后，有多种计算机语言可以编程解决此问题。

### 2.功能分析

八皇后在棋盘上分布的各种可能的格局数目非常大，约等于2的32次方种，但是，可以将一些明显不满足问题要求的格局排除掉。由于任意两个皇后不能同行，即每行只能放置一个皇后，因此将第i个皇后放在第i航上，这样在放置第i个皇后时，只要考虑它与前i-1个皇后处于不同列和不同对角线位置上即可。

解决这个问题采用回溯法，首先将第一个皇后放置在第一行第一列，然后，依次在下一行上放置一个皇后，直到八个皇后全部放置安全。在放置每个皇后时，都依次兑每一列进行检测，首先检测放在第一列是否与已放置的皇后冲突，如不冲突，则将皇后放置在该列，否则，选择改行的下一列进行检测。如整行的八列都冲突，则回到上一行，重新选择位置，依次类推。

## 设计

### 成员与操作设计

因为该题目的皇后数量是由操作者输入，所以这里要用到vector这样的动态数组对皇后的位置进行计算。成功的摆法可以用一个二维数组mark进行储存，有皇后的地方值为1，不然则为0.

操作函数：

void output(vector<int> &vec) //输出皇后位置

int check(int t,vector<int> &vec) //检查皇后位置是否满足

void BackTrace(int t,vector<int> &vec) //回溯算法

## 实现

### 回溯算法实现

因为N皇后问题不能同行同列，所以我们可以用从第一行的方法到最后一行的方法，利用数组下标代表行数，数组下标所对应的值代表皇后在该行的所在列数，当该行该列皇后的位置不满足时再返回到上一行将上一行皇后的位置进行改变在进行后续的检查，通过这样进行函数的递归直到皇后全部安放完成递归进行return。

void BackTrace(int t,vector<int> &vec)

{

if (t > N)

{

sum++;

output(vec);

}

else

{

for (int i = 1; i <= N; i++)

{

vec[t] = i;

if (check(t,vec))

{

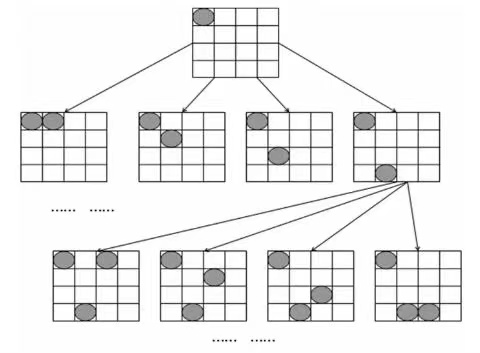
BackTrace(t + 1,vec);

}

}

}

}



### 判断皇后位置是否满足要求

考虑到我们是一行一行对皇后位置进行确定，因此只需要判断列数是否相同以及是否在对角线上即可。对于列数是否相同判断只需对数组对应的值是否相等进行判断即可，而是否在对角线的判断，可以直到如果在对角线上时两个坐标的行数之差的绝对值等于列数之差的绝对值，因此根据以上可以写出皇后位置合法的判断。

int check(int t,vector<int> &vec)

{

for (int j = 1; j < t; j++)

{

if ((vec[j] == vec[t]) || (abs(vec[j] - vec[t]) == abs(j - t)))

return 0;

}

return 1;

}

### 主要系统的实现

程序主要系统实现比较简单，对输入皇后的数量进行判断，不能小于4，否则题目无解，输入正确情况下开始进行函数的调用。

int main()

{

cout << "现有N\*N的棋盘，放入N个皇后，要求所有皇后不在同一行.列和同一斜线上!" << endl;

cout << endl;

cout << "请输入皇后的个数(大于等于4）:";

while (1)

{

cin >> N;

if (cin.fail() || N < 4)

{

cin.clear();

cin.ignore(1024, '\n');

cout << "输入错误，请再次输入考生人数 :" << endl;

continue;

}

break;

}

cout << endl;

cout << "皇后摆法:" << endl;

cout << endl;

vector<int> vec(N+1, 0);

BackTrace(1,vec);

cout << "共有" << sum << "种解法!" << endl;

return 0;

}

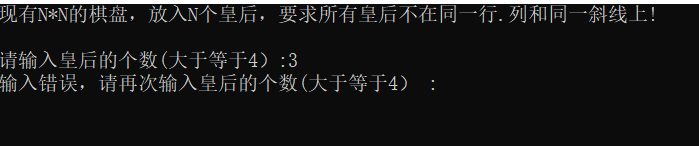
## 测试

### 1.输入不符合要求

**测试用例：**3

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

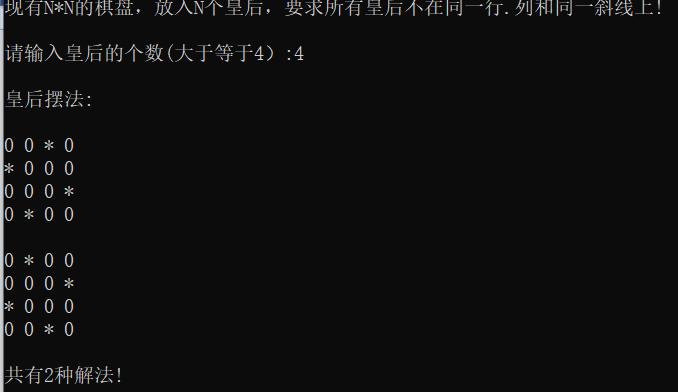
****

### 2．正确输入

**测试用例：**4

**预期结果：**程序输出正确结果。

**实验结果：**

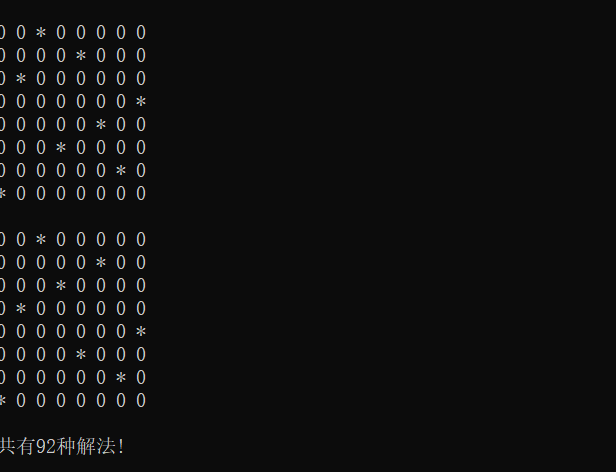


### 3.正确输入

**测试用例：**8

**预期结果：**程序输出正确结果。

**实验结果：**

****

### 正确输入

**测试用例：**10

**预期结果：**程序输出正确结果724，但花费了较长时间。

**实验结果：**

