# N皇后问题

## 一.项目背景：

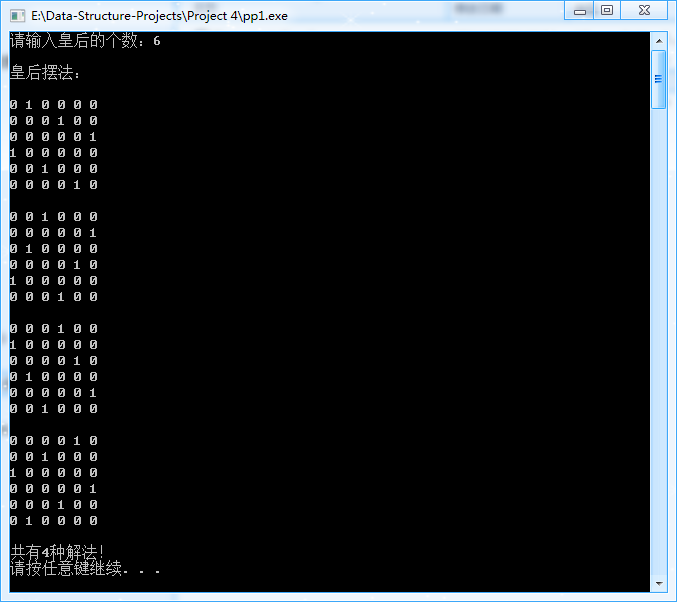
八皇后问题是一个古老而著名的问题，是回溯算法的经典问题。该问题是十九世纪著名的数学家高斯在1850年提出的：在8\*8的国际象棋棋盘上，安放8个皇后，要求没有一个皇后能够“吃掉”任何其它一个皇后，即任意两个皇后不能处于同一行，同一列或者同一条对角线上，求解有多少种摆法。

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得到结论，有92中摆法。

本实验拓展了N皇后问题，即皇后个数由用户输入。

## 二.项目功能

输入皇后个数，程序输出详细解法及解法个数：



## 三.程序描述

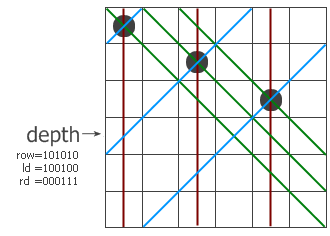
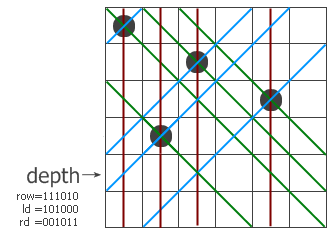
八皇后是一个NP问题，解决这个问题采用回溯法，首先将第一个皇后放置在第一行第一列，然后，依次在下一行上放置一个皇后，直到八个皇后全部放置安全。在放置每个皇后时，都依次兑每一列进行检测，首先检测放在第一列是否与已放置的皇后冲突，如不冲突，则将皇后放置在该列，否则，选择改行的下一列进行检测。如整行的八列都冲突，则回到上一行，重新选择位置，依次类推。

对于该回溯法，我采用了位运算来实现，原因有二：

1.实现速度快。在高级语言程序中，位运算就是利用二进制做运算，在机器中不需要做十进制与二进制的转换，故计算速度最快。

2.节省空间。普通的回溯法需要使用数组来保存状态，而位运算只需要用数字来保存状态，大大节省了空间。

具体思路如下：

如上图所示，row、ld和rd，分别表示在纵列和两个对角线方向的限制条件下这一行的哪些地方不能放。我们以6×6的棋盘为例，看看程序是怎么工作的。假设现在已经递归到第四层，前三层放的子已经标在左图上了。红色、蓝色和绿色的线分别表示三个方向上有冲突的位置，位于该行上的冲突位置就用row、ld和rd中的1来表示。把它们三个并起来，得到该行所有的禁位，取反后就得到所有可以放的位置（用pos来表示）。然后用pos and (not pos + 1)取出最右边的那个1。这样，p就表示该行的某个可以放子的位置，把它从pos中移除并递归调用test过程。注意递归调用时三个参数的变化，每个参数都加上了一个禁位，但两个对角线方向的禁位对下一行的影响需要平移一位。最后，如果递归到某个时候发现row=111111了，说明六个皇后全放进去了，找到的解的个数加一。

//row、ld和rd，分别表示在纵列和两个对角线方向的限制条件下这一行的哪些地方不能放

void test(int row, int ld, int rd)

{

int pos, p;

if (row != upperlim)

{

//取出当前可以放的状态,用1表示可以放

pos = upperlim&~(row | ld | rd);

while (pos)

{

//取出最右边的第一个1

p = pos&-pos;

ans[m] = p;

m++;

//减去这个状态

pos -= p;

test(row + p, (ld + p) << 1, (rd + p) >> 1);

m--;

}

}

else

{

printAns(); //打印答案

sum++;

}

}