

F 盗墓笔记之青铜巨门——by Arthur

题目描述

看密匝匝蚁排兵,乱纷纷蜂酿蜜,急攘攘蝇争血。天真和 Thor 刚刚躲过蝙蝠一样怪物的洗礼,便赶上了这千年不遇的 $n*n$ 鬼俑阵列。

聪明的 Thor 很快便发现,阵列中只有普通步兵俑和执旗步兵俑,其中执旗步兵俑有 n 个。且任意行、列以及斜行里都不会出现两个或两个以上的执旗步兵俑。

他还发现,阵列每过一段时间便会进行一次变换,变换后的阵列仍然符合以上特点。

惊鸿一瞥,似乎小哥也在阵列里,而且就是这些执旗步兵俑中的一个。急于找到小哥的天真不知如何是好。

Thor:"很简单嘛,写一个程序就好了。"

天真:"怎么写?"

Thor 看向了你.....

现请你来计算一下,这 $n*n$ 的鬼俑阵列,一共会有多少种排列方式呢?

输入

多组测试数据。

每组测试数据只有一行,为一个整数 n ($1 \leq n \leq 15$), 含义见题目描述。

输出

对于每组数据,请输出所有可能出现的排列形式的个数。

输入样例

8

输出样例

92

解题分析

很简单喽，打表呗——

0,
1,
0,
0,
2,
10,
4,
40,
92,
352,
724,
2680,
14200,
73712,
365596

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.



上面的算法不算数的哦~



然后让我们进入正题。本题是 N 皇后问题的原版题目，未做任何改变，考察的就是结合数组与递归的回溯算法。

然而朴素的回溯会超时（Atrhur 承认这题的时限卡得有点不到位，Sorry....）这里可以对其进行几个进一步的优化。

- ① 细观察所有 N 皇后的可行解，会发现一个规律，每一行有且仅有一个皇后，也就是说每一行只可能有一个皇后，不可能多，也不可能少。这样一来，行就不需要计入考虑喽，直接看列。
- ② 对于每个皇后来说，只要考虑左上、右下、左下、右上四个斜行以及一个竖列（横行已经被忽略了，参见①）这样我们可以忽略掉棋盘的影响，而将这五趟直接看成五个状态来保存，这样就压缩下来了。而且判断的时候也只需要考虑这五个状态，这样时间也会进一步的压缩下来。

回溯神马的，参见代码。

参考代码（一秒版）

```
#include<stdio.h>
int a[20],b[20],c[50],d[50],n,i;
a 保存放入皇后的位置，b 为皇后的竖列，c 与 d 对应两个斜行，由于两个斜行
被分为四段，c 与 d 的大小要为应有的二倍。b、c、d 的状态 1 代表填满，0 代
表空置。
int t;//t 为计数器
void find(int i)//i 代表当前放的是第 i 个皇后
{
    for(int j=1;j<=n;j++)//对其他的皇后的状态进行判断
        if(b[j]==0&& c[i+j]==0&& d[i-j+n]==0)
            { //找到一个空位。
                a[i]=j; //记录位置
                b[j]=c[i+j]=d[i-j+n]=1;
                //与这一皇后相关的位置全部填满
                if(i<n) find(i+1);
                //如果这不是最后一枚皇后，继续寻找
            }
        else t++;
        //否则这是一种方案，计数器+1
        b[j]=c[i+j]=d[i-j+n]=0;
        //将当前皇后的位置置空，循环结束，从而实现回溯
    }
}

int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        t=0;
        for(i=1;i<=n;i++)
            b[i]=c[i]=d[i]=c[i+n]=d[i+n]=0;//全部置空
        find(1);//从第一个皇后开始寻找
        printf("%d\n",t);
    }
}
```