#### 位运算法★★★

|  |  |
| --- | --- |
| man | 位运算法解决Ｎ皇后问题应该是目前最快的方法之一，但要想弄懂此算法，前提是必须对位运算有深入的了解才行。  即用数组来记录棋盘，有皇后的地方赋值为1，没有皇后的地方赋值为0，这可以用二进制的0和1来表示。 |

假设n＝6时，此时已尝试搜索到了第4层。其中虚线分别表示列限制、正斜线限制和反斜线限制，如图2.15所示。

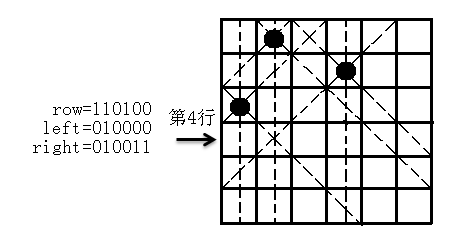


图2.15

定义3个二进制变量row、left、right，row的每一位上的数字代表对应的列是否已经放了皇后，left的每一位上的数字代表当前行的对应列是否受正斜线的影响，right的每一位上的数字代表当前行的对应列是否受反斜线影响。

可以看到，第4行的第1，2，4，5，6列上已经不能放皇后了，而这恰好是row、left和right三个数的合并值即110111。

在尝试第5行时，row的值仍然适用，但right和left的值就不再适用了，我们需要将left<<1（向左移动一位）就可以将正斜线对该行的影响位置转换到对下一行的影响位置，即100000；将right>>1向右移动一位就可以将反斜线对该行的影响位置转换到对下一行的影响位置即001001。

当然在尝试进入第5行时，需先将第4行因新放上的皇后而导致的影响计算在内，即将现有的3个数通过位运算得出一个可行位置的值p，比如当p＝001000就表示第4行第3列可放，然后将row，left，right分别加上p就可以得到下一行的标记状况（正反斜线要左移右移）。

具体算法实现为：

（1）定义一个数lim＝(1 << n)－1，即11111…11（n个1），以防止运算中出现的左移右移产生的超界情况；

（2）定义一个数pos来记录当前行全部的可行位置即pos＝lim & ~(row | left | right)；

（3）取出最右边的一个可行位置1进行尝试，即p＝pos & -pos，注意–pos＝~pos+1；

（4）将该位置的p从pos中移除，即pos＝pos－p；

（5）left和right分别左移一位、右移一位后递归继续下一行的尝试。

参考程序如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | //n皇后问题 — 位运算  #include <bits/stdc++.h>  using namespace std;  int lim,sum;  void test(int row,int left,int right)  {  int pos,p;  if(row!=lim) //未填满，即不为11111111时  {  pos=lim & ~(row | left | right); //取反后得到所有可以放的位置  while(pos!=0)  {  p=pos & -pos; //取出最右边的1进行尝试  pos=pos-p; //将该位置的P从pos中移除  test(row+p,(left+p)<<1,(right+p)>>1);  }  }  else  ++sum;  }  int main()  {  int n;  scanf("%d",&n);  lim=(1 << n)-1; //结果为11111111  test(0,0,0); //纵列，左对角线，右对角线  printf("%d\n",sum);  return 0;  } |