##### 位运算法

全排列问题可以转化为n“车”问题：中国象棋中，“车”的攻击范围为横竖直线，在n×n的棋盘，放置n个彼此不能互相攻击的车的所有方案恰恰对应n的全排列方案。例如当棋盘为3×3时，以棋盘行数对应全排列数的位，以棋子所在列数对应该位的数，则一一对应如图2.20所示。

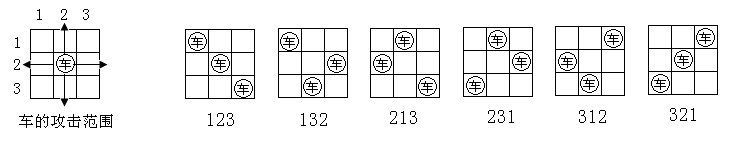


图2.20

由于棋盘格子的棋子放置有两种状态，即放置或者不放置，因此可以转化为1和0的二进制存储，二进制的位值一一对应棋子的放置情况如图2.21所示。

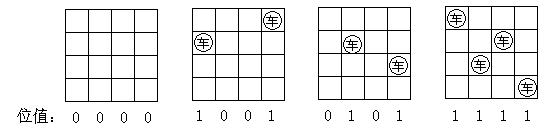


图2.21

可以考虑使用一个整型变量例如layout的32个位来对应当前棋盘棋子的放置情况。即最大对应32×32的棋盘，求出最多32个元素的全排列。显然，当程序由初始状态即空棋盘0运行至layout值全为11111…111（n个1）时，则一个棋子摆放方案成功，输出该方案。其中11111…111（n个1）可通过(1<<n)－1运算得出。

具体摆放过程以n＝6为例，当前3行棋子已摆好的状态如图2.22所示。

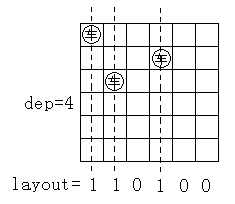


图2.22

设有深搜函数DFS(int dep，int layout)，则求第4行棋子如何摆放即为DFS(4，001011)。

layout以1表示该处棋子已放置，以0表示该处棋子未放置，但如果以1表示该处棋子未放置，以0表示该处棋子已放置，将对寻找该行第一个未放置棋子的位置更方便些，因此可定义一个整型变量Canput＝(~layout)&full，即可求出所需要的值，如图2.23所示。

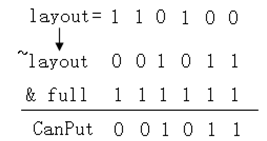


图2.23

从右到左查找第一个1的位置即该行可放棋子的位置，可用CanPut&(-CanPut)获得。

在该行可放棋子的位置放上棋子（0变为1），保存该布局后，继续递归求第5行棋子如何摆放即DFS(5，layout＋CanPut&(-CanPut))直到结束，如图2.24所示。

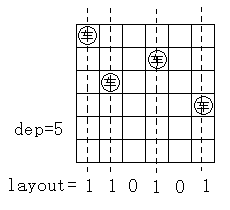


图2.24

参考代码如下所示。

|  |  |
| --- | --- |
| 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  33  34  35  36  37  38  39  40  41  42  43 | //全排列问题 — 位运算  #include <bits/stdc++.h>  using namespace std;  int N,a[33],full,bitmod[39],Count;  void Print()  {  for(int i=1; i<N; ++i)  cout<<a[i];  cout<<a[N]<<'\n';  }  void DFS(int dep,int layout)  {  if(layout==full) //如果所有值均被取走，则输出一个排列  {  Print();  ++Count;  }  else  {  int p,CanPut=(~layout)&full;//获得当前可取的数  while(CanPut)  {  p=CanPut&(-CanPut); //找到最右边为1的数  CanPut-=p; //将该数取出  a[dep]=bitmod[p%37]; //保存该棋子的位置  DFS(dep+1,layout+p);  }  }  }  int main()  {  scanf("%d",&N);  for(int i=0; i<N; ++i)  bitmod[(1<<i) % 37]= i+1; //建立哈希表  full=(1<<N)-1; //初始为111111...111111，表示所有数均被取走  DFS(1,0);  printf("%d\n",Count);  return 0;  } |

该程序的变量p，保存该行自右到左找到的第1个可放棋子的位置，例如找到的位置为自右向左第3个位置时，p的值应为二进制100，即a[dep]的值应赋值为3，那么，如何将二进制100转换为十进制的3呢，一个巧妙的办法是事先将诸如1，10，100，1000，10000…1000…000之类的二进数通过对一个大于32的质数37取余的预处理映射为十进制的1，2，3，…，N保存到数组bitmod[]，如表2.1所示。

表2.1　预处理过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列数 | 4 | 3 | 2 | 1 |
| P值（二进制） | 1000 | 100 | 10 | 1 |
| P%37 | 8 | 4 | 2 | 1 |
| bitmod[] | bitmod[8]＝4 | bitmod[4]＝3 | bitmod[2]＝2 | bitmod[1]＝1 |

这样当取得P值例如P＝1000时，由于1000%37＝8，且bitmod[8]＝4，则当P＝1000时，位置为4。这即是所谓的哈希表（散列表）的简单应用。