# 问题描述

\*问题描述：在n\*n格的棋盘上放置彼此不受攻击的n个皇后。按照国际象棋的规则，皇后可以攻击与之处在同一行或同一列或同一斜线上的棋子。n皇后问题等价于在n\*n的棋盘上放置n个皇后，任何2个皇后不放在同一行或同一列或同一斜线上。

\*算法设计：设计一个解n皇后问题的队列式分支限界法，计算在n\*n个方格上放置彼此不受攻击的n个皇后的一个放置方案。

\*数据输入：由文件input.txt给出输入数据。第一行有1个正整数n。

\*结果输出：将计算的彼此不受攻击的n个皇后的一个放置方案输出到文件output.txt。文件的第一行是n个皇后的放置方案。

输入文件：

input.txt

5

output.txt

1 3 5 2 4

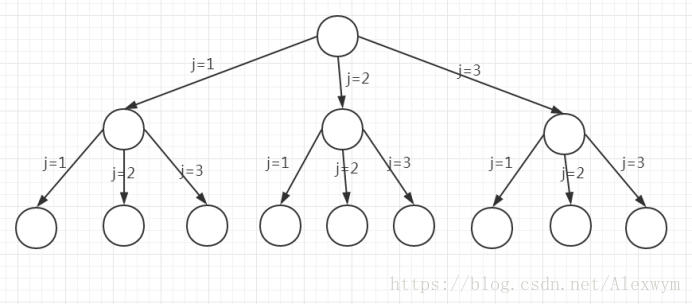
# 约束条件

对于每一个放置点而言，它需要考虑四个方向上是否已经存在皇后。分别是行列，45度斜线和135度斜线。

1. 行：每一行只放一个皇后，直到我们把最后一个皇后放到最后一行的合适位置，则算法结束。
2. 列：列相同的约束条件，只需判断j是否相等即可。
3. 45度斜线和135度斜线：约束条件——当前棋子和已放置好的棋子不能存在行数差的绝对值等于列数差的绝对值的情况，若存在则说明两个棋子在同一条斜线上

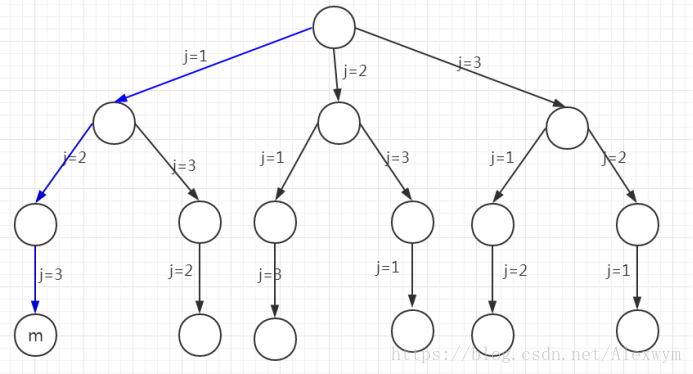
# 子集树

我们把行约束条件和其它约束条件放在一起，即认为每一行，棋子都有n个位置可以放置。于是它的空间结构如下



# 排列树

我们把行约束条件单独拿出来，也就是我们认为总共有八个皇后，这八个皇后必须都要放上去，但是放的位置不同，显然这是一个排列树。于是它的空间结构如下



# 排列树代码

|  |
| --- |
| #include<iostream>  #include<ctime>  using namespace std;  int num;  //剪枝函数  bool isOK(int n, int pieces[])  {  //判断当前状态是否合理，即皇后会不会互相攻击  for (int i = 1; i <= n-1; i++)  {  for (int j = i + 1; j <= n; j++)  {  int left = -(j - i);//向左的斜线  int right = (j - i);//向右的斜线  if (pieces[j] == pieces[i] + left||pieces[j] == pieces[i] + right)  {  //第i行皇后和第j行皇后会互相攻击  return false;  }  }  }  //所有皇后都不会互相攻击  return true;  }  void nQueen(int n, int t, int pieces[])  {  if (t > n)  {  //计数  num++;  //打印  for (int i = 1; i <= n; i++)  {  for (int j = 1; j < pieces[i]; j++)  cout << "- ";  cout << pieces[i]<<" ";  for (int j = pieces[i] + 1; j <= n; j++)  cout << "- ";  cout << endl;  }  cout << endl;  }  else  {  for (int i = t; i <= n; i++)  {  swap(pieces[t], pieces[i]);  //可以放，进入子节点  if (isOK(t, pieces)) nQueen(n, t + 1, pieces);  swap(pieces[t], pieces[i]);  }  }  }  int main()  {  int n;  cin >> n;  int \*pieces = new int[n + 1];  for (int i = 1; i <= n; i++) pieces[i] = i;  nQueen(n, 1, pieces);  cout << num << endl;  cout << "OK" << endl;  //system("pause");  } |

# 子集树代码

算了不写了。。。