# 回溯法实现四皇后问题

硬件一班 171491107 郭伟嘉

## 一：问题分析：

回溯法思路，就是将所有的结果变成一棵树，从树的结点开始访问，采用深度优先策略，从树的根结点开始访问，如果满足条件，继续访问下一层，如果不满足条件，返回上一个结点，继续访问其它结点。重复操作。

首先放置第一列，有四种放法，如果第一列，放置在第一个，再放置第二列，也有四种放法，很显然，第二列的第一种放法，不符合条件，这种放法下面所的子树也就不必访问了，节省了访问时间，然后继续第三列，第四列。就可以得到结果了。

## 二：试验代码：

**public** **class** AI {

**public** **static** **int** *Q*=4;//四皇后问题

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[][] dp=**new** **int**[*Q*][*Q*];

**int** i,j;

**for**(i=0;i<*Q*;i++)

{

**for**(j=0;j<*Q*;j++)

{

dp[i][j]=0;

}

}

*quenn*(0,dp);

}

**private** **static** **void** quenn(**int** m, **int**[][] dp) {

// **TODO** Auto-generated method stub

**if**(m==*Q*){//递归结束的条件

**for**(**int** i=0;i<*Q*;i++)

{

**for**(**int** j=0;j<*Q*;j++)

{

System.***out***.print(dp[i][j]+" ");

}

System.***out***.println("\n");

}

System.***out***.println("-------------------");

}

**for**(**int** i=0;i<*Q*;i++)

{

**if** (*isCorrt*(i,m,dp)) {

dp[i][m]=1;

*quenn*(m+1, dp);

dp[i][m]=0;

}

}

}

**private** **static** **boolean** isCorrt(**int** i, **int** j, **int**[][] dp) {

// **TODO** Auto-generated method stub

**int** s, t;

**for**(s=i,t=0; t<*Q*; t++)

**if**(dp[s][t]==1 && t!=j)

**return** **false**;

**for**(t=j,s=0; s<*Q*; s++)

**if**(dp[s][t]==1 && s!=i)

**return** **false**;

**for**(s=i-1,t=j-1; s>=0&&t>=0; s--,t--)

**if**(dp[s][t]==1)

**return** **false**;

**for**(s=i+1,t=j+1; s<*Q*&&t<*Q*;s++,t++)

**if**(dp[s][t]==1)

**return** **false**;

**for**(s=i-1,t=j+1; s>=0&&t<*Q*; s--,t++)

**if**(dp[s][t]==1)

**return** **false**;

**for**(s=i+1,t=j-1; s<*Q*&&t>=0; s++,t--)

**if**(dp[s][t]==1)

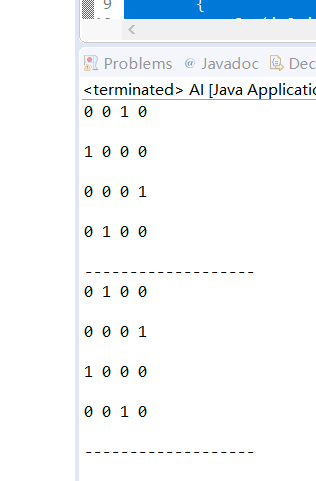
**return** **false**;

**return** **true**;

}

}

## 三：试验结果：



## 四.实验总结：

其实感觉回溯法有类似穷举，把所有的结果都列出来。只是回溯法，把结果变成了一棵树，如果在某个结点的时候，已经不满足条件了，那后面的选择肯定不满足条件，这样就节省了遍历所有的结果的时间。提高了效率。