回溯法实现四皇后问题

171491106 安南 硬件一班

1. 实验题目

回溯法实现四皇后问题

1. 算法简介

回溯法按深度优先策略搜索问题的解空间树。首先从根节点出发搜索解空间树，当算法搜索至解空间树的某一节点时，先利用**剪枝函数**判断该节点是否可行（即能得到问题的解）。如果不可行，则跳过对该节点为根的子树的搜索，逐层向其祖先节点回溯；否则，进入该子树，继续按深度优先策略搜索。回溯法的基本行为是搜索，搜索过程使用剪枝函数来为了避免无效的搜索。剪枝函数包括两类：1. 使用约束函数，剪去不满足约束条件的路径；2.使用限界函数，剪去不能得到最优解的路径。问题的关键在于如何定义问题的解空间，转化成树（即解空间树）。解空间树分为两种：子集树和排列树。两种在算法结构和思路上大体相同。

1. 实验目的

成功编写一个回溯算法，输出四皇后问题的所有答案。

1. 实验代码

**package** 四皇后;

**public** **class** 四皇后 {

**public** **static** **int** *Q*=4; //代表四皇后

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

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

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

**int** i,j;

//初始化

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

{

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

{

dp[i][j]=0;

}

}

*que*(0,dp);

}

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

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

**if**(m==*Q*){//递归结束条件

**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("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//递归计算

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

{

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

dp[i][m]=1;

*que*(m+1, dp);

dp[i][m]=0;

}

}

}

//判断这个位置能不能放皇后

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

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

**int** s, t; //s代表行,t代表列

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

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

**return** **false**;//判断行

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

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

**return** **false**;//判断列

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

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

**return** **false**;//判断左上方

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

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

**return** **false**;//判断右下方

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

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

**return** **false**;//判断右上方

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

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

**return** **false**;//判断左下方

**return** **true**;

}

}

1. 实验结果



六、实验总结

当问题是要求满足某种性质（约束条件）的所有解或最优解时，往往使用回溯法。