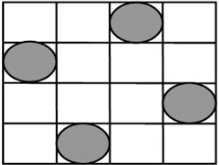
# 四皇后问题

回溯法算法思想：

回溯法的基本思想是：可以构建出一棵解空间树，通过探索这棵解空间树，可以得到四皇后问题的一种或几种解。这样的解空间树有四棵

# 问题：



在如上图所示的4×4的棋盘上，按列来摆放棋子，首先因为皇后棋子不能在同一列，所以先排除有2个或2个以上的棋子在同一列的情况，所以第一个棋子在第一列有4种摆放方法（第1列第1行，第1列第2行，第1列第3行，第1列第4行），同样第二个棋子在第二列有4种，同样第三个棋子在第三列有4种，同样第四个棋子在第四列有4种，所以进行简单的排除不在同一列的情况后，还有4×4×4×4=256种可能，但是在这256种可能里，依然存在比如棋子在同一行，或在45度斜线上的情况出现。另一个角度思考，所有的满足四皇后问题的摆放方式一定都存在于这256种情况之中。简单的理解就是：这256种棋盘局面包含了所有满足4皇后问题的解，但是不包含全部的棋盘局面。

代码：

#include<stdio.h>

int count = 0;

int isCorrect(int i, int j, int (\*Q)[4])

{

int s, t;

for(s=i,t=0; t<4; t++)

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

return 0;//判断行

for(t=j,s=0; s<4; s++)

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

return 0;//判断列

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

if(Q[s][t]==1)

return 0;//判断左上方

for(s=i+1,t=j+1; s<4&&t<4;s++,t++)

if(Q[s][t]==1)

return 0;//判断右下方

for(s=i-1,t=j+1; s>=0&&t<4; s--,t++)

if(Q[s][t]==1)

return 0;//判断右上方

for(s=i+1,t=j-1; s<4&&t>=0; s++,t--)

if(Q[s][t]==1)

return 0;//判断左下方

return 1;//否则返回

}

void Queue(int j, int (\*Q)[4])

{

int i,k;

if(j==4){//递归结束条件

for(i=0; i<4; i++){

//得到一个解，在屏幕上显示

for(k=0; k<4; k++)

printf("%d ", Q[i][k]);

printf("\n");

}

printf("\n");

count++;

return ;

}

for(i=0; i<4; i++){

if(isCorrect(i, j, Q)){//如果Q[i][j]可以放置皇后

Q[i][j]=1;//放置皇后

Queue(j+1, Q);//递归深度优先搜索解空间树

Q[i][j]=0;//这句代码就是实现回溯到上一层

}

}

}

int main()

{

int Q[4][4];

int i, j;

for(i=0; i<4; i++)

for(j=0; j<4; j++)

Q[i][j] = 0;

Queue(0, Q);

printf("The number of the answers are %d\n", count);

return 0;

}