**数据结构实验报告**

**实验五 求解皇后问题（递归法）**

学 生 姓 名 ：杨仕振

学 号 ：2015301500349

二○一六年十一月

**一、简介**

【问题描述】

请编写一个程序求解皇后问题：在n\*n的方格棋盘上，放置n个皇后，要求每个皇后不同行、不同列、不同左右对角线。由用户来输入皇后的个数，不低于4，不能超过20；要求输出所有的解。

【提示】

采用递归方法求解。

**二、算法说明**

主体结构与采用循环方式实现的结构相似，但是在最大的循环的内部使用了一个递归函数。

程序中定义了几个全局变量：

int n;//记录棋盘大小

int a;//记录当前位置横坐标

int b;//记录当前位置纵坐标

int i;//重复性检验时所使用的计数变量

stack \*s//储存路线的栈指针

程序中使用的数据结构如下：

typedef struct {

int x;

int y;

}chess;

“棋子”，用它在“棋盘”上的坐标来表示

typedef struct {

chess c[Maxsize];

int top;

}stack;

用来记录可行组合的栈

程序中用到的函数：

void printpath();//用以打印符合条件的路径

void push();//用以入栈（出栈用s->top—代替）

void loadpath();//进行扫描和判断是否可以入栈或打印的核心函数

基本思想：

假象一个n\*n的棋盘,从左向右为横轴，坐标为0~n-1；

从上而下为纵轴，坐标为0~n-1；以左上角为（0,0）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | .. | n-1 |
| 0 |  |  |  |
| .. |  |  |  |
| n-1 |  |  |  |

核心函数loadpath():

1. 先判断当前格子能否入栈
2. 如果能入栈，那么就将当前位置推向下一行，并将横坐标置0
3. 不能入栈的情况下，如果栈满，就将路线打印，如果当前位置达到了最右端或最顶端的格子，则返回栈内上一个位置，直到不触及边界或除了开始点之外的顶点，然后将当前位置向右一格。函数执行完完整的一次。

**三、测试结果**

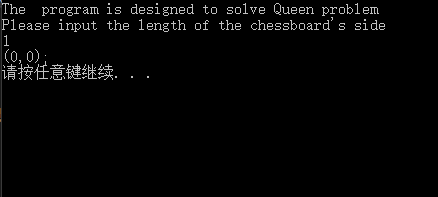


图1 当n=1时的输出

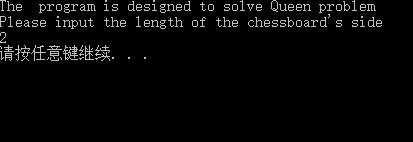


图2 当n=2时的输出

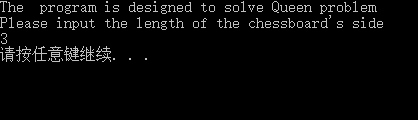


图3 当n=3时的输出

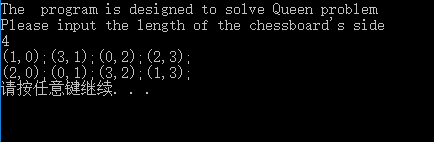


图4 当n=4时的输出

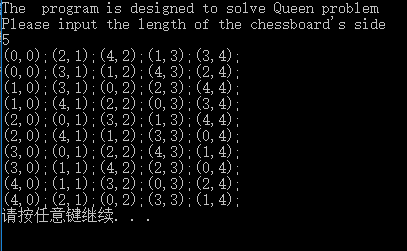


图5 当n=5时的输出

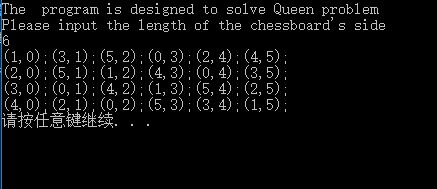


图6 当n=6时的输出

其中：

1输入的目的是为了检测只有一个棋子的特殊情况。

5输入的目的是为了检测当棋盘右下角在路径中时能否正常的输出和终止循环。

所有的输出均符合预期输出的坐标结果，均通过。

当n逐渐增大时，程序运行所耗费的时间将会急剧上升，结果也很多，这里就不一一列出了。

**四、分析与探讨**

循环最大只有两层嵌套，时间复杂度为O(n2),当无法入栈时结束递归函数重新循环，故空间复杂度最大为O(n2)，最小为O(n)。

**附录：源代码**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#define Maxsize 20

int n, a, b, i;

typedef struct {

int x;

int y;

}chess;//规定记录棋子的坐标方式

typedef struct {

chess c[Maxsize];

int top;

}stack;//储存棋子组合的栈

stack \*s;

void printpath()//打印组合

{

for (int j = 0; j<n; j++)

printf("(%d,%d);", s->c[j].x, s->c[j].y);

printf("\n");

}

void Push()//入栈操作

{

s->top++;

s->c[s->top].x = a;

s->c[s->top].y = b;

}

void loadpath()

{

for (i = 0; i <= s->top; i++)//判断当前的格子是否能够入栈

if (b==n||a == s->c[i].x || abs(s->c[i].x - a) == abs(s->c[i].y - b) || s->top == n - 1)//条件：不同对角线且不同列

break;//若冲突，则中止循环

if (i > s->top)

{

Push();

b++;//推向下一行

a = 0;//将横坐标归为0

loadpath();

}

else

{

if (s->top == n - 1)

printpath();//打印组合

while (a == n - 1 && b != 0||b==n)//当达到了最右端或最顶端的格子（不包含第一行的情况）

{

a = s->c[s->top].x;

b = s->c[s->top].y;//将当前格子置为栈顶格子

s->top--;//出栈

}

a++;//将当前位置置为当前格子右侧位置

}

}

int main()

{

a = b = 0;

s = (stack \*)malloc(sizeof(stack));

s->top = -1;

printf("The program is designed to solve Queen problem\n");

printf("Please input the length of the chessboard's side(larger than zero)\n");

do {

scanf\_s("%d", &n);

} while (n<1);

while (!((s->top == -1) && (a == n) && (b == 0)))//判断是否已经扫描完了所有应扫描的格子

loadpath();

system("PAUSE");

return 0;

}