棋盘覆盖实验报告

19122003 陶立明

一.算法思想及动态方程

1.算法思想

当k>0时，找到棋盘的中心点，然后将棋盘划分为4个小棋盘。通过将特殊方格左上角的（x,y）坐标与棋盘中心点的（x,y）坐标对比，判断特殊方格存在于哪一个象限。然后对棋盘中心点附近的三个象限的三个方格实施L型骨牌覆盖。之后，四个小棋盘都拥有了特殊方格，再依次递归四个象限的小棋盘。即一个大问题划分为了四个相同的小问题。依次递归即可得到每个小问题的解，从底向上可以得到大问题的解。满足[分治](https://so.csdn.net/so/search?q=%E5%88%86%E6%B2%BB&spm=1001.2101.3001.7020)策略。

2.动态方程

二.程序分析及测试情况

1.关键函数

void chessboard(int x1,int y1,int x2,int y2,int index\_x,int index\_y,int side){//形参为，棋盘左上角，右下角，及特殊方格坐标

if (side < 2) return;//棋盘长度小于2 x 2时结束递归

int area = 0, xmid = (x1 + x2) / 2, ymid = (y1 + y2) / 2; //计算棋盘中心点坐标，xmid ymid

if (index\_x < xmid) {

if (index\_y < ymid) {

area = 1;

}

else

area = 3;

}

else if (index\_y < ymid) {

area = 2;

}

else

area = 4; //计算特殊方格所在象限

Sleep(1000);//睡眠函数

switch (area)

{

case 1:

setfillcolor(GREEN);

fillrectangle(xmid, ymid - length, xmid + length, ymid);//填充第二象限

fillrectangle(xmid - length, ymid, xmid, ymid + length);//填充第三象限

fillrectangle(xmid, ymid, xmid + length, ymid + length); //填充第四象限

chessboard(x1, y1, xmid, ymid, index\_x, index\_y, side / 2);//递归第一象限

chessboard(xmid, y1, x2, ymid, xmid, ymid - length, side / 2);//递归第二象限

chessboard(x1, ymid, xmid, y2, xmid-length, ymid, side / 2);//递归第三象限

chessboard(xmid, ymid, x2, y2, xmid , ymid, side / 2);//递归第四象限

break;

case 2:

setfillcolor(RED);

fillrectangle(xmid - length, ymid - length, xmid, ymid);//填充第一象限

fillrectangle(xmid - length, ymid, xmid, ymid + length);//填充第三象限

fillrectangle(xmid, ymid, xmid + length, ymid + length); //填充第四象限

chessboard(x1, y1, xmid, ymid, xmid-length,ymid-length, side / 2);//递归第一象限

chessboard(xmid, y1, x2, ymid, index\_x, index\_y, side / 2);//递归第二象限

chessboard(x1, ymid, xmid, y2, xmid - length, ymid, side / 2);//递归第三象限

chessboard(xmid, ymid, x2, y2, xmid, ymid, side / 2);//递归第四象限

break;

case 3:

setfillcolor(WHITE);

fillrectangle(xmid - length, ymid - length, xmid, ymid);//填充第一象限

fillrectangle(xmid, ymid - length, xmid + length, ymid);//填充第二象限

fillrectangle(xmid, ymid, xmid + length, ymid + length); //填充第四象限

chessboard(x1, y1, xmid, ymid, xmid - length, ymid - length, side / 2);//递归第一象限

chessboard(xmid, y1, x2, ymid, xmid, ymid - length, side / 2);//递归第二象限

chessboard(x1, ymid, xmid, y2, index\_x, index\_y, side / 2);//递归第三象限

chessboard(xmid, ymid, x2, y2, xmid, ymid, side / 2);//递归第四象限

break;

case 4:

setfillcolor(BLUE);

fillrectangle(xmid - length, ymid - length, xmid, ymid);//填充第一象限

fillrectangle(xmid, ymid - length, xmid + length, ymid);//填充第二象限

fillrectangle(xmid - length, ymid, xmid, ymid + length);//填充第三象限

chessboard(x1, y1, xmid, ymid, xmid - length, ymid - length, side / 2);//递归第一象限

chessboard(xmid, y1, x2, ymid, xmid, ymid - length, side / 2);//递归第二象限

chessboard(x1, ymid, xmid, y2, xmid - length, ymid, side / 2);//递归第三象限

chessboard(xmid, ymid, x2, y2, index\_x, index\_y, side / 2);//递归第四象限

break;}

}

**分析：利用erasyx库的函数来填充棋盘方格。以特殊方格在第一象限为例，我们要填充中心点旁二三四象限的方格，（注意他们的x,y坐标是不同的），然后递归第一二三四象限的小棋盘（注意小棋盘的左上角，右下角，及特殊方格的坐标是不同的）**

**主函数：**

int main()

{

while (1) { //永真循环，多次实验

int k, x, y, side;

cout << "请输入数据：" << endl;

cin >> k >> x >> y;

side = pow(2, k);

initgraph(1000, 1000, EX\_SHOWCONSOLE);

for (int i = 0; i < side; i++)

{

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

{

rectangle(length \* j, length \* i, length \* (j + 1), length \* (i + 1));

}

} //绘制棋盘

setfillcolor(WHITE);

fillrectangle(x \* length, y \* length, (x + 1) \* length, (y + 1) \* length);

chessboard(0, 0, side \* length, side \* length, x \* length, y \* length, side);

Sleep(500);

closegraph();

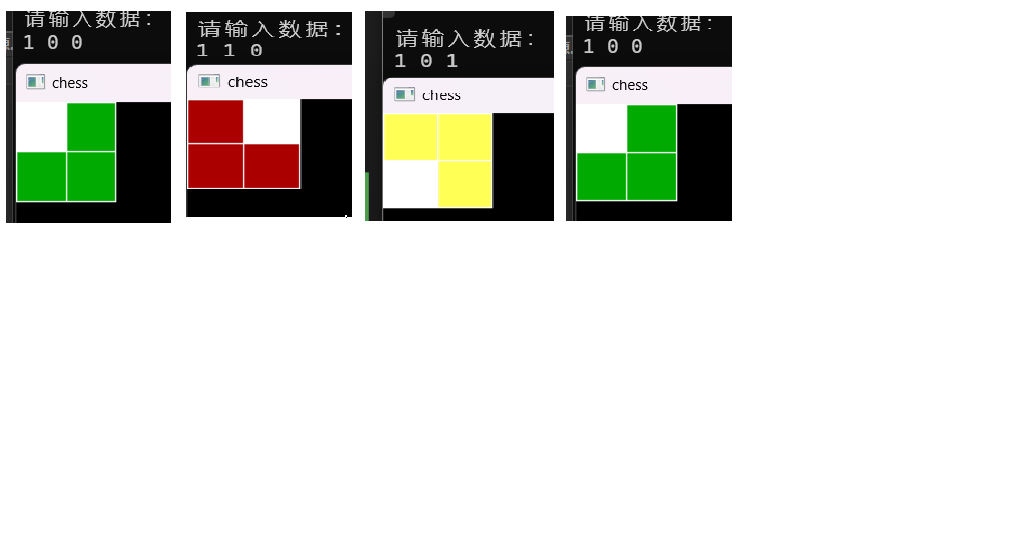
}

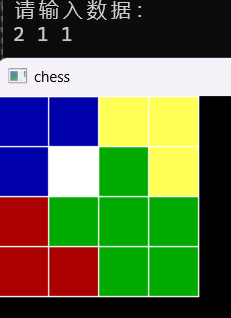
return 0;

}

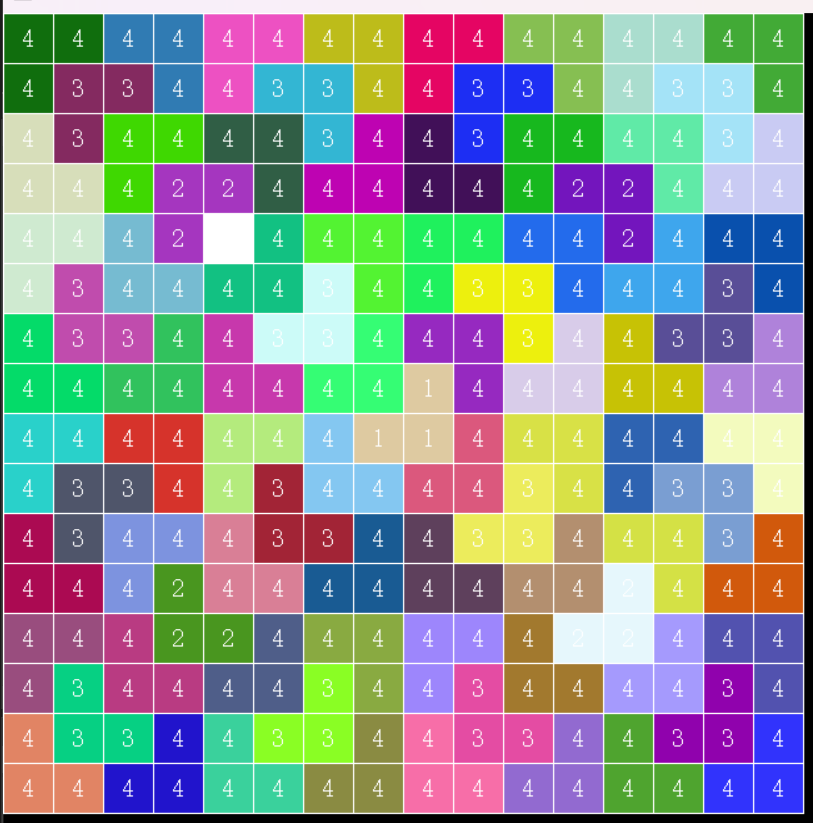
2.测试结果

特殊方格为白色





下面测试数据为 4 4 4，因为棋盘较大，**对代码改进一下（填充颜色随机，在骨牌上标注数字，棋盘方格大小可变）**，使骨牌覆盖更加清晰可辩



三.问题及体会。

整个实验在做的时候逻辑是很顺的，算法思想也不能说难，但在实际编写起来还是“困难重重”，主要原因是对easyx库的函数不太熟悉，然后定义的递归函数chessboard的形参有六个，棋盘左上角，右下角，及特殊方格坐标，因为对easyx的xy坐标不熟悉所以频频填错。