**华南农业大学**

**《算法分析与设计》课程实验**

**实验2：棋盘覆盖**

年级专业： 21信息与计算科学

学生姓名： 吴嘉豪

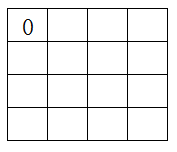
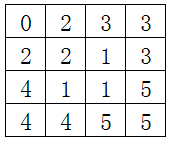
学生学号： 202125810321

指导老师： 赵峰

实验时间： 2024年04月15日

一、实验内容

在一个的棋盘上，有一个0号方格。用分治法设计算法、编写程序，用L型骨牌覆盖满除0号方格外的所有其它方格，且骨牌不得重叠（可以旋转放置）。0号方格标记为0，被第一个L型骨牌覆盖的三个方格标记为1，被第二个L型骨牌覆盖的三个方格标记为2，...。记录运行时间，输出覆盖结果。

棋盘示例 骨牌示例 覆盖结果示例

二、实验步骤

1、用分治法设计算法、编写程序；

2、针对三种取值，随机生成0号方格位置，运行程序生成覆盖结果。每个值运行程序两次，记录运行时间，对覆盖结果截图，记录进实验报告；

3、推导算法的时间复杂度，分析：

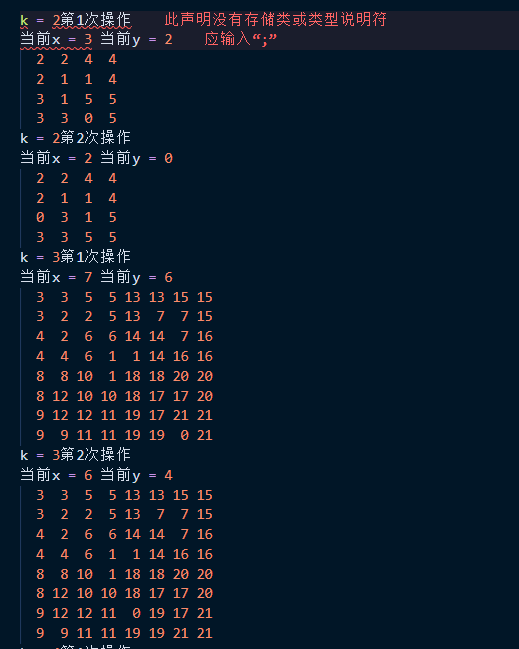
（1）时间复杂度是否与实际运行时间一致；

（2）存不存在复杂度更低的算法；

4、记录实验中遇到的问题与错误。

三、实验结果

2.



3

1. 实际上 ,除了分治的过程， 到最后 都是再填充， 所以时间复杂度 就是方格的个数的平方， 假设现在 为k，那么 有n = 2^k， 那么 时间复杂度就是O(4 ^ k) .当前还有一些 函数创建的开销，但是我不考虑再内

推导过程为 T(n) = 4 \* T(n / 4) 且 n == 4 的 时候 遍历 4次 所以可以推出上面的式子。

1. 我认为是不存在的， 因为为了实现这个算法， 我们最后需要给每一个元素都填充值。 那么就意味着要遍历所有的方格， 所以时间复杂度最低是O(4 ^ n)

4.

再编码的过程中， 因为把象限之间的关系 和 x 和 y的 关系搞乱了， 导致一开始程序 跑出来的结果有点问题。 不过后来排查出来了

背景图案

描述已自动生成

四、源程序代码

#*include*<iostream>

#*include*<string>

#*include*<algorithm>

#*include*<vector>

#*include*<functional>

#*include* <cstdlib>

#*include*<map>

#*include*<set>

#*include* <iomanip>

#*include*<cstdio>

#*include*<cstring>

#*include* <ctime>

using namespace std;

int k = 0;

pair<int,int> *getFourIndex* (int x, int y) {

*return* *make\_pair*((x / 2), (y / 2));

}

vector<vector<int>>graph;

void *solve*() {

    int n = (1 << k);

// *分配一个初始空间*

    graph.*resize*(n);

*for*(int i = 0; i < n; ++i){

        graph*[*i*]* *=* *vector*<int>(n, -1);

    }

    int zeroX,   zeroY;

    zeroX = *rand*() % n;// *随机生成 x， y*

    zeroY = *rand*() % n;

// *添加0的位置*

    graph*[*zeroX*][*zeroY*]* = 0;

    int nowTag = 1;

// *查找象限*

    function<void(int ,int,int, int)> fillempty = [&](int left, int right, int up , int down){

*for*(int i = left ; i <= right; ++i){

*for*(int j = up; j <= down; ++j){

*if*(graph*[*i*][*j*]* == -1) {

                    graph*[*i*][*j*]* = nowTag;

                }

            }

        }

        nowTag++;

    };

    function<*vector*<pair<int,int>>(int, int, int, int)> fillDimension = [&](int x, int y, int curX, int curY){

        vector<pair<int,int>> nums;

// *第一象限判断*

*if*(curX <= x && curY <= y) {

            nums.*push\_back*(*make\_pair*(curX, curY));

        }

*else*{

            nums.*push\_back*(*make\_pair*(x, y));

            graph*[*x*][*y*]* = nowTag;

        }

// *第二象限判断*

*if*(curX <= x && curY >= y + 1){

            nums.*push\_back*(*make\_pair*(curX, curY));

        }

*else*{

            nums.*push\_back*(*make\_pair*(x , y + 1));

            graph*[*x*][*y + 1*]* = nowTag;

        }

// *第三象限判断*

*if*(curX >= x + 1 && curY <= y){

            nums.*push\_back*(*make\_pair*(curX, curY));

        }

*else* {

            nums.*push\_back*(*make\_pair*(x + 1, y));

            graph*[*x + 1*][*y*]* = nowTag;

        }

// *第四象限*

*if*(curX >= x + 1 && curY >= y + 1){

            nums.*push\_back*(*make\_pair*(curX, curY));

        }

*else*{

            nums.*push\_back*(*make\_pair*(x + 1, y + 1));

            graph*[*x + 1*][*y + 1*]* = nowTag;

        }

        nowTag++;

// *fill*

*return* nums;

    };

// *left right up down*

    function<void(int ,int, int ,int, int ,int)> div = [&](int left, int right, int up, int down, int curX, int curY){

// *如果当前区块的长度为 2 的 时候 就 终止*

*if*(right - left == 1 && down - up == 1){

*fillempty(*left, right, up, down*)*;

*return* ;

        }

// *它们的 中轴*

        int midx = (left + right)>>1;

        int midy = (up + down)>>1;

// *给每个象限的 进行一个 填充 获取到一个数组， 存入 第一象限 第二 第三 第四*

        auto nums =  *fillDimension(*midy, midx, curX, curY*)*;

// *分成四个区块*

*div(*left, midx, up, midy, nums*[*0*]*.first, nums*[*0*]*.second*)*;// *第一象限*

*div(*midx + 1, right , up, midy, nums*[*1*]*.first, nums*[*1*]*.second*)*;// *第二象限*

*div(*left, midx, midy + 1, down, nums*[*2*]*.first, nums*[*2*]*.second*)*;// *第三象限*

*div(*midx + 1, right, midy + 1, down, nums*[*3*]*.first, nums*[*3*]*.second*)*;// *第四象限*

    };

    function<void()> printGraph = [&](){

        cout*<<*"当前x = "*<<*zeroX*<<*" 当前y = "*<<*zeroY*<<endl*;

*for*(int i = 0; i < n; ++i){

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

                cout*<<setw*(3)*<<*graph*[*i*][*j*]*;

            }

            cout*<<endl*;

        }

    };

*div(*0, n - 1, 0, n -1, zeroX, zeroY*)*;

*printGraph()*;

}

int *main*(){

    int t;

    t = 5;

*while*((t--) > 2){

*srand*(*time*(nullptr));

        k = 6 - t;

        cout*<<*"k = "*<<*k*<<*"第1次操作"*<<endl*;

*solve*();

        cout*<<*"k = "*<<*k*<<*"第2次操作"*<<endl*;

*solve*();

    }

}

五、评语及评分

|  |
| --- |
| 评语及评分  评阅人签名： |