作业

- 哨兵布置问题。一个博物馆由排成m*n个矩阵阵列的陈列室组成，需要在陈列室中设立哨位，每个哨位上的哨兵除了可以监视自己所在陈列室外，还可以监视他上、下、左、右四个陈列室，试基于分支限界法给出一个最佳哨位安排方法，使得所有陈列室都在监视之下，但使用的哨兵最少。
- 数据输入：由文件input.txt给出输入数据。第1行有2个正整数m和n。
- 结果输出：将计算出的哨兵人数及其最佳哨位安排输出到文件output.txt。文件的第1行是哨兵人数，接下来的m行中每行n个数，0表示无哨位，1表示哨位。
- 输入示例：m=n=4.

源代码如下所示（具体思路和分析均已包含在注释中）：

输出截图如下所示：

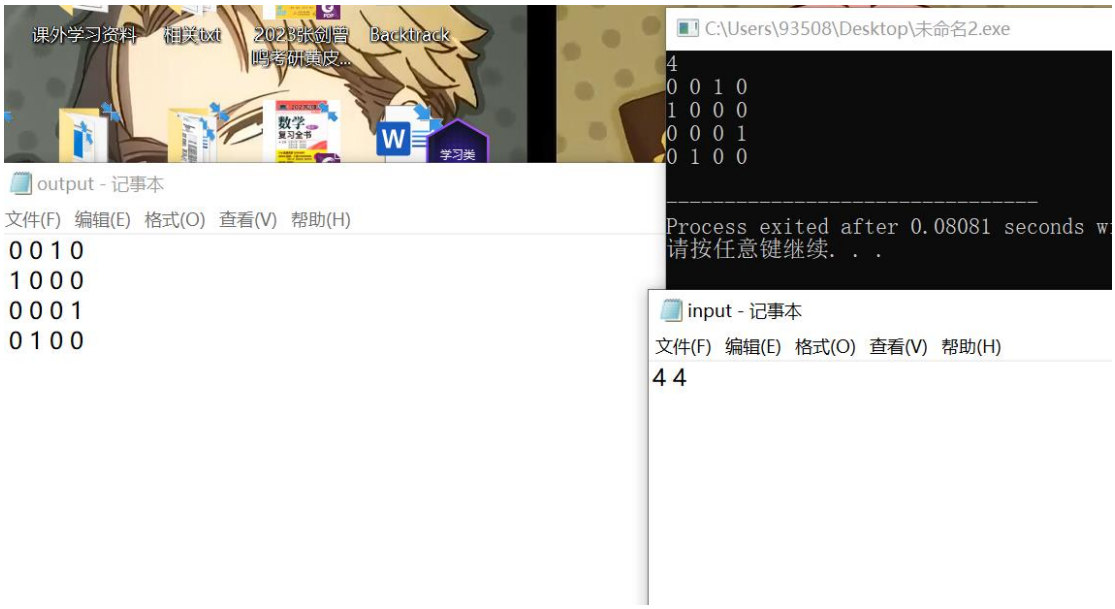
```
#include <iostream>
#include <string>
using namespace std;

//行数和列数
int m,n;
//位置数组（自身+上下左右）
int f[5][2] = { {0,0} , {0,1} , {0,-1} , {1,0} , {-1,0} };
//结果数组（哨兵位置）
int res[30][30];
//哨兵个数
int ans;
//暂时存储哨兵位置
int sentinel[30][30];
//暂时存储哨兵个数
int p;
//哨兵能看到的位置
int vision[30][30];
//被监视的陈列室个数
int visions;
//函数声明
void Sentinel(int x,int y,int c,int d);
```

```
void search(int i,int j);

//主函数
int main()
{
    //读取 input.txt
    FILE* file1 = fopen ( "C:\\Users\\93508\\Desktop\\input.txt" ,
"r" );
    if (file1 == NULL){
        cout << "文件读取失败！" << endl;
        return 0;
    }
    fscanf ( file1 , "%d%d" , &m , &n );
    fclose ( file1 );

    //剪枝，哨兵个数不会超过(n*m/3+1)个
    ans = n*m/3 + 2;
    p = 0;
    //设置边界
    for(int i = 0 ; i <= m + 1 ; i++){
        vision[i][0] = vision[i][n+1] = 1;
    }

    for(int i = 0 ; i <= n + 1 ; i++){
        vision[0][i] = vision[m+1][i] = 1;
    }
    //开始搜索
    search(1,1);
    //输出结果并写入 output.txt
    cout << ans << endl;
    FILE* file2 = fopen ( "C:\\Users\\93508\\Desktop\\output.txt" ,
"w" );
    for(int i = 1; i <= m ; i++){
        for(int j = 1 ; j <= n ; j++){
            cout << res[i][j] << ' ';
            fprintf ( file2,"%2d" , res[i][j] );
        }
        cout << endl;
        fprintf (file2, "\n" );
    }
    fclose ( file2 );
}

//搜索位置
```

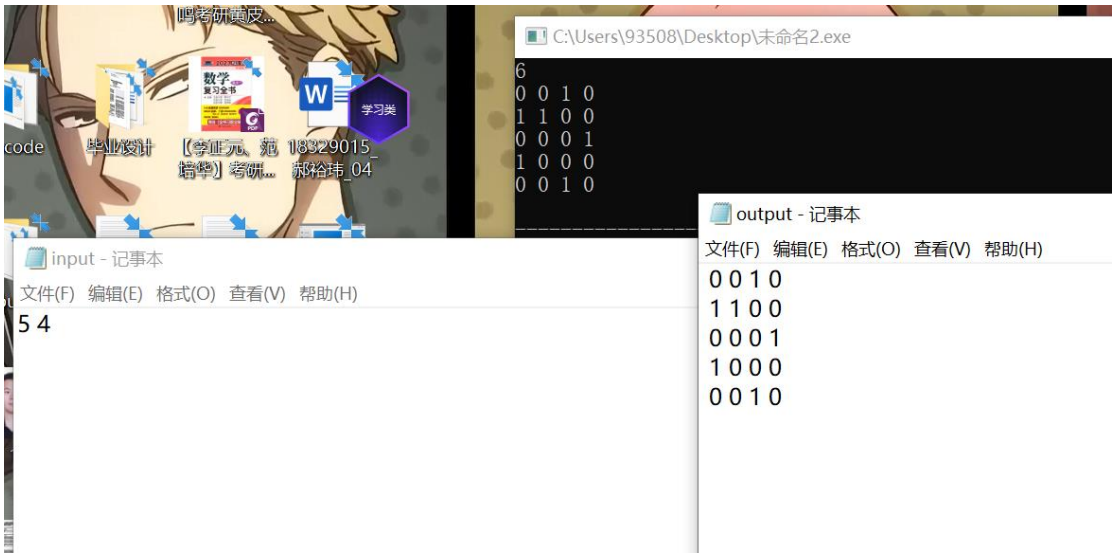
```
void search(int i,int j)
{
    if(p >= ans){
        return;
    }
    //该位置若已被某个哨兵观察到则跳过
    while(i <= m && vision[i][j]!=0 ) {
        j++;
        //换行
        if(j > n){
            i++;
            j = 1;
        }
    }
    //更新答案
    if(i>m) {
        ans = p;
        //更新后的答案复制到最终结果数组 res 中
        memcpy(res, sentinel, sizeof(sentinel));
        return;
    }

    if(i < m){
        Sentinel(i+1,j,i,j);
    }
    if(vision[i][j+1] == 0){
        Sentinel(i,j,i,j);
    }
    if(j < n && (vision[i][j+1] == 0 || vision[i][j+2] == 0)){
        Sentinel(i,j+1,i,j);
    }
}

void Sentinel(int x,int y,int c,int d)
{
    sentinel[x][y] = 1;
    p++;
    for(int i=0 ; i<=4; i++){
        int xx = x + f[i][0];
        int yy = y + f[i][1];
        vision[xx][yy]++;
        if(vision[xx][yy] == 1){
            visions++;
        }
```

```
    }

    search(c,d+1);

    sentinel[x][y] = 0;
    p--;
    for(int i=0 ; i<=4 ; i++){
        int xx = x + f[i][0];
        int yy = y + f[i][1];
        vision[xx][yy]--;
        if(vision[xx][yy]==0){
            visions--;
        }
    }
}
```

样例 1：



样例 2：


C:\Users\93508\Desktop\未命名2.exe
6
0 0 1 0
1 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
input - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
5 4
output - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
0 0 1 0
1 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0

样例 3：


C:\Users\93508\Desktop\未命名2.exe
6
0 0 1 0 0
1 0 0 0 1
0 0 0 0 0
0 1 1 1 0
input - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
4 5
output - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
0 0 1 0 0
1 0 0 0 1
0 0 0 0 0
0 1 1 1 0