分类	任务名称	起止时间
分析	阅读雨课堂开发任务	0:00 --3:00
任务1	编写LRC算法	3:00 -- 15:00
任务2	编写Possible Number 算法	15:00 -- 18:00
任务3	编写测试功能代码（test.cpp）	18:00 -- 26:49
子任务3.1	编写test.cpp 输入部分	18:00 -- 21:00
子任务3.2	编写test.cpp 重复使用某算法、直至无法填入数字	21:00 -- 23:00
子任务3.3	编写test.cpp 判断数独是否完成，输出结果	23:00 -- 26:49
任务4	使用测试用例一	26:49 -- 29:30
任务5	访问数独wiki网站获取下一个测试用例	29:30 -- 31:00
任务6	使用第二个测试用例，特别观察Possible Number算法的输出	31:00 -- 33:00

备注

一、**分析以及任务1~任务2由于录屏工具意外关闭而丢失。** 代码编写过程重复就丢失了意义，因此没有重复。（已经跟助教报备）

二、任务2完成速度相较于任务1尤其快，是因为两者在代码中有大量重合部分，只需要修改核心逻辑就可以完成。(在代码展示的注释中会体现)

三、工程中刻意避免了使用AI

代码展示

以下为 $LRC.hpp$, 代码的算法事先部分，对于任务1~任务2

```cpp
#include<cstdio>
#include<array>
#include<vector>
#include<cstring>
using std::array;

using std::vector;

array<array<vector<int> , 9> , 9> LastRemainingCellInference(array<array<int , 9> , 9> mat)
{
    bool vis[9][9][10];
    memset(vis , 0 , sizeof vis); // vis数组表示每个格子上1~9是否允许填写。True 不允许 ， False 则
    // 以下为扫描每个格子，计算vis；LRC算法与 Possible Number 算法 都有这个公共部分， 编写过程中直接
    for(int i = 0;i < 9;i ++)
    {
        for(int j = 0;j < 9;j ++)
            if(mat[i][j] != 0)
            {
                int id = mat[i][j];
                for(int k = 0;k < 9;k ++)
                    vis[i][k][id] = vis[k][j][id] = 1;

                int L = i  / 3 , R = j / 3;
                for(int k = 0;k < 3;k ++)
                    for(int l = 0;l < 3;l ++)
                        vis[L * 3 + k][R * 3 + l][id] = 1;

                for(int k = 1;k <= 9;k ++) vis[i][j][k] = 1;
            }
    }

    array<array<vector<int> , 9> , 9> ret;

    for(int i = 0;i < 9;i ++)
        for(int j = 0;j < 9;j ++)
            if(mat[i][j] == 0)
            {
                for(int k = 1;k <= 9;k ++)
                    if(vis[i][j][k] == 0)
                    {
                        bool flag = 1;
                        for(int l = 0;l < 9;l ++)
                            if(l != j && vis[i][l][k] == 0) flag = 0; //对同一行判断
                        if(flag == 1)
                        {
                            ret[i][j].push_back(k);
```

```
                        break;
                    }

                    flag = 1;
                    for(int l = 0;l < 9;l ++)
                        if(l != i && vis[l][j][k] == 0) flag = 0; //对同一列判断
                    if(flag == 1)
                    {
                        ret[i][j].push_back(k);
                        break;
                    }

                    flag = 1;
                    int L = i / 3 , R = j / 3; // 对同一九宫格判断

                    for(int a = 0;a < 3;a ++)
                        for(int b = 0;b < 3;b ++)
                        {
                            int dx = L * 3 + a , dy = R * 3 + b;
                            if(dx == i && dy == j) continue;
                            if(vis[dx][dy][k] == 0) flag = 0;
                        }
                    if(flag == 1)
                    {
                        ret[i][j].push_back(k);
                        break;
                    }
                }
            }

    return ret;
}

array<array<vector<int> , 9> , 9> PossibleNumberInference(array<array<int , 9> , 9> mat)
{
    bool vis[9][9][10];
    memset(vis , 0 , sizeof vis); // 两个算法复用部分

    for(int i = 0;i < 9;i ++)
    {
        for(int j = 0;j < 9;j ++)
            if(mat[i][j] != 0)
            {
                int id = mat[i][j];
                for(int k = 0;k < 9;k ++)
                    vis[i][k][id] = vis[k][j][id] = 1;
```

```
                int L = i  / 3 , R = j / 3;
                for(int k = 0;k < 3;k ++)
                    for(int l = 0;l < 3;l ++)
                        vis[L * 3 + k][R * 3 + l][id] = 1;

                for(int k = 1;k <= 9;k ++) vis[i][j][k] = 1;
            }
    }

    array<array<vector<int> , 9> , 9> ret; //实际上，vis数组本质就是计算出 Possible Number 算法的

    for(int i = 0;i < 9;i ++)
        for(int j = 0;j < 9;j ++)
            if(mat[i][j] == 0)
            {
                for(int k = 1;k <= 9;k ++)
                    if(vis[i][j][k] == 0)
                        ret[i][j].push_back(k);
            }
    return ret;
}
```

以下为 $test.cpp$， 对应任务3~6部分。

```cpp
#include"LRC.hpp"
#include<cstdio>
#include<iostream>
#define Matrix array<array<int , 9> , 9>
#define MatrixMap array<array<vector<int> , 9> , 9>

using std::cin;
using std::cout;
using std::endl;

Matrix input;

void test(MatrixMap (*f)(Matrix))
{
    Matrix mat = input;
    while(true) \\重复使用某算法，直到无法确定任何一个格子
    {
        int flag = 0;
        MatrixMap ret = f(mat);
        for(int i = 0;i < 9;i ++)
            for(int j = 0;j < 9;j ++)
                if(ret[i][j].size() == 1)
                    mat[i][j] = *ret[i][j].begin() , flag = 1; \\候选集只有一个数字，那么格子上就
        if(!flag) break;
    }

    bool isfull = 1;
    for(int i = 0;i < 9;i ++)
        for(int j = 0;j < 9;j ++)
            if(mat[i][j] == 0) {isfull = 0;break;}
    \\分情况输出调试信息或者结果
    if(isfull)
    {
        cout << "puzzle completed" << endl;
        for(int i = 0;i < 9;i ++ , cout << endl)
            for(int j = 0;j < 9;j ++)
                cout << mat[i][j] << ' ';
    }
    else
    {
        cout << "puzzle incompleted" << endl;
        for(int i = 0;i < 9;i ++ , cout << endl)
            for(int j = 0;j < 9;j ++)
                cout << mat[i][j] << ' ';
        cout << "last output" << endl;
        MatrixMap ret = f(mat);
```

```
        for(int i = 0;i < 9;i ++ , cout << endl)
            for(int j = 0;j < 9;j ++)
            {
                cout << '[';
                for(auto v:ret[i][j]) cout << v << ' ';
                cout << ']';
            }
    }
}

int main()
{
    freopen("test.in","r",stdin); \\使用文件输入获得测试用例

    for(int i = 0;i < 9;i ++)
        for(int j = 0;j < 9;j ++)
            cin >> input[i][j];

    test(LastRemainingCellInference);\\使用函数指针方便重复测试两个方法
    test(PossibleNumberInference);
    return 0;
}
```