愉快的代码【8.6】

【7.27】我们特意加入了题解系统，也就是说，姐姐也会重新做一遍给你们的题目（包括选做），然后在第二天的题目前给出姐姐自己的代码和注释作为题解或参考

如果觉得自己的代码略为臃肿，可以参考对比一下姐姐的代码；

如果觉得姐姐的代码不如自己的优秀，也可以尽情地嘲讽姐姐～

【7.30】我们特意加入了团队系统，因为感觉到你们有点像是独立学习的样子，比如说姐姐和你们之间有交流，但是你们之间有没有交流呢姐姐就感受不到啦

所以正好在洛谷上发现了一个团队系统，我们可以在这上面布置作业呀（当然姐姐也会继续以 pdf 形式布置作业，你们也还是要以 pdf 形式交作业哈），然后你们就可以在上面看到其它妹妹们的代码呀（包括 AC 代码和还未 AC 的整个过程的代码和分数呀），觉得她们表现不够自己好的话，就可以在群里尽情地嘲笑她们呀～

然后那上面还有一个比赛功能哇，具体形式和我们平时的机考差不多，暑假差不多结束了我们也会有一次期末模拟机考的哈～

如果你们开心的话，你们也可以联合起来给姐姐布置一次平时的作业呀，或者给姐姐安排一次机考呀，你们都是团队的管理员了哈

【8.1】准备给你们留个有趣的团队大作业：给姐姐安排一次机考～

具体时间、题数、难度、知识点待定～

【8.3】经过了某些人性与道德的思考，得出了一个奇怪的想法：

“我今天把代码解决了，明天姐姐的代码还有兴趣看嘛”

那就当天放出来好啦～

同样地：如果觉得姐姐的代码不如自己的优秀，也可以尽情地嘲讽姐姐～

今天的题目：

知识点：DP

今天正式开始 DP 的学习～

根据 LeetCode 官号对 DP 的解释：

力扣（LeetCode）
已认证的官方帐号

550 人赞同了该回答

动态规划问题一直是大厂面试时最频繁出现的算法题，主要原因在于此类问题灵活度高，思维难度大，没有很明显的套路做法。

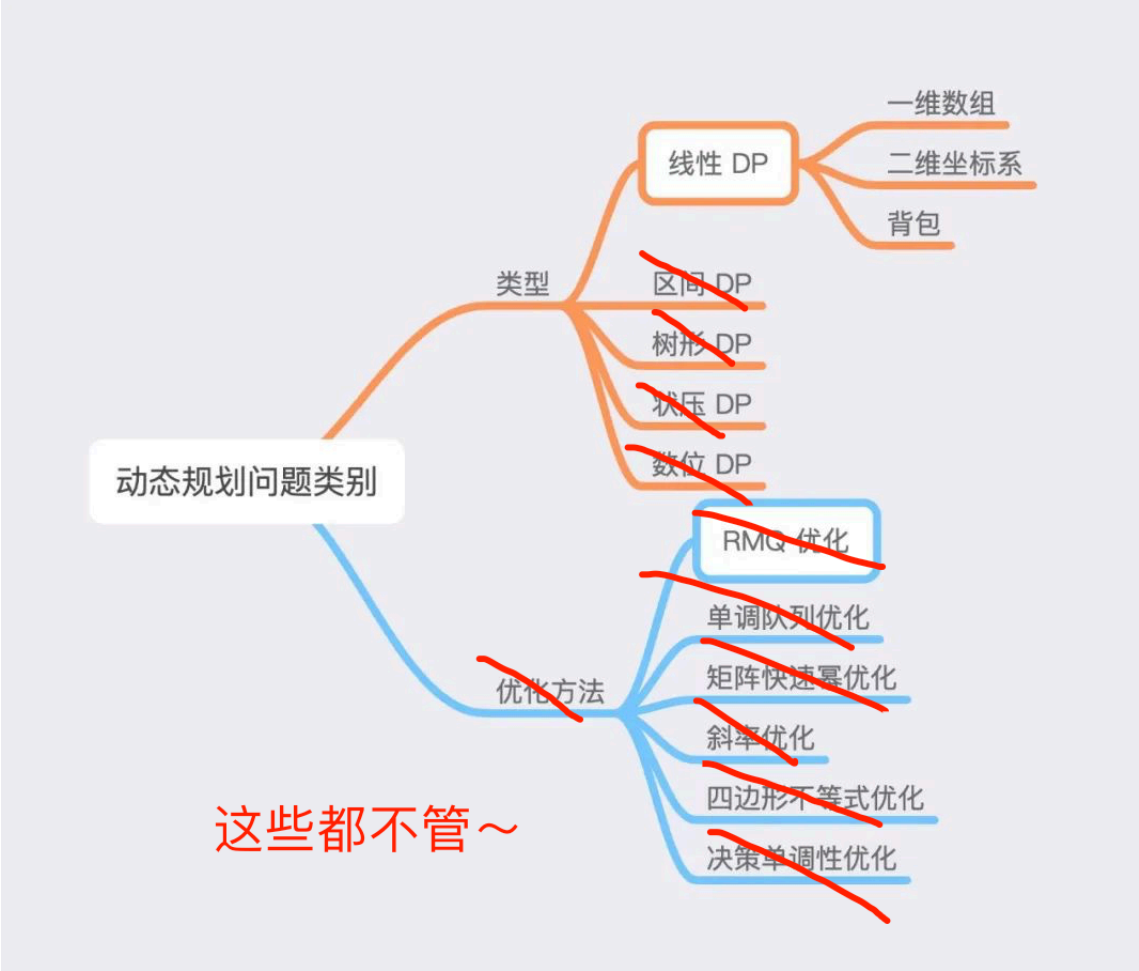
~~所以请尝试自学 DP~~

再从 LeetCode 那要来 DP 的分类图后

我们这个假期主要的目标就是“ 线性 DP ”啦

根据个人的难度理解：二维 < 一维 < 背包

所以今天先从二维做起～



1、https://www.luogu.com.cn/problem/P1002

2、https://www.luogu.com.cn/problem/P1130

今天的答案：

8.6问题1：

```
/*
    洛谷P1002：过河卒
    思想：可以发现，这也是一个迷宫题哦，但是由于这个迷宫比较空，如果用搜索的话，搜索约2^(20+20)次会TLE（虽然20挺小的，但是还是会TLE），因此考虑用 DP 来做。
    需要注意的是，这里卒的走向是只能往右和往下，因此满足 DP 的无后效性，转移方程为 dp[i][j] = dp[i-1][j]+dp[i][j-1]，注意控制点的方案为0即可
    时间复杂度：O(nm)
```

```
*/

#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;

int main() {
    int i,j,k,x,y,x2,y2;
    cin >> x >> y >> x2 >> y2;
    ll dp[x+2][y+2];
    x2++; y2++; // 原来是从0开始计数，修改成从1开始，好做
    int c[9] = {x2-1,x2-1,x2-2,x2-2,x2+1,x2+1,x2+2,x2+2,x2}; // 马的九个控制点
    int d[9] = {y2-2,y2+2,y2-1,y2+1,y2-2,y2+2,y2-1,y2+1,y2};
    for (i=0; i<x+2; i++)
        for (j=0; j<y+2; j++)
            dp[i][j] = 0;
    dp[1][0] = 1; // 初始状态
    for (i=1; i<x+2; i++) {
        for (j=1; j<y+2; j++) {
            dp[i][j] = dp[i-1][j]+dp[i][j-1]; // DP/递推
            for (k=0; k<9; k++)
                if(i == c[k] and j == d[k]) dp[i][j] = 0; // 如果是控制点，置0
        }
    }
    cout << dp[x+1][y+1] << endl;
}
```

8.6问题2：

```
/*
    洛谷P1130：红牌
    思想：比较明显的递推，这个小组可以从这个小组前一步或者上一个小组前一步转移过来，因此转移方程为 dp[i][j] = min(dp[i][j-1],dp[(i+n-1)%n][j-1])+a[i][j]
    时间复杂度：O(nm)
*/

#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;

int main() {
    int i,j,n,m,ans=99999999;
    cin >> m >> n; // 需要注意的是，它是先输入矩阵的宽再输入矩阵的长
    int a[n][m],dp[n][m];
    for (i=0; i<n; i++)
        for (j=0; j<m; j++) cin >> a[i][j];
```

```
    for (i=0; i<n; i++) dp[i][0] = a[i][0]; // 初始状态
    for (j=1; j<m; j++)
        for (i=0; i<n; i++)
            dp[i][j] = min(dp[i][j-1],dp[(i+n-1)%n][j-1])+a[i][j]; // 递推
    for (i=0; i<n; i++) ans = min(ans,dp[i][m-1]); // 取最小值
    cout << ans << endl;
}
```

Interesting thing:

没啦