

Aho-Corasick algorithm

高频题为此颜色

普通题为此色

算法题目一般从毫无感觉 -> 看懂别人的思路  
-> 培养自己的题感 -> 写出最优解法

```
class Solution {
    public int climbStairs(int n) {
        if (n == 2) return n;
        int firstStep = 1;
        int secondStep = 2;
        int total = 0;
        for (int i = 3; i <= n; ++i) {
            total = firstStep + secondStep;
            firstStep = secondStep;
            secondStep = total;
        }
        return total;
    }
}
```

爬楼梯

fibonacci  
当前一步 等于 前一步可能性 + 前前一步可能性

做题思路

做题思

路

对题目懵逼了

嵌套数组的遍历

Linked List

写下思路步骤  
1. 2. 3  
写所有可能的解法，不要有约束跟限制  
coding: first, check input  
valid

化繁为简，列出点最简单的情  
况  
找最近子问题的重复性，  
机器只能 for 或 recursion

从左至  
从右至  
左

熟能生巧，套路固  
定

Fibonacci

使用记忆化搜索缓存中间结果

Count the paths

paths 拆分成 求 A、B 到终点路径有多少步的子问题（分治，几种选择就几个子问题）

```
paths[start, end] =
    paths[A, end]
    +
    paths[D, end] + paths[C, end]
    +
    paths[E, end] + paths[F, end]
```

```
int countPaths(int n, int m, int row, int col) {
    if (isValidSquare(row, col, n, m)) return 0;
    if (isValidGrid(row, col, n, m)) return countPaths(row, col, n, m);
    return countPaths(row, col, n, m);
}
```

triangle

<https://leetcode-cn.com/problems/triangle/solution/di-gui-ji-yi-hua-sou-suo-zai-dao-dp-by-csm/>

[https://leetcode.com/problems/triangle/discuss/38735/Python-easy-to-understand-solutions-\(top-down-bottom-up\)](https://leetcode.com/problems/triangle/discuss/38735/Python-easy-to-understand-solutions-(top-down-bottom-up))

```
public int minimumTotal(List<Integer>> triangle) {
    // bottom to top
    int m = triangle.size();
    int[] dp = new int[m + 1];
    for (int i = m - 1; i >= 0; --i)
        for (int j = 0; j <= i; ++j)
            dp[j] = Math.min(dp[j], dp[j + 1]) + triangle.get(i).get(j);
    return dp[0];
}
```

注意 j 的 loop 条件是 j <= i, 因为 dp 的长度是 m + 1;

Coin Change

<https://leetcode-cn.com/problems/coin-change/solution/dong-tai-gui-hua-tao-lu-xiang-jie-by-weilaishu-ke/>

子问题，不断靠近 0 元，BFS(层数代表用了几个硬币，找层数用的最少的)

DP:  
a. subproblems  
b. DP array:  
f(n) = min(f(n - k), for k in [1, 2, 5]) + 1  
c. DP 方程:

House Robber

对当前房子可偷可不偷，技巧：开二维数组来存储偷不偷状态，对于复杂问题要开多个维度记录约束的条件  
a[i][0, 1]: 0: 偷, 1: 不偷

DP 公式  
dp[i][0] = Max(a[i - 1][0], a[i - 1][1])  
dp[i][1] = Max(a[i - 1][0], a[i - 1][1])

```
解法2
max = max(prevPax + x, currMax)
public int rob(int[] num) {
    int prevMax = 0;
    int currMax = 0;
    for (int x : num) {
        int temp = currMax;
        currMax = Math.max(prevMax + x, currMax);
        prevMax = temp;
    }
    return currMax;
}
```

动态规划

Unique Paths

方法1  
- 创建二维数组  
- 先将第一排跟第一列的 items fill 1  
- 使用递推公式:  
dp[i][j] = dp[i - 1][j] + dp[i][j - 1]  
- return dp[m - 1][n - 1]  
方法二 (节省了一个维度的空间，复杂度与方法一致)  
只保留前一行的值，然后更新当前行每个 col 的值  
- 初始化 new int[n], fill items to be 1  
- 从 i(row) = 1, j(column) = 1 开始遍历二维数组  
- curr[j] += curr[j - 1]; // += 这一步相当于把当前坐标的 col - 1 的值也加进去了，满足了 递推公式 dp[i][j] = dp[i - 1][j] + dp[i][j - 1]  
方法三 牛 1 - 1  
class Solution {  
 public int uniquePaths(int n, int m) {  
 if (m <= 0 || n <= 0) return 0;  
 long res = 1;  
 for (int i = 1; i <= m; i++) {  
 res = res \* (i + n - 1) / i;  
 }  
 return (int)res;  
 }  
}

状态转移方程 (DP 方程)

完整逻辑:  
if (a[i][j] == "空地") {  
 opt[i][j] = opt[i + 1, j] + opt[i][j + 1]  
} else {  
 opt[i][j] = 0  
}

Maximum Subarray

```
/**  
 * 状态转移方程: sum[i] = max(sum[i - 1] + a[i], a[i])  
 * 如果 sum > 0, 则说明 sum 对结果有增益效果, 则 sum 保留并加上当前遍历数字  
 * 如果 sum <= 0, 则说明 sum 对结果无增益效果, 需要舍弃, 则 sum 直接更新为当前遍历数字  
 * 每次比较 sum 跟 0 的大小, 将最大值赋给 sum, 最后返回 sum 值  
 * 时间复杂度: O(n)  
 */  
public int maxSubArray(int[] nums) {  
    int result = nums[0];  
    int sum = nums[0];  
    for (int i = 1; i < nums.length; ++i) {  
        // sum[i] = max(sum[i - 1] + a[i], a[i])  
        sum = Math.max(sum + nums[i], nums[i]);  
        result = Math.max(result, sum);  
    }  
    return result;  
}
```

Longest Common Subsequence

	0	1	2	3	4	5	6		
str2 a b a b c d e	0	-	a	b	a	b	c	d	e
0	-	0	0	0	0	0	0	0	0
1	a	0	0	1	1	1	1	1	1
2	c	0	0	1	1	2	2	2	2
3	e	0	0	1	1	2	2	3	3

```
class Solution {  
    public int longestCommonSubsequence(String text1, String text2) {  
        if (text1.length() == 0 || text2.length() == 0) return 0;  
        char[] chars1 = text1.toCharArray();  
        char[] chars2 = text2.toCharArray();  
        int[] dp = new int[chars1.length + 1][chars2.length + 1];  
        for (int i = 1; i <= chars1.length; ++i)  
            for (int j = 1; j <= chars2.length; ++j)  
                dp[i][j] = chars1[i - 1] == chars2[j - 1] ? dp[i - 1][j - 1] + 1 : Math.max(dp[i - 1][j], dp[i][j - 1]);  
        return dp[chars1.length][chars2.length];  
    }  
}
```

```
str1 = "ABACDC"  
str2 = "BACBAD"  
dp[5][1] = dp[5 - 1][1] + 1; // dp[4][1] = 1  
dp[5][2] = Math.max(dp[5 - 1][2], dp[5][1 - 1] + 1);  
dp[5][3] = dp[5 - 1][3] + 1; // dp[4][3] = 2  
dp[5][4] = dp[5 - 1][4] + 1; // dp[4][4] = 3  
dp[5][5] = dp[5 - 1][5] + 1; // dp[4][5] = 4  
dp[5][6] = dp[5 - 1][6] + 1; // dp[4][6] = 5  
dp[5][7] = dp[5 - 1][7] + 1; // dp[4][7] = 6  
dp[5][8] = dp[5 - 1][8] + 1; // dp[4][8] = 7  
dp[5][9] = dp[5 - 1][9] + 1; // dp[4][9] = 8  
dp[5][10] = dp[5 - 1][10] + 1; // dp[4][10] = 9  
dp[5][11] = dp[5 - 1][11] + 1; // dp[4][11] = 10  
dp[5][12] = dp[5 - 1][12] + 1; // dp[4][12] = 11  
dp[5][13] = dp[5 - 1][13] + 1; // dp[4][13] = 12  
dp[5][14] = dp[5 - 1][14] + 1; // dp[4][14] = 13  
dp[5][15] = dp[5 - 1][15] + 1; // dp[4][15] = 14  
dp[5][16] = dp[5 - 1][16] + 1; // dp[4][16] = 15  
dp[5][17] = dp[5 - 1][17] + 1; // dp[4][17] = 16  
dp[5][18] = dp[5 - 1][18] + 1; // dp[4][18] = 17  
dp[5][19] = dp[5 - 1][19] + 1; // dp[4][19] = 18  
dp[5][20] = dp[5 - 1][20] + 1; // dp[4][20] = 19  
dp[5][21] = dp[5 - 1][21] + 1; // dp[4][21] = 20  
dp[5][22] = dp[5 - 1][22] + 1; // dp[4][22] = 21  
dp[5][23] = dp[5 - 1][23] + 1; // dp[4][23] = 22  
dp[5][24] = dp[5 - 1][24] + 1; // dp[4][24] = 23  
dp[5][25] = dp[5 - 1][25] + 1; // dp[4][25] = 24  
dp[5][26] = dp[5 - 1][26] + 1; // dp[4][26] = 25  
dp[5][27] = dp[5 - 1][27] + 1; // dp[4][27] = 26  
dp[5][28] = dp[5 - 1][28] + 1; // dp[4][28] = 27  
dp[5][29] = dp[5 - 1][29] + 1; // dp[4][29] = 28  
dp[5][30] = dp[5 - 1][30] + 1; // dp[4][30] = 29  
dp[5][31] = dp[5 - 1][31] + 1; // dp[4][31] = 30  
dp[5][32] = dp[5 - 1][32] + 1; // dp[4][32] = 31  
dp[5][33] = dp[5 - 1][33] + 1; // dp[4][33] = 32  
dp[5][34] = dp[5 - 1][34] + 1; // dp[4][34] = 33  
dp[5][35] = dp[5 - 1][35] + 1; // dp[4][35] = 34  
dp[5][36] = dp[5 - 1][36] + 1; // dp[4][36] = 35  
dp[5][37] = dp[5 - 1][37] + 1; // dp[4][37] = 36  
dp[5][38] = dp[5 - 1][38] + 1; // dp[4][38] = 37  
dp[5][39] = dp[5 - 1][39] + 1; // dp[4][39] = 38  
dp[5][40] = dp[5 - 1][40] + 1; // dp[4][40] = 39  
dp[5][41] = dp[5 - 1][41] + 1; // dp[4][41] = 40  
dp[5][42] = dp[5 - 1][42] + 1; // dp[4][42] = 41  
dp[5][43] = dp[5 - 1][43] + 1; // dp[4][43] = 42  
dp[5][44] = dp[5 - 1][44] + 1; // dp[4][44] = 43  
dp[5][45] = dp[5 - 1][45] + 1; // dp[4][45] = 44  
dp[5][46] = dp[5 - 1][46] + 1; // dp[4][46] = 45  
dp[5][47] = dp[5 - 1][47] + 1; // dp[4][47] = 46  
dp[5][48] = dp[5 - 1][48] + 1; // dp[4][48] = 47  
dp[5][49] = dp[5 - 1][49] + 1; // dp[4][49] = 48  
dp[5][50] = dp[5 - 1][50] + 1; // dp[4][50] = 49  
dp[5][51] = dp[5 - 1][51] + 1; // dp[4][51] = 50  
dp[5][52] = dp[5 - 1][52] + 1; // dp[4][52] = 51  
dp[5][53] = dp[5 - 1][53] + 1; // dp[4][53] = 52  
dp[5][54] = dp[5 - 1][54] + 1; // dp[4][54] = 53  
dp[5][55] = dp[5 - 1][55] + 1; // dp[4][55] = 54  
dp[5][56] = dp[5 - 1][56] + 1; // dp[4][56] = 55  
dp[5][57] = dp[5 - 1][57] + 1; // dp[4][57] = 56  
dp[5][58] = dp[5 - 1][58] + 1; // dp[4][58] = 57  
dp[5][59] = dp[5 - 1][59] + 1; // dp[4][59] = 58  
dp[5][60] = dp[5 - 1][60] + 1; // dp[4][60] = 59  
dp[5][61] = dp[5 - 1][61] + 1; // dp[4][61] = 60  
dp[5][62] = dp[5 - 1][62] + 1; // dp[4][62] = 61  
dp[5][63] = dp[5 - 1][63] + 1; // dp[4][63] = 62  
dp[5][64] = dp[5 - 1][64] + 1; // dp[4][64] = 63  
dp[5][65] = dp[5 - 1][65] + 1; // dp[4][65] = 64  
dp[5][66] = dp[5 - 1][66] + 1; // dp[4][66] = 65  
dp[5][67] = dp[5 - 1][67] + 1; // dp[4][67] = 66  
dp[5][68] = dp[5 - 1][68] + 1; // dp[4][68] = 67  
dp[5][69] = dp[5 - 1][69] + 1; // dp[4][69] = 68  
dp[5][70] = dp[5 - 1][70] + 1; // dp[4][70] = 69  
dp[5][71] = dp[5 - 1][71] + 1; // dp[4][71] = 70  
dp[5][72] = dp[5 - 1][72] + 1; // dp[4][72] = 71  
dp[5][73] = dp[5 - 1][73] + 1; // dp[4][73] = 72  
dp[5][74] = dp[5 - 1][74] + 1; // dp[4][74] = 73  
dp[5][75] = dp[5 - 1][75] + 1; // dp[4][75] = 74  
dp[5][76] = dp[5 - 1][76] + 1; // dp[4][76] = 75  
dp[5][77] = dp[5 - 1][77] + 1; // dp[4][77] = 76  
dp[5][78] = dp[5 - 1][78] + 1; // dp[4][78] = 77  
dp[5][79] = dp[5 - 1][79] + 1; // dp[4][79] = 78  
dp[5][80] = dp[5 - 1][80] + 1; // dp[4][80] = 79  
dp[5][81] = dp[5 - 1][81] + 1; // dp[4][81] = 80  
dp[5][82] = dp[5 - 1][82] + 1; // dp[4][82] = 81  
dp[5][83] = dp[5 - 1][83] + 1; // dp[4][83] = 82  
dp[5][84] = dp[5 - 1][84] + 1; // dp[4][84] = 83  
dp[5][85] = dp[5 - 1][85] + 1; // dp[4][85] = 84  
dp[5][86] = dp[5 - 1][86] + 1; // dp[4][86] = 85  
dp[5][87] = dp[5 - 1][87] + 1; // dp[4][87] = 86  
dp[5][88] = dp[5 - 1][88] + 1; // dp[4][88] = 87  
dp[5][89] = dp[5 - 1][89] + 1; // dp[4][89] = 88  
dp[5][90] = dp[5 - 1][90] + 1; // dp[4][90] = 89  
dp[5][91] = dp[5 - 1][91] + 1; // dp[4][91] = 90  
dp[5][92] = dp[5 - 1][92] + 1; // dp[4][92] = 91  
dp[5][93] = dp[5 - 1][93] + 1; // dp[4][93] = 92  
dp[5][94] = dp[5 - 1][94] + 1; // dp[4][94] = 93  
dp[5][95] = dp[5 - 1][95] + 1; // dp[4][95] = 94  
dp[5][96] = dp[5 - 1][96] + 1; // dp[4][96] = 95  
dp[5][97] = dp[5 - 1][97] + 1; // dp[4][97] = 96  
dp[5][98] = dp[5 - 1][98] + 1; // dp[4][98] = 97  
dp[5][99] = dp[5 - 1][99] + 1; // dp[4][99] = 98  
dp[5][100] = dp[5 - 1][100] + 1; // dp[4][100] = 99  
dp[5][101] = dp[5 - 1][101] + 1; // dp[4][101] = 100  
dp[5][102] = dp[5 - 1][102] + 1; // dp[4][102] = 101  
dp[5][103] = dp[5 - 1][103] + 1; // dp[4][103] = 102  
dp[5][104] = dp[5 - 1][104] + 1; // dp[4][104] = 103  
dp[5][105] = dp[5 - 1][105] + 1; // dp[4][105] = 104  
dp[5][106] = dp[5 - 1][106] + 1; // dp[4][106] = 105  
dp[5][107] = dp[5 - 1][107] + 1; // dp[4][107] = 106  
dp[5][108] = dp[5 - 1][108] + 1; // dp[4][108] = 107  
dp[5][109] = dp[5 - 1][109] + 1; // dp[4][109] = 108  
dp[5][110] = dp[5 - 1][110] + 1; // dp[4][110] = 109  
dp[5][111] = dp[5 - 1][111] + 1; // dp[4][111] = 110  
dp[5][112] = dp[5 - 1][112] + 1; // dp[4][112] = 111  
dp[5][113] = dp[5 - 1][113] + 1; // dp[4][113] = 112  
dp[5][114] = dp[5 - 1][114] + 1; // dp[4][114] = 113  
dp[5][115] = dp[5 - 1][115] + 1; // dp[4][115] = 114  
dp[5][116] = dp[5 - 1][116] + 1; // dp[4][116] = 115  
dp[5][117] = dp[5 - 1][117] + 1; // dp[4][117] = 116  
dp[5][118] = dp[5 - 1][118] + 1; // dp[4][118] = 117  
dp[5][119] = dp[5 - 1][119] + 1; // dp[4][119] = 118  
dp[5][120] = dp[5 - 1][120] + 1; // dp[4][120] = 119  
dp[5][121] = dp[5 - 1][121] + 1; // dp[4][121] = 120  
dp[5][122] = dp[5 - 1][122] + 1; // dp[4][122] = 121  
dp[5][123] = dp[5 - 1][123] + 1; // dp[4][123] = 122  
dp[5][124] = dp[5 - 1][124] + 1; // dp[4][124] = 123  
dp[5][125] = dp[5 - 1][125] + 1; // dp[4][125] = 124  
dp[5][126] = dp[5 - 1][126] + 1; // dp[4][126] = 125  
dp[5][127] = dp[5 - 1][127] + 1; // dp[4][127] = 126  
dp[5][128] = dp[5 - 1][128] + 1; // dp[4][128] = 127  
dp[5][129] = dp[5 - 1][129] + 1; // dp[4][129] = 128  
dp[5][130] = dp[5 - 1][130] + 1; // dp[4][130] = 129  
dp[5][131] = dp[5 - 1][131] + 1; // dp[4][131] = 130  
dp[5][132] = dp[5 - 1][132] + 1; // dp[4][132] = 131  
dp[5][133] = dp[5 - 1][133] + 1; // dp[4][133] = 132  
dp[5][134] = dp[5 - 1][134] + 1; // dp[4][134] = 133  
dp[5][135] = dp[5 - 1][135] + 1; // dp[4][135] = 134  
dp[5][136] = dp[5 - 1][136] + 1; // dp[4][136] = 135  
dp[5][137] = dp[5 - 1][137] + 1; // dp[4][137] = 136  
dp[5][138] = dp[5 - 1][138] + 1; // dp[4][138] = 137  
dp[5][139] = dp[5 - 1][139] + 1; // dp[4][139] = 138  
dp[5][140] = dp[5 - 1][140] + 1; // dp[4][140] = 139  
dp[5][141] = dp[5 - 1][141] + 1; // dp[4][141] = 140  
dp[5][142] = dp[5 - 1][142] + 1; // dp[4][142] = 141  
dp[5][143] = dp[5 - 1][143] + 1; // dp[4][143] = 142  
dp[5][144] = dp[5 - 1][144] + 1; // dp[4][144] = 143  
dp[5][145] = dp[5 - 1][145] + 1; // dp[4][145] = 144  
dp[5][146] = dp[5 - 1][146] + 1; // dp[4][146] = 145  
dp[5][147] = dp[5 - 1][147] + 1; // dp[4][147] = 146  
dp[5][148] = dp[5 - 1][148] + 1; // dp[4][148] = 147  
dp[5][149] = dp[5 - 1][149] + 1; // dp[4][149] = 148  
dp[5][150] = dp[5 - 1][150] + 1; // dp[4][150] = 149  
dp[5][151] = dp[5 - 1][151] + 1; // dp[4][151] = 150  
dp[5][152] = dp[5 - 1][152] + 1; // dp[4][152] = 151  
dp[5][153] = dp[5 - 1][153] + 1; // dp[4][153] = 152  
dp[5][154] = dp[5 - 1][154] + 1; // dp[4][154] = 153  
dp[5][155] = dp[5 - 1][155] + 1; // dp[4][155] = 154  
dp[5][156] = dp[5 - 1][156] + 1; // dp[4][156] = 155  
dp[5][157] = dp[5 - 1][157] + 1; // dp[4][157] = 156  
dp[5][158] = dp[5 - 1][158] + 1; // dp[4][158] = 157  
dp[5][159] = dp[5 - 1][159] + 1; // dp[4][159] = 158  
dp[5][160] = dp[5 - 1][160] + 1; // dp[4][160] = 159  
dp[5][161] = dp[5 - 1][161] + 1; // dp[4][161] = 160  
dp[5][162] = dp[5 - 1][162] + 1; // dp[4][162] = 161  
dp[5][163] = dp[5 - 1][163] + 1; // dp[4][163] = 162  
dp[5][164] = dp[5 - 1][164] + 1; // dp[4][164] = 163  
dp[5][165] = dp[5 - 1][165] + 1; // dp[4][165] = 164  
dp[5][166] = dp[5 - 1][166] + 1; // dp[4][166] = 165  
dp[5][167] = dp[5 - 1][167] + 1; // dp[4][167] = 166  
dp[5][168] = dp[5 - 1][168] + 1; // dp[4][168] = 167  
dp[5][169] = dp[5 - 1][169] + 1; // dp[4][169] = 168  
dp[5][170] = dp[5 - 1][170] + 1; // dp[4][170] = 169  
dp[5][171] = dp[5 - 1][171] + 1; // dp[4][171] = 170  
dp[5][172] = dp[5 - 1][172] + 1; // dp[4][172] = 171  
dp[5][173] = dp[5 - 1][173] + 1; // dp[4][173] = 172  
dp[5][174] = dp[5 - 1][174] + 1; // dp[4][174] = 173  
dp[5][175] = dp[5 - 1][175] + 1; // dp[4][175] = 174  
dp[5][176] = dp[5 - 1][176] + 1; // dp[4][176] = 175  
dp[5][177] = dp[5 - 1][177] + 1; // dp[4][177] = 176  
dp[5][178] = dp[5 - 1][178] + 1; // dp[4][178] = 177  
dp[5][179] = dp[5 - 1][179] + 1; // dp[4][179] = 178  
dp[5][180] = dp[5 - 1][180] + 1; // dp[4][180] = 179  
dp[5][181] = dp[5 - 1][181] + 1; // dp[4][181] = 180  
dp[5][182] = dp[5 - 1][182] + 1; // dp[4][182] = 181  
dp[5][183] = dp[5 - 1][183] + 1; // dp[4][183] = 182  
dp[5][184] = dp[5 - 1][184] + 1; // dp[4][184] = 183  
dp[5][185] = dp[5 - 1][185] + 1; // dp[4][185] = 184  
dp[5][186] = dp[5 - 1][186] + 1; // dp[4][186] = 185  
dp[5][187] = dp[5 - 1][187] + 1; // dp[4][187] = 186  
dp[5][188] = dp[5 - 1][188] + 1; // dp[4][188] = 187  
dp[5][189] = dp[5 - 1][189] + 1; // dp[4][189] = 188  
dp[5][190] = dp[5 - 1][190] + 1; // dp[4][190] = 189  
dp[5][191] = dp[5 - 1][191] + 1; // dp[4][191] = 190  
dp[5][192] = dp[5 - 1][192] + 1; // dp[4][192] = 191  
dp[5][193] = dp[5 - 1][193] + 1; // dp[4][193] = 192  
dp[5][194] = dp[5 - 1][194] + 1; // dp[4][194] = 193  
dp[5][195] = dp[5 - 1][195] + 1; // dp[4][195] = 194  
dp[5][196] = dp[5 - 1][196] + 1; // dp[4][196] = 195  
dp[5][197] = dp[5 - 1][197] + 1; // dp[4][197] = 196  
dp[5][198] = dp[5 - 1][198] + 1; // dp[4][198] = 197  
dp[5][199] = dp[5 - 1][199] + 1; // dp[4][199] = 198  
dp[5][200] = dp[5 - 1][200] + 1; // dp[4][200] = 199  
dp[5][201] = dp[5 - 1][201] + 1; // dp[4][201] = 200  
dp[5][202] = dp[5 - 1][202] + 1; // dp[4][202] = 201  
dp[5][203] = dp[5 - 1][203] + 1; // dp[4][203] = 202  
dp[5][204] = dp[5 - 1][204] + 1; // dp[4][204] = 203  
dp[5][205] = dp[5 - 1][205] + 1; // dp[4][205] = 204  
dp[5][206] = dp[5 - 1][206] + 1; // dp[4][206] = 205  
dp[5][207] = dp[5 - 1][207] + 1; // dp[4][207] = 206  
dp[5][208] = dp[5 - 1][208] + 1; // dp[4][208] = 207  
dp[5][209] = dp[5 - 1][209] + 1; // dp[4][209] = 208  
dp[5][210] = dp[5 - 1][210] + 1; // dp[4][210] = 209  
dp[5][211] = dp[5 - 1][211] + 1; // dp[4][211] = 210  
dp[5][212] = dp[5 - 1][212] + 1; // dp[4][212] = 211  
dp[5][213] = dp[5 - 1][213] + 1; // dp[4][213] = 212  
dp[5][214] = dp[5 - 1][214] + 1; // dp[4][214] = 213  
dp[5][215] = dp[5 - 1][215] + 1; // dp[4][215] = 214  
dp[5][216] = dp[5 - 1][216] + 1; // dp[4][216] = 215  
dp[5][217] = dp[5 - 1][217] + 1; // dp[4][217] = 216  
dp[5][218] = dp[5 - 1][218] + 1; // dp[4][218] = 217  
dp[5][219] = dp[5 - 1][219] + 1; // dp[4][219] = 218  
dp[5][220] = dp[5 - 1][220] + 1; // dp[4][220] = 219  
dp[5][221] = dp[5 - 1][221] + 1; // dp[4][221] = 220  
dp[5][222] = dp[5 - 1][222] + 1; // dp[4][222] = 221  
dp[5][223] = dp[5 - 1][223] + 1; // dp[4][223] = 222  
dp[5][224] = dp[5 - 1][224]
```