Leetcode\_174\_DungeonGame\_地牢游戏\_Hard

# Leetcode\_174\_DungeonGame\_地牢游戏\_Hard

## 题目介绍

/\*\*

\* 难度：Hard

\* https://leetcode.com/problems/dungeon-game/description/

\*

\* The demons had captured the princess (P) and imprisoned her

\* in the bottom-right corner of a dungeon.

\* The dungeon consists of M x N rooms laid out in a 2D grid.

\* Our valiant knight (K) was initially positioned in the top-left room

\* and must fight his way through the dungeon to rescue the princess.

\* The knight has an initial health point represented by a positive integer.

\* If at any point his health point drops to 0 or below, he dies immediately.

\*

\* Some of the rooms are guarded by demons, so the knight loses health (negative integers)

\* upon entering these rooms; other rooms are either empty (0's) or contain magic orbs

\* that increase the knight's health (positive integers).

\*

\* In order to reach the princess as quickly as possible, the knight decides to move

\* only rightward or downward in each step.

\* Write a function to determine the knight's minimum initial health

\* so that he is able to rescue the princess.

\* For example, given the dungeon below, the initial health of the knight must be

\* at least 7 if he follows the optimal path RIGHT-> RIGHT -> DOWN -> DOWN.

\*-2 (K) -3 3

\* -5 -10 1

\* 10 30 -5 (P)

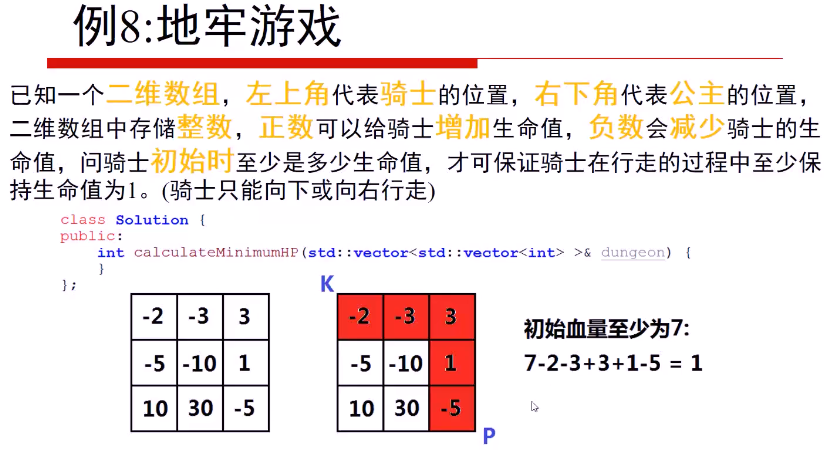
\* Note:

\* The knight's health has no upper bound.

\* Any room can contain threats or power-ups,

\* even the first room the knight enters and the bottom-right room

\* where the princess is imprisoned.



## 思路分析

动态规划思想：

\* 思路分析：动态规划思想。

\* 从右下向左上推进。

\* dp[i][j]表示到达当前点之前至少需要的血量值；

\* 因为血量至少是1才表示存活，因此到达该点至少需要血量值1；

\* 又当前消耗可能为负值也可能为正值，但是无论如何，加上该方格的血量，

\* 最后只是要大于等于1；即终点方格的dp[i][j]=max{1,1-dungeon[i][j]};

\* 如终点方格为-5，这时到达该方格前至少需要血量为1-(-5)=6;

\* 终点方格为5，这时到达该方格前至少需要血量max{1,1-5}=1;(因为小于1表示已死亡)

\* 状态转移方程：

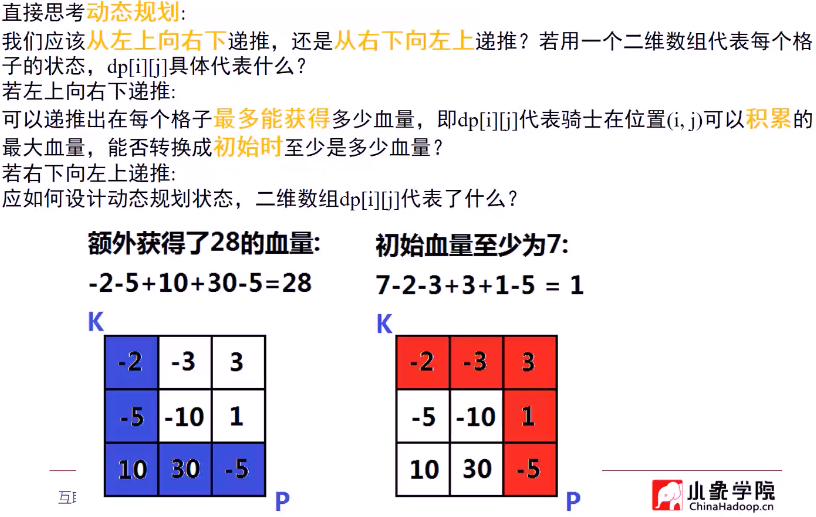
\* dp[i][j] = max{1,min(dp[i][j+1],dp[i+1][j])}

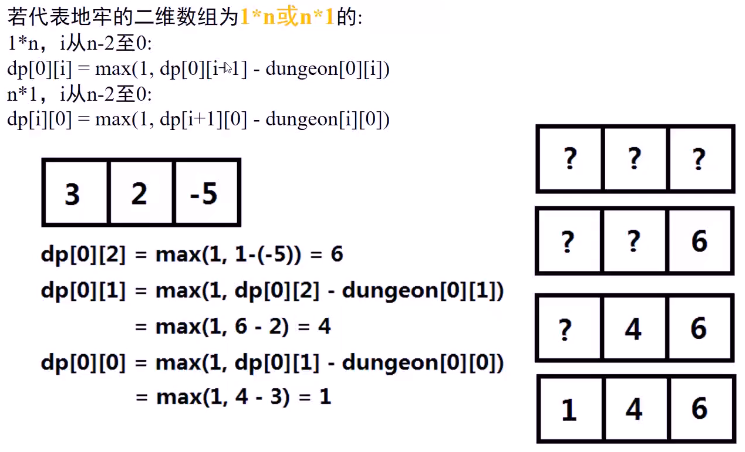
\* 初始化：终点dp[rows-1][columns-1]、最后一行和最后一列

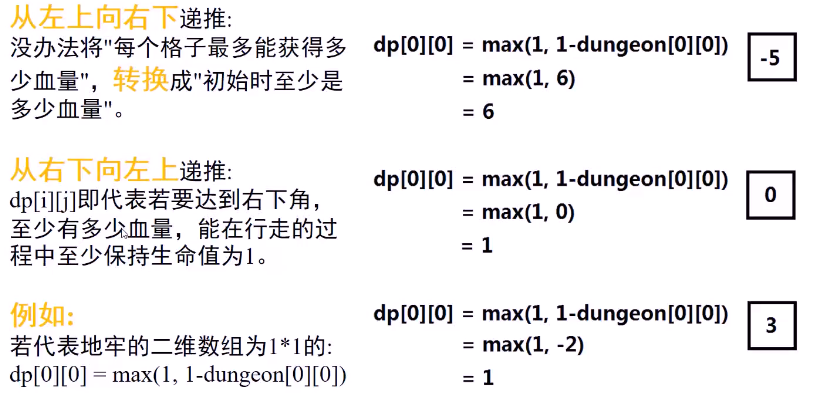
\* dp[rows-1][columns-1] = max{1,1-dungeon[i][j]};

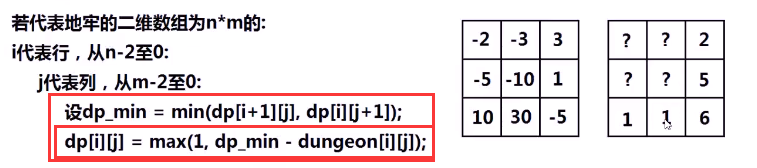
\* dp[rows-1][j] = max{1,dp[rows-1][j+1]-dungeon[i][j]};

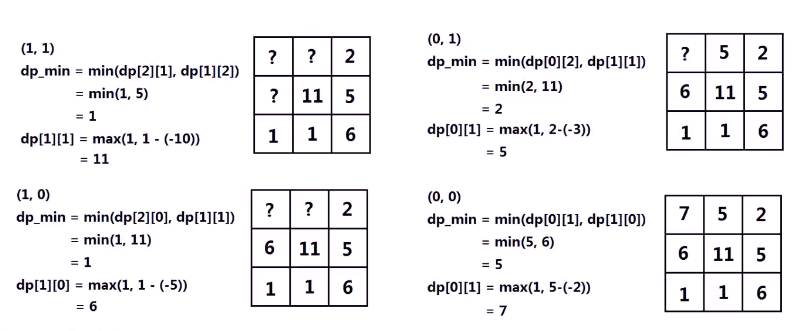
\* dp[i][columns-1] = max{1,dp[i+1][columns-1]-dungeon[i][j]};











## Java代码

public int calculateMinimumHP(int[][] dungeon) {

if(dungeon == null||dungeon.length == 0) return 0;

int rows = dungeon.length,columns = dungeon[0].length;

//创建dp

int[][] dp = new int[rows][columns];

//初始化

**dp[rows-1][columns-1] = Math.max(1,1-dungeon[rows-1][columns-1]);**

for(int i = rows-2;i >= 0;i--)

**dp[i][columns-1] = Math.max(1,dp[i+1][columns-1]-dungeon[i][columns-1]);**

for(int j = columns-2;j >= 0;j--)

**dp[rows-1][j] = Math.max(1,dp[rows-1][j+1]-dungeon[rows-1][j]);**

//开始执行

for(int i = rows-2;i >= 0;i--){

for(int j = columns-2;j >= 0;j--){

**dp[i][j] = Math.max(1,Math.min(dp[i+1][j],dp[i][j+1])-dungeon[i][j]);**

}

}

return dp[0][0];

}

