Leetcode\_120\_Triangle\_三角形\_Medium

# Leetcode\_120\_Triangle\_三角形\_Medium

## 问题介绍

\* 难度：Medium

\* <https://leetcode.com/problems/triangle/description/>

\* 题目介绍：

\* Given a triangle, find the minimum path sum from top to bottom.

\* Each step you may move to adjacent numbers on the row below.

\* For example, given the following triangle

\* [

\* [2],

\* [3,4],

\* [6,5,7],

\* [4,1,8,3]

\* ]

\* The minimum path sum from top to bottom is 11 (i.e., 2 + 3 + 5 + 1 = 11).

\* Note: Bonus point if you are able to do this using only O(n) extra space,

\* where n is the total number of rows in the triangle.



## 思路分析

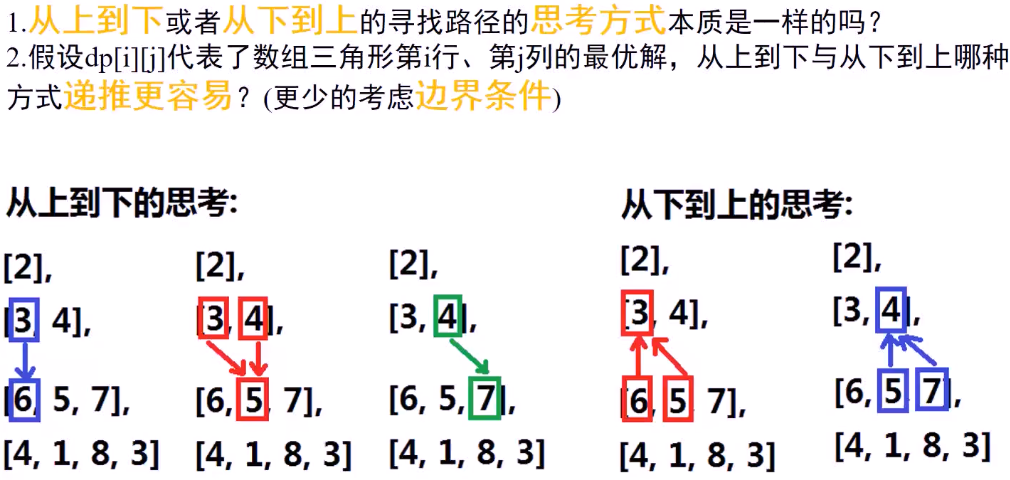
\* 思路分析：

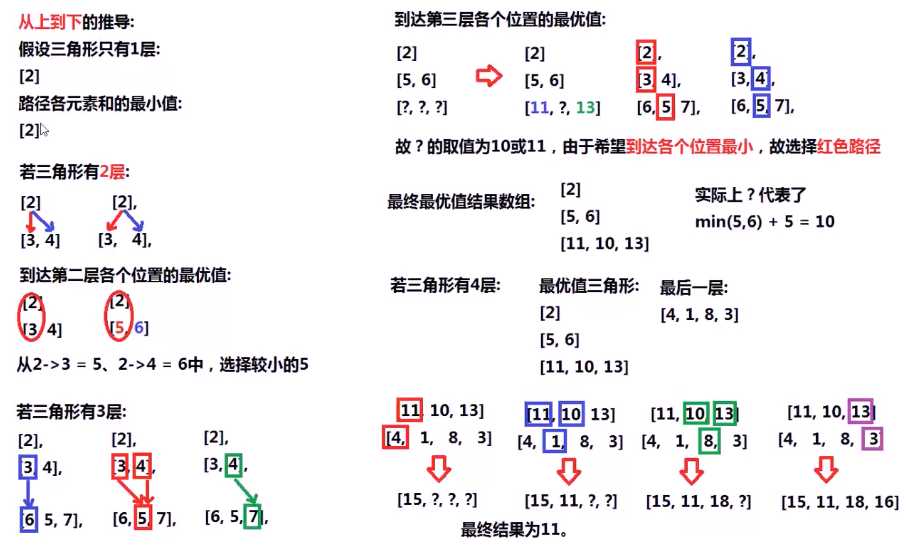
\* 动态规划状态方程：dp[i][j]表示从下到上到达(i,j)的最小路径之和；

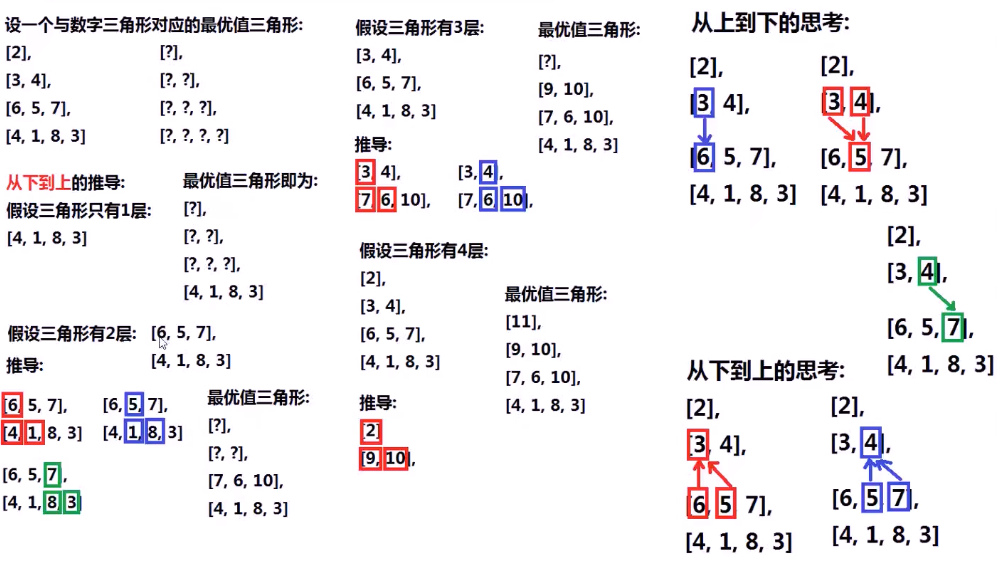
\* dp[i][j] = min{dp[i+1][j],dp[i+1][j+1]} + triangle[i][j];

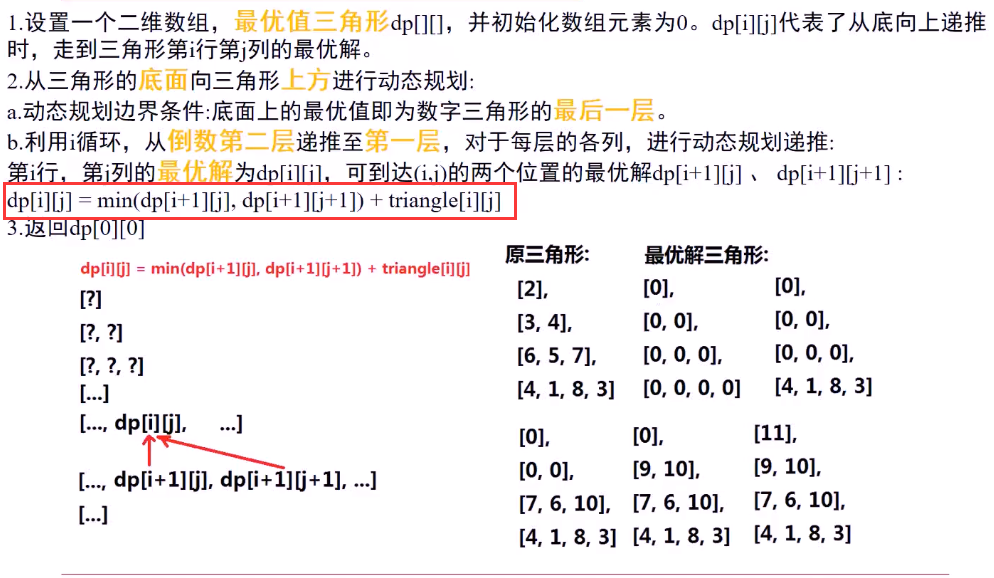
\* 边界条件：初始最后一行dp[n][j] = triangle[n][j];

\* 最终返回dp[0][0]即可。









## Java代码

//利用int[][]二维数组实现dp

public int **minimumTotal**(List<List<Integer>> triangle) {

if(triangle == null||triangle.size() == 0) return 0;

int rows = triangle.size();

//建立dp数组，dp存放从下到上到达该点的最小路径之和

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

for(int i = 0;i<rows;i++)

dp[i] = new int[i+1];

//初始化最后一行

for(int column = 0;column < rows;column++){

dp[rows-1][column] = triangle.get(rows-1).get(column);

}

//遍历计算dp[row][column]

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

for(int column = 0;column <= row;column++){

dp[row][column] = Math.min(dp[row+1][column],dp[row+1][column+1])

+ triangle.get(row).get(column);

}

}

**return dp[0][0];**

}

//利用List<List<Integer>>做dp实现

public int **minimumTotal2**(List<List<Integer>> triangle) {

if(triangle == null||triangle.size() == 0) return 0;

int rows = triangle.size();

//dp存放从下到上到达该点的最小路径之和

ArrayList<List<Integer>> dp = new ArrayList<List<Integer>>(rows);

List<Integer> rowList = null;

for(int i = 0;i < rows;i++) {

rowList = new ArrayList<Integer>(i + 1);

for(int j = 0;j <= i;j++)

rowList.add(j,0);

dp.add(rowList);

}

//初始化最后一行

Collections.copy(rowList,triangle.get(rows-1));

dp.set(rows-1,rowList);

//遍历计算dp[row][column]

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

for(int column = 0;column <= row;column++){

dp.get(row).set(column,Math.min(dp.get(row+1).get(column),dp.get(row+1).get(column+1))+triangle.get(row).get(column));

}

}

**return dp.get(0).get(0);**

}

