1. 题目
2. 题干

给定一个三角形，找出自顶向下的最小路径和。每一步只能移动到下一行中相邻的结点上。

相邻的结点 在这里指的是 下标 与 上一层结点下标 相同或者等于 上一层结点下标 + 1 的两个结点。

1. 示例

例如，给定三角形：

[

[2],

[3,4],

[6,5,7],

[4,1,8,3]

]

自顶向下的最小路径和为 11（即，2 + 3 + 5 + 1 = 11）。

1. 题解
2. 思路

这种题目，第一想法是通过递归+回溯求解。但是按照递归来做之后，发现有的用例是超时的，那就用动态规划做吧。题目的思路是除了第一层的节点外，下面每层的节点都是由上一层的相邻节点之间进行比较来选择路径和较小的一个进行转移的。因此，每一层节点的状态都是由上一层节点的状态转移过来的，因此设置一个数组f[i][j]表示从三角形顶部（0,0）走到位置（i,j）的最小路径和。从上一层的相邻节点到下一层节点的状态转移方程为：c[i][j]为(i，j)位置的元素值。

f[i][j] = min(f[i-1][j-1], f[i-1][j]) + c[i][j]

注意，在j=0时，即在每行的最左边时，只能由上一层的最左边节点转移，即f[i-1][0]转移下来；在j=i时，即每行的最右边时，只能由上一层的最右边节点转移，即f[i-1][i-1]转移下来。

边界条件是f[0][0] = c[0][0]。在三角形的顶部，最小路径和就等于对应位置的元素值。这样，可以一层一层地填补f[][]数组，最后取出f[n-1]数组中的最小值即可。

这种状态转移的题目就要用动态规划来解决！！！！！！！！！！！！

1. 代码实现

动态规划：

class Solution {

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

        int n = triangle.size();

        int[][] f = new int[n][n];

        f[0][0] = triangle.get(0).get(0);

        for(int i = 1; i < n; ++i){

            f[i][0] = f[i-1][0] + triangle.get(i).get(0);

            for(int j = 1; j < i; ++j){

                f[i][j] = Math.min(f[i-1][j-1],f[i-1][j]) + triangle.get(i).get(j);

            }

            f[i][i] = f[i-1][i-1] + triangle.get(i).get(i);

        }

        int minNum  = f[n-1][0];

        for(int i = 0; i < n; ++i){

            minNum = Math.min(minNum, f[n-1][i]);

        }

        return minNum;

    }

}