0120. 三角形最小路径和

ITCharge 大约 3 分钟

- 标签：数组、动态规划
- 难度：中等

题目链接

- 0120. 三角形最小路径和 - 力扣

题目大意

描述：给定一个代表三角形的二维数组 $triangle$，$triangle$ 共有 n 行，其中第 i 行（从 0 开始编号）包含了 $i + 1$ 个数。

我们每一步只能从当前位置移动到下一行中相邻的节点上。也就是说，如果正位于第 i 行第 j 列的节点，那么下一步可以移动到第 $i + 1$ 行第 j 列的位置上，或者第 $i + 1$ 行，第 $j + 1$ 列的位置上。

要求：找出自顶向下的最小路径和。

说明：

- $1 \le triangle.length \le 200$。
- $triangle[0].length == 1$。
- $triangle[i].length == triangle[i - 1].length + 1$。
- $-10^4 \le triangle[i][j] \le 10^4$。

示例：

- 示例 1：

```py
输入：triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
输出：11
解释：如下面简图所示：
   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）。

- 示例 2：

```py
输入：triangle = [[-10]]
输出：-10
```

解题思路

思路 1：动态规划

1. 划分阶段

按照行数进行阶段划分。

2. 定义状态

定义状态 $dp[i][j]$ 表示为：从顶部走到第 i 行（从 0 开始编号）、第 j 列的位置时的最小路径和。

3. 状态转移方程

由于每一步只能从当前位置移动到下一行中相邻的节点上，想要移动到第 i 行、第 j 列的位置，那么上一步只能在第 $i-1$ 行、第 $j-1$ 列的位置上，或者在第 $i-1$ 行、第 j 列的位置上。则状态转移方程为：

$dp[i][j] = min(dp[i-1][j-1], dp[i-1][j]) + triangle[i][j]$。其中 $triangle[i][j]$ 表示第 i 行、第 j 列位置上的元素值。

4. 初始条件

在第 0 行、第 j 列时，最小路径和为 $triangle[0][0]$，即 $dp[0][0] = triangle[0][0]$。

5. 最终结果

根据我们之前定义的状态，$dp[i][j]$ 表示为：从顶部走到第 i 行（从 0 开始编号）、第 j 列的位置时的最小路径和。为了计算出最小路径和，则需要再遍历一遍 $dp[size-1]$ 行的每一列，求出最小值即为最终结果。

思路 1：动态规划代码

```py
class Solution:
    def minimumTotal(self, triangle: List[List[int]]) -> int:
        size = len(triangle)
        dp = [[0 for _ in range(size)] for _ in range(size)]
        dp[0][0] = triangle[0][0]

        for i in range(1, size):
            dp[i][0] = dp[i - 1][0] + triangle[i][0]
            for j in range(1, i):
                dp[i][j] = min(dp[i - 1][j - 1], dp[i - 1][j]) + triangle[i][j]
            dp[i][i] = dp[i - 1][i - 1] + triangle[i][i]

        return min(dp[size - 1])
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$。两重循环遍历的时间复杂度是 $O(n^2)$，最后求最小值的时间复杂度是 $O(n)$，所以总体时间复杂度为 $O(n^2)$。
- **空间复杂度**：$O(n^2)$。用到了二维数组保存状态，所以总体空间复杂度为 $O(n^2)$。