120. Triangle

Given a `triangle` array, return *the minimum path sum from top to bottom*.

For each step, you may move to an adjacent number of the row below. More formally, if you are on index `i` on the current row, you may move to either index `i` or index `i + 1` on the next row.

Example 1:

```
Input: triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
Output: 11
Explanation: The triangle looks like:
   2
  3 4
 6 5 7
4 1 8 3
The minimum path sum from top to bottom is 2 + 3 + 5 + 1 = 11 (underlined
above).
```

Example 2:

```
Input: triangle = [[-10]]
Output: -10
```

Constraints:

- `1 <= triangle.length <= 200`
- `triangle[0].length == 1`
- `triangle[i].length == triangle[i - 1].length + 1`
- `-104 <= triangle[i][j] <= 104`

```python
class Solution:
    def minimumTotal(self, triangle: List[List[int]]) -> int:
        dp = []
        for row in triangle:
            temp = len(row)
            temp2 = []
            for i in range(temp):
                temp2.append(0)
            dp.append(temp2)
```

```
        for i in range(len(dp)-1,-1,-1):
            for j in range(len(dp[i])-1,-1,-1):
                if i==len(dp)-1:
                    dp[i][j] = triangle[i][j]
                else:
                    target1 = dp[i+1][j]
                    target2 = dp[i+1][j+1]
                    dp[i][j] = min(triangle[i][j]+target1,triangle[i]
[j]+target2)
        return dp[0][0]
```