931. Minimum Falling Path Sum

Given an `n x n` array of integers `matrix`, return *the minimum sum of any falling path through* `matrix`.

A falling path starts at any element in the first row and chooses the element in the next row that is either directly below or diagonally left/right. Specifically, the next element from position `(row, col)` will be `(row + 1, col - 1)`, `(row + 1, col)`, or `(row + 1, col + 1)`.

Example 1:

```
Input: matrix = [[2,1,3],[6,5,4],[7,8,9]]
Output: 13
Explanation: There are two falling paths with a minimum sum underlined below:
[[2,1,3],      [[2,1,3],
 [6,5,4],       [6,5,4],
 [7,8,9]]       [7,8,9]]
```

Example 2:

```
Input: matrix = [[-19,57],[-40,-5]]
Output: -59
Explanation: The falling path with a minimum sum is underlined below:
[[-19,57],
 [-40,-5]]
```

Example 3:

```
Input: matrix = [[-48]]
Output: -48
```

Constraints:

- `n == matrix.length`
- `n == matrix[i].length`
- `1 <= n <= 100`
- `-100 <= matrix[i][j] <= 100`

-
  ```
  def minFallingPathSum(self, matrix: List[List[int]]) -> int:
          n = len(matrix)
          dp = [[0]*n for i in range(n)]
          for i in range(n):
              for j in range(n):
                  if i==0:
                      dp[i][j]=matrix[i][j]
                  elif j==0 and i>0:
                      target1 = dp[i-1][j]
                      target2 = dp[i-1][j+1]
                      dp[i][j] = min(target1,target2)+matrix[i][j]
                  elif j==n-1 and i>0:
                      target1 = dp[i-1][j]
                      target2 = dp[i-1][j-1]
                      dp[i][j] = min(target1,target2)+matrix[i][j]
                  else:
                      target1 = dp[i-1][j-1]
                      target2 = dp[i-1][j]
                      target3 = dp[i-1][j+1]
                      dp[i][j] = min(target1,target2,target3)+matrix[i]
  [j]
          return min(dp[n-1])
  ```