给你一个 n x n 的 方形 整数数组 matrix ，请你找出并返回通过 matrix 的下降路径 的 最小和 。

下降路径 可以从第一行中的任何元素开始，并从每一行中选择一个元素。在下一行选择的元素和当前行所选元素最多相隔一列（即位于正下方或者沿对角线向左或者向右的第一个元素）。具体来说，位置 (row, col) 的下一个元素应当是 (row + 1, col - 1)、(row + 1, col) 或者 (row + 1, col + 1) 。

```
class Solution {
public:
  int minFallingPathSum(std::vector<std::vector<int>>& matrix) {
    int row = matrix.size(), col = 0;
    if (row != 0) {
      col = matrix[0].size();
    } else {
      return 0;
    }

    std::vector<std::vector<int>> memo =
        std::vector<std::vector<int>>(row + 1, std::vector<int>(col, 0));
    // base case
    for (int i = 0; i < col; i++) {
      memo[0][i] = matrix[0][i];
    }

    // dp 数组计算
    for (int i = 1; i < row; i++) {
      for (int j = 0; j < col; j++) {
        int min    = getMin(i, j, row, col, memo);
        memo[i][j] = min + matrix[i][j];
      }
    }

    // 查找最小值，在最后一行
    int res = INT_MAX;
    for (int i = 0; i < col; i++) {
      res = res < memo[row - 1][i] ? res : memo[row - 1][i];
    }

    return res;
  }

private:
  int getMin(int                                  i,
             int                                  j,
             int                                  row,
             int                                  col,
             std::vector<std::vector<int>>& memo) {
    int a = INT_MAX, b = INT_MAX, c = INT_MAX;
```

```
      if (isInArea(i - 1, j - 1, row, col)) {
        a = memo[i - 1][j - 1];
      }

      if (isInArea(i - 1, j, row, col)) {
        b = memo[i - 1][j];
      }

      if (isInArea(i - 1, j + 1, row, col)) {
        c = memo[i - 1][j + 1];
      }

      return std::min(std::min(a, b), c);
    }

    bool isInArea(int i, int j, int row, int col) {
      return i >= 0 && i < row && j >= 0 && j < col;
    }
  };
```