```
class Solution {
    public int minFallingPathSum(int[][] arr) {
        int r = arr.length, c = arr[0].length, res = Integer.MAX_VALUE;

        for (int i = 1; i < r; i++) {
            for (int j = 0; j < c; j++) {
                int left = (j == 0) ? Integer.MAX_VALUE : arr[i-1][j-1];
                int mid = arr[i-1][j];
                int right = (j == c - 1) ? Integer.MAX_VALUE : arr[i-1][j+1];
                arr[i][j] += Math.min(Math.min(left, mid), right);
            }
        }

        for (int j = 0; j < c; j++) {
            res = Math.min(res, arr[r-1][j]);
        }

        return res;
    }
}
```

Given a **square** array of integers `A`, we want the **minimum**sum of a *falling path* through `A`.

A falling path starts at any element in the first row, and chooses one element from each row. The next row's choice must be in a column that is different from the previous row's column by at most one.