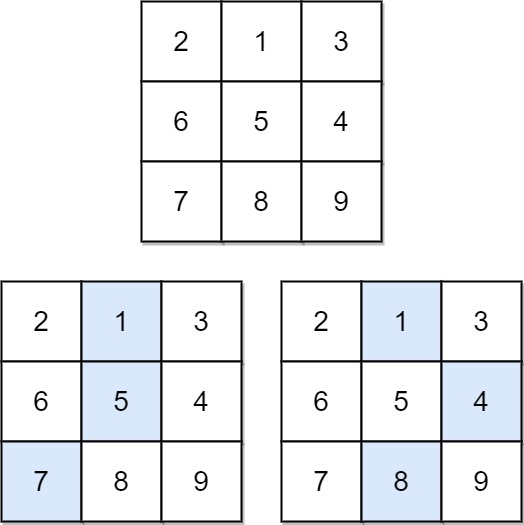
Given an n x n array of integers matrix, return *the****minimum sum****of any****falling path****through* matrix.

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:**

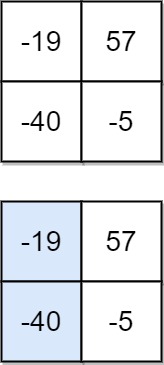


**Input:** matrix = [[2,1,3],[6,5,4],[7,8,9]]

**Output:** 13

**Explanation:** There are two falling paths with a minimum sum as shown.

**Example 2:**



**Input:** matrix = [[-19,57],[-40,-5]]

**Output:** -59

**Explanation:** The falling path with a minimum sum is shown.

**Solution:**

class Solution {

public int minFallingPathSum(int[][] A) {

int dp[][] = new int[A.length][A.length];

int min = Integer.MAX\_VALUE;

for(int i = 0; i < A.length; i++)

{

dp[0][i] = A[0][i];

}

for(int i = 1; i < A.length; i++)

{

for(int j = 0; j < A.length; j++)

{

if(j == 0)

{

dp[i][j] = A[i][j] + Math.min(dp[i-1][j], dp[i-1][j+1]);

}

else if(j == A.length - 1)

{

dp[i][j] = A[i][j] + Math.min(dp[i-1][j-1], dp[i-1][j]);

}

else

{

dp[i][j] = A[i][j] + Math.min(Math.min(dp[i-1][j-1], dp[i-1][j]), dp[i-1][j+1]);

}

}

}

for(int i = 0; i < A.length; i++)

{

if(dp[A.length-1][i] < min) min = dp[A.length-1][i];

}

return min;

}

}