931. Minimum Falling Path Sum

Medium,
Dynamic Programming.

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.

Example 1:

```
Input: [[1,2,3],[4,5,6],[7,8,9]]
Output: 12
Explanation:
The possible falling paths are:
[1,4,7], [1,4,8], [1,5,7], [1,5,8], [1,5,9]
[2,4,7], [2,4,8], [2,5,7], [2,5,8], [2,5,9], [2,6,8], [2,6,9]
[3,5,7], [3,5,8], [3,5,9], [3,6,8], [3,6,9]
The falling path with the smallest sum is [1,4,7], so the answer is 12.
```

Note:

```
1 <= A.length == A[0].length <= 100
-100 <= A[i][j] <= 100
```

解法

初始化跟A[][]一样的数组dp[][],当i=0时跟A是一模一样的。
第二行开始，会根据列的不同有三种情况，
第一种是j=0时，只能选去j=j和j=j+1
第二种是j=lenCol, 只能选j=j-1,j=j
第三种是其他情况， j=j, j=j-1, j=j+1

follow up,

减少空间复杂度的一种解法是，不设置完整的dp，可以设置奇偶数组轮流遍历。

java

```
class Solution {
    public int minFallingPathSum(int[][] A) {
        int lenRow = A.length;
        int lenCol = A[0].length;
        int dp[][] = new int[lenRow][lenCol];
        for(int j=0; j<lenCol; j++){
            dp[0][j] = A[0][j];
        }
        for(int i=1; i<lenRow; i++){
            for(int j=0; j<lenCol; j++){
                if(j==0){
                    dp[i][j]=Math.min(dp[i-1][j]+A[i][j],dp[i-1][j+1]+A[i][j]);
                }else if(j==(lenCol-1)){
                    dp[i][j]=Math.min(dp[i-1][j]+A[i][j],dp[i-1][j-1]+A[i][j]);
                }else{
                    dp[i][j]=Math.min(dp[i-1][j]+A[i][j],Math.min(dp[i-1][j-1]+A[i][j],dp[i-1][j+1]+A[i]
[j]));
                }
            }
        }
        int res = Integer.MAX_VALUE;
        for(int j=0; j<lenCol; j++){
            res = Math.min(dp[lenRow-1][j],res);
        }
        return res;
    }
}
```