题目：

A sequence of numbers is called a **wiggle sequence** if the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with fewer than two elements is trivially a wiggle sequence.

For example, `[1,7,4,9,2,5]` is a wiggle sequence because the differences (6,-3,5,-7,3) are alternately positive and negative. In contrast,`[1,4,7,2,5]` and `[1,7,4,5,5]` are not wiggle sequences, the first because its first two differences are positive and the second because its last difference is zero.

Given a sequence of integers, return the length of the longest subsequence that is a wiggle sequence. A subsequence is obtained by deleting some number of elements (eventually, also zero) from the original sequence, leaving the remaining elements in their original order.

Examples:

```
Input: [1,7,4,9,2,5]
Output: 6
The entire sequence is a wiggle sequence.


Input: [1,17,5,10,13,15,10,5,16,8]
Output: 7
There are several subsequences that achieve this length. One is [1,17,10,13,10,16,8].


Input: [1,2,3,4,5,6,7,8,9]
Output: 2
```

1.时间：O（N）；空间：O（N）

```
class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {
        if (nums.size() < 2) return nums.size();
        std::vector<int> dp(nums.size(), 0);
        std::vector<int> diff(nums.size(), 0);
        for (int i = 1; i < nums.size(); ++i){
            if (nums[i] == nums[i - 1]) {
                diff[i] = diff[i - 1];  /* 1, 0, 1  这个应该返回 1，而不是 2，因为 0 去掉后变成 1,1 */
            } else
                diff[i] = nums[i] - nums[i - 1] > 0 ? 1 : -1;
        }
        dp[0] = 1;
        for (int i = 1; i < nums.size(); ++i){
            if (diff[i] == 0 || diff[i] == diff[i-1]) dp[i] = dp[i - 1];
            else dp[i] = dp[i - 1] + 1;
        }
        return dp.back();
    }
};
```

2.时间：O（N）；空间：O（1） -->isInit 的判断和赋值操作导致运行时间相比 1 大幅增加

```
class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {
        if (nums.size() < 2) return nums.size();
        int count = 1;
        /* isInit 用于解决 isInc 在开始不确定的情形 ,例如 3,3,3,1,2，这里在 0~2 的下标区间中 isInc 是无法确定的 ,count 计数也无法确定 */
        bool isInc = false, isInit = false;;
        for (int i = 1; i < nums.size(); ++i){
            if (nums[i] > nums[i - 1] && (!isInit || !isInc)){
                count++;
                isInc = true;
                isInit = true;
            } else if (nums[i] < nums[i - 1] && (!isInit || isInc)){
                count++;
                isInc = false;
                isInit = true;
            }
        }
```

```
        return count;
    }
};
```

3.时间：O（N）; 空间：（1） -->相比 2，优化掉 isInit，运行时间和 1 相同，但没有额外空间消耗

```
class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {
        if (nums.size() < 2) return nums.size();
        int startIndex = 1;
        while (nums[startIndex] == nums[startIndex - 1]){
            startIndex++;
        }
        int count = 1;
        bool isInc = nums[startIndex] > nums[startIndex - 1] ? true : false;
        isInc = !isInc;/* 保证 startIndex 可以正确计算 */

        for (int i = startIndex; i < nums.size(); ++i){
            if (nums[i] > nums[i - 1] && !isInc){
                count++;
                isInc = true;
```

```
            } else if (nums[i] < nums[i - 1] && isInc){
                count++;
                isInc = false;
            }
        }
        return count;
    }
};
```