

动态规划：413. 等差数列划分

主要思想：

状态表示：dp[i] 表示以i位置为结尾所有子数组中等差数列的个数

状态转移方程 1. 如果 $c-b = b-a$ $dp[i] = dp[i-1] + 1$

2. 如果 $c-b \neq b-a$ $dp[i] = 0$;

代码：

```
1 class Solution {
2 public:
3     int numberOfArithmeticSlices(vector<int>& nums) {
4         int n = nums.size();
5         int sum = 0;
6         vector<int> dp(n);
7         for(int i = 2; i < n; i++)
8         {
9             if(nums[i] - nums[i-1] == nums[i-1] - nums[i-2])
10            {
11                dp[i] = dp[i-1] + 1;
12            }
13            else
14            {
15                dp[i] = 0;
16            }
17        }
18        for(auto e : dp)
19        {
20            sum += e;
21        }
22        return sum;
23    }
24 };
```

动态规划：978. 最长湍流子数组

主要思想：

状态表示：1.f[i] 表示最后一个为“上升”的最长湍流子数组

2.g[i] 表示最后一个为“下降”的最长湍流子数组

状态转移方程：

1.如果 $num < num-1$ $g[i] = f[i-1] + 1$

2. $num > num-1$ $f[i] = g[i-1] + 1$

代码：

```
1 class Solution {
2 public:
3     int maxTurbulenceSize(vector<int>& arr) {
4         int n = arr.size();
5         vector<int> f(n,1);
6         auto g = f;
7         int ret = 1;
8         for(int i = 1; i < n;i++)
9         {
10             if(arr[i] > arr[i-1])
11             {
12                 f[i] = g[i-1] + 1;
13             }
14             else if(arr[i] < arr[i-1])
15             {
16                 g[i] = f[i-1] + 1;
17             }
18             ret = max(ret,max(f[i],g[i]));
19         }
20         return ret;
21     }
22 }
23 };
```

