1. Maximum Product Subarray

Given an integer array nums, find the contiguous subarray within an array (containing at least one number) which has the largest product.

**Example 1:**

Input: [2,3,-2,4]  
Output: 6  
Explanation: [2,3] has the largest product 6.

**Example 2:**

Input: [-2,0,-1]  
Output: 0  
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.

**解法** 动态规划。需要设置两个dp数组，分别保存到位置i的最大和最小连续乘积，这是因为最小的乘积可能是负的，如果再乘上一个负数会变成比较大数

$$
\mathrm{dp\\_max}[0] = \mathrm{dp\\_min}[0] = \mathrm{nums}[0]\\
\mathrm{dp\\_max[i]} = \max\{\mathrm{nums}[i], \mathrm{dp\\_max}[i-1]\*\mathrm{nums}[i], \mathrm{dp\\_min}[i-1]\*\mathrm{nums}[i]\}\\
\mathrm{dp\\_min[i]} = \min\{\mathrm{nums}[i], \mathrm{dp\\_max}[i-1]\*\mathrm{nums}[i], \mathrm{dp\\_min}[i-1]\*\mathrm{nums}[i]\}
$$

对于数组dp\_max和dp\_min，每次更新只用两个连续的数，因此可以将空间复杂度优化为

class Solution {  
public:  
 int maxProduct(vector<int>& nums) {  
 int pre\_f = nums[0], pre\_g = nums[0];  
 int res = pre\_f;  
 for(int i = 1; i < nums.size(); ++i){  
 int tmp1 = max(nums[i], max(pre\_f\*nums[i], pre\_g\*nums[i]));  
 int tmp2 = min(nums[i], min(pre\_f\*nums[i], pre\_g\*nums[i]));  
 res = max(res, tmp1);  
 pre\_f = tmp1;  
 pre\_g = tmp2;  
 }  
 return res;  
 }  
};