1. Product of Array Except Self

Given an array nums of *n* integers where *n* > 1, return an array output such that output[i] is equal to the product of all the elements of nums except nums[i].

**Example:**

Input: [1,2,3,4]  
Output: [24,12,8,6]

**Constraint:** It’s guaranteed that the product of the elements of any prefix or suffix of the array (including the whole array) fits in a 32 bit integer.

**Note:** Please solve it **without division** and in O(*n*).

**Follow up:** Could you solve it with constant space complexity? (The output array **does not** count as extra space for the purpose of space complexity analysis.)

**解** left-right。计算第i个数左边和右边的乘积

class Solution {  
public:  
 vector<int> productExceptSelf(vector<int>& nums) {  
 vector<int>ans;  
 vector<int>left(nums.size(), 1), right(nums.size(), 1);  
 for(int i = 1; i < nums.size(); ++i){  
 left[i] = left[i-1] \* nums[i-1];  
 }  
 for(int i = nums.size()-1; i > 0; --i){  
 right[i-1] = right[i]\*nums[i];  
 }  
 for(int i = 0; i < nums.size(); ++i){  
 ans.push\_back(left[i]\*right[i]);  
 }  
 return ans;  
 }  
};

优化：在从右往左扫描时，right数组中的数字只用了一次，因此用一个变量R代替即可

class Solution {  
public:  
 vector<int> productExceptSelf(vector<int>& nums) {  
 vector<int>ans(nums.size(), 1);  
 for(int i = 1; i < nums.size(); ++i){  
 ans[i] = ans[i-1] \* nums[i-1];  
 }  
 int R = 1;  
 for(int i = nums.size()-1; i >= 0; --i){  
 ans[i] \*= R;  
 R \*= nums[i];  
 }  
 return ans;  
 }  
};