给定一个整数数组 `nums` ，找出一个序列中乘积最大的连续子序列（该序列至少包含一个数）。

示例 1:

输入： `[2,3,-2,4]`
输出： `6`
解释： 子数组 `[2,3]` 有最大乘积 `6`。

示例 2:

输入： `[-2,0,-1]`
输出： `0`
解释： 结果不能为 `2`, 因为 `[-2,-1]` 不是子数组。

分析，这题是过去做过的，看了下过去的代码，是以贪心算法来写的，对数组进行从左往右和从右往左的两次遍历，用`zs`,标记数组为连续正数的乘法，用`fs`来将所有非`0`的数进行乘法标记。然后当为`0`时，这两数置为`1`.

```
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int max=nums[0],zs=1,fs=1;
        for(int i=0;i<nums.size();i++)
        {
            if(nums[i]<0)
            {
                zs=1;
                fs*=nums[i];
                max=fs>max?fs:max;
            }
            else if(nums[i]==0)
            {
                zs=1;
                fs=1;
                max=0>max?0:max;
            }
            else
            {
                zs*=nums[i];
                fs*=nums[i];
                max=zs>max?zs:max;
                max=fs>max?fs:max;
            }
        }
        zs=1;
        fs=1;
        for(int i=nums.size()-1;i>=0;i--)
```

```
        {
             if(nums[i]<0)
            {
                zs=1;
                fs*=nums[i];
                max=fs>max?fs:max;
            }
            else if(nums[i]==0)
            {
                zs=1;
                fs=1;
                max=0>max?0:max;
            }
            else
            {
                zs*=nums[i];
                fs*=nums[i];
                max=zs>max?zs:max;
                max=fs>max?fs:max;
            }
        }
        return max;
    }
};
```