2 Maximum Product Subarray Leetcode152 maximun subarray leetcode52 最长子序列和 最大子序列乘积

笔记本: Dynamic Programing
创建时间: 10/13/2019 2:54 PM **更新时间:** 10/20/2019 12:22 PM
作者: tanziqi1756@outlook.com

---

152. Maximum Product Subarray

难度 中等 278 收藏 分享 切换为中文

题目描述 评论 (155) 题解(39) New 提交记录

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.
```

```java
class Solution {
    public int maxProduct(int[] nums) {
        // 这题很多坑
        // [-2] output: -2
        // [-4, -3] output: 12
        // [2, 3, -2, 4] output: 6
        // [-2, 0 -1] output: 0
        // 用最大子序列和的方法得稍作修改
        // 因为存在负数乘以负数
        // 所以要记录min_pro

        int temp_min = 1;
        int temp_max = 1;
        int maxPro = Integer.MIN_VALUE;

        for( int i = 0; i < nums.length; i++ ) {
            if( nums[i] < 0 ) {
                int temp = temp_min;
                temp_min = temp_max;
                temp_max = temp;
            }
            temp_min = Math.min(temp_min*nums[i], nums[i]);
            temp_max = Math.max(temp_max*nums[i], nums[i]);
            maxPro = Math.max(temp_max, maxPro);
        }
        return maxPro;
        // 动画演示
        // -> -> -> -> -> -> -> -> -> -> -> ->
        //        initial  2  3 -2  4  -2
        // imax     1      2  6 -2  4   96
        // imin     1      2  3 -12 -48 -2
        // max     ---     2  6  6   6  96
    }
}
```

53. Maximum Subarray

难度 简单 1283 收藏 分享 切换为中文

题目描述 | 评论 (654) | 题解(159) New | 提交记录

Given an integer array `nums` , find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

```
Input: [-2,1,-3,4,-1,2,1,-5,4],
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

```java
class Solution {
    public int maxSubArray(int[] nums) {
        int ans = nums[0];
        int sum = 0;
        for(int i = 0; i < nums.length; i++) {
            if(sum > 0) {
                sum += nums[i];
            }
            else {
                sum = nums[i];
            }
            ans = Math.max(ans, sum);
        }
        return ans;
    }
}
```