1. 题目
2. 题干

给你一个整数数组 nums ，请你找出数组中乘积最大的连续子数组（该子数组中至少包含一个数字），并返回该子数组所对应的乘积。

1. 示例

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。

1. 题解
2. 思路

这是一个很熟悉的题目，找出数组中的连续子数组，要求是乘积最大。之前，我们做过找出和最大的子数组，这里的乘积与和是不一样的，乘积需要考虑数字的正负号的问题。如果当前的数是负数，那么你就需要乘以一个尽量小的数来求得最大乘积；如果当前数为正数，那么你就需要乘以一个尽量大的数以求得最大乘积。因此，我们需要维护两个数组，一个是求第i个数之前的最小连续子数组乘积，一个是求第i个数之前的最大连续子数组乘积。

要求第i个数之前的最大连续子数组乘积就不仅仅需要比较fMax[i-1]\*nums[i]和nums[i]，还需要考虑到当前值可能为负数，添加一个比较项，fMin[i-1]\*nums[i]。

同理，第i个数之前的最小连续子数组乘积还需要添加一个比较项fMax[i-1]\*nums[i]。

最后，从最大值数组中找出最大的即可，

1. 代码实现

Java：

class Solution {

    public int maxProduct(int[] nums) {

        int n = nums.length;

        int[] dp = new int[n];

        if(nums == null || n == 0){

            return 0;

        }

        int[] fMin = nums.clone();

        int[] fMax = nums.clone();

        for(int i = 1; i < n; ++i){

            fMax[i] = Math.max(fMax[i-1]\*nums[i], Math.max(fMin[i-1]\*nums[i], nums[i]));

            fMin[i] = Math.min(fMin[i-1]\*nums[i], Math.min(fMax[i-1]\*nums[i], nums[i]));

        }

        int ans = nums[0];

        for(int i = 0; i < n; ++i){

            ans = Math.max(ans, fMax[i]);

        }

        return ans;

    }

}