Subarray Product Less Than K

LeetCode Admin Oct 23, 2017

Approach #1: Binary Search on Logarithms [Accepted]

Intuition

Because $\log(\prod_i x_i) = \sum_i \log x_i$, we can reduce the problem to subarray *sums* instead of subarray products. The motivation for this is that the product of some arbitrary subarray may be way too large (potentially `1000^50000`), and also dealing with sums gives us some more familiarity as it becomes similar to other problems we may have solved before.

Algorithm

After this transformation where every value `x` becomes `log(x)`, let us take prefix sums `prefix[i+1] = nums[0] + nums[1] + ... + nums[i]`. Now we are left with the problem of finding, for each `i`, the largest `j` so that `nums[i] + ... + nums[j] = prefix[j] - prefix[i] < k`.

Because `prefix` is a monotone increasing array, this can be solved with binary search. We add the width of the interval `[i, j]` to our answer, which counts all subarrays `[i, k]` with `k <= j`.

Python

```python
class Solution(object):
    def numSubarrayProductLessThanK(self, nums, k):
        if k == 0: return 0
        k = math.log(k)

        prefix = [0]
        for x in nums:
            prefix.append(prefix[-1] + math.log(x))

        ans = 0
        for i, x in enumerate(prefix):
            j = bisect.bisect(prefix, x + k - 1e-9, i+1)
            ans += j - i - 1
        return ans
```

Java

```java
class Solution {
    public int numSubarrayProductLessThanK(int[] nums, int k) {
        if (k == 0) return 0;
        double logk = Math.log(k);
        double[] prefix = new double[nums.length + 1];
        for (int i = 0; i < nums.length; i++) {
            prefix[i+1] = prefix[i] + Math.log(nums[i]);
        }

        int ans = 0;
        for (int i = 0; i < prefix.length; i++) {
            int lo = i + 1, hi = prefix.length;
            while (lo < hi) {
                int mi = lo + (hi - lo) / 2;
                if (prefix[mi] < prefix[i] + logk - 1e-9) lo = mi + 1;
                else hi = mi;
            }
            ans += lo - i - 1;
```

```
            }
            return ans;
        }
    }
```

Complexity Analysis

- Time Complexity: $O(N \log N)$, where N is the length of `nums`. Inside our for loop, each binary search operation takes $O(\log N)$ time.
- Space Complexity: $O(N)$, the space used by `prefix`.

Approach #2: Sliding Window [Accepted]

Intuition

For each `right`, call `opt(right)` the smallest `left` so that the product of the subarray `nums[left] * nums[left + 1] * ... * nums[right]` is less than `k`. `opt` is a monotone increasing function, so we can use a sliding window.

Algorithm

Our loop invariant is that `left` is the smallest value so that the product in the window `prod = nums[left] * nums[left + 1] * ... * nums[right]` is less than `k`.

For every right, we update `left` and `prod` to maintain this invariant. Then, the number of intervals with subarray product less than `k` and with right-most coordinate `right`, is `right - left + 1`. We'll count all of these for each value of `right`.

Python

```
class Solution(object):
    def numSubarrayProductLessThanK(self, nums, k):
        if k <= 1: return 0
        prod = 1
        ans = left = 0
        for right, val in enumerate(nums):
            prod *= val
            while prod >= k:
                prod /= nums[left]
                left += 1
            ans += right - left + 1
        return ans
```

Java

```
class Solution {
    public int numSubarrayProductLessThanK(int[] nums, int k) {
        if (k <= 1) return 0;
        int prod = 1, ans = 0, left = 0;
        for (int right = 0; right < nums.length; right++) {
            prod *= nums[right];
            while (prod >= k) prod /= nums[left++];
            ans += right - left + 1;
        }
        return ans;
    }
}
```

Complexity Analysis

- Time Complexity: $O(N)$, where N is the length of `nums`. `left` can only be incremented at most `N` times.
- Space Complexity: $O(1)$, the space used by `prod`, `left`, and `ans`.

Share

Comments (90)

Sort by: **Best**

Type comment here... (Markdown supported)

Preview Comment

Article Commenting Rules

1. This comment section is for questions and comments regarding this **LeetCode article**. All posts must respect our LeetCode Community Rules.

2. Concerns about errors or bugs in the article, problem description, or test cases should be posted on LeetCode Feedback, so that our team can address them.

prabhjyot28 Apr 28, 2020

Note that the Sliding window approach is only valid because numbers are positive.

329 Show 4 Replies Reply

willye Sep 22, 2019

Read more

577 Show 27 Replies Reply

dance-henry Mar 15, 2020

For those who are confused with ans += right - left + 1 in "Sliding Window" approach. Because the element pointed by the "right" is the new one we are investigating, as long as we find a subarray with prod < k after being divided by nums[left], ans should be increased by the length of the subarray because the product is ever increasing. For example, [10,5,2,6], when right points to 6, left points to 5, ans added 3, which is due to [5,2,6],[2.6],and[6].

62 Show 4 Replies Reply

StriverKaDriver Aug 13, 2021

People underestimate the sliding window whereas it is frequently asked in interviews and it can also be quite challenging at times.

59 Reply

lelouchmatlab2018 Jul 21, 2018

solution 2 I think for line "while (prod >= k) prod /= nums[left++];" it should also guard "left<=right". (I got runTimeError without that for a test case [1,1,1])

48 Show 6 Replies Reply

obpb4ae4twxy Oct 25, 2020

Read more

13 Show 1 Replies Reply

shreyas88 Oct 24, 2017

Sorry if this is a naive question, why do you use 1e-9 term in the binary search expression below?
prefix[mi] < prefix[i] + logk - 1e-9

15 Show 7 Replies Reply

park29 Jul 30, 2019

What a beautiful implementation!!! Learned a lot

15 Reply

rajincse Oct 24, 2017

@awice right. I missed the if(k<=1) return 0; part. Thanks.

5 Show 1 Replies Reply

professionalusername Nov 16, 2018

neither of these solutions worked for a hackerrank problem that is an exact copy. There must be some hole in the test cases here...
I even tried things like long long, rounding, and other solutions from geeksforgeeks, my own, etc. But 4 test cases on hackerrank always failed (wrong answer, not timeout or other). I believe the constraints were the same.

7 Show 3 Replies Reply

1 2 3 4 5 6 ... 9