Subarray Product Less Than K

Your are given an array of positive integers `nums`.

Count and print the number of (contiguous) subarrays where the product of all the elements in the subarray is less than `k`.

Example 1:

```
Input: nums = [10, 5, 2, 6], k = 100
Output: 8
Explanation: The 8 subarrays that have product less than 100 are: [10], [5], [2], [6]
, [10, 5], [5, 2], [2, 6], [5, 2, 6].
Note that [10, 5, 2] is not included as the product of 100 is not strictly less than
k.
```

Note:

- `0 .`
- `0 .`
- `0 .`

Solution 1

Here are two solutions with similar ideas.

For nums[i], count range [left, i], whose product is just < k

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

For nums[i], count range [i, j)

```
class Solution {
public:
    int numSubarrayProductLessThanK(vector<int>& nums, int k) {
        int n = nums.size(), ans = 0, prod = 1, j = 0;
        for (int i = 0; i < n; i++) {
            if (i > 0 && i <= j)
                prod =  prod/nums[i-1];
            else
                j = i;
            while (j < n &&  prod*nums[j] < k) prod *= nums[j++];
            ans += j-i;
        }
        return ans;
    }
};
```

written by zestypanda original link here

Solution 2

Thanks for @ohazyi and @awice pointing it out. I've updated my code below and it passes all test cases now.

```
public int numSubarrayProductLessThanK(int[] nums, int k) {
            if (k < 2) {
                return 0;
            }
            int result = 0;
            int product = 1;
            for (int i = 0, right = 0; right < nums.length; right++) {
                product *= nums[right];
                while (i < nums.length && product >= k) {
                    product /= nums[i++];
                }
                result += right - i + 1;
            }
            return result;
        }
```

written by fishercoder original link here

Solution 3

C# solution

Iterate over the array and add the length of the maximum subarray for each location.

For e.g.input array nums = [10, 5, 2, 6] and k = 100

lengths at each location are [1, 2, 2, 3]
return sum of all lengths, that is 1 + 2 + 2 + 3 = 8;

For corner cases like k = 0 and k = 1 return meaningful values.

```
public int NumSubarrayProductLessThanK(int[] nums, int k) {
            if (nums == null || k == 0 || k == 1)
            {
                return 0;
            }

            int p = 1;
            int i = 0;
            int len = 0;
            int count = 0;

            while (i < nums.Length)
            {
                p *= nums[i];

                if (p < k)
                {
                    count += ++len;
                }
                else
                {
                    while (p >= k)
                    {
                        p /= nums[i - len];
                        len--;
                    }

                    count += ++len;
                }

                i++;
            }

            return count;
    }
```

written by jainchethan87 original link here