<https://leetcode.com/problems/subarray-product-less-than-k/>

Given an array of integers nums and an integer k, return *the number of contiguous subarrays where the product of all the elements in the subarray is strictly 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.

**Example 2:**

Input: nums = [1,2,3], k = 0

Output: 0

**Constraints:**

1 <= nums.length <= 3 \* 10^4

1 <= nums[i] <= 1000

0 <= k <= 10^6

**Attempt 1: 2022-09-09 (30min, spend time to figure out condition avoid forever loop)**

class Solution {

    public int numSubarrayProductLessThanK(int[] nums, int k) {

        int product = 1;

        int count = 0;

        int len = nums.length;

        int i = 0;

        // e.g nums = {10,5,2,6}, k = 100

        // i = 0, j = 0 -> product = 10 < 100, count += (0 - 0 + 1) = 1

        // {10}

        // i = 0, j = 1 -> product = 10 \* 5 < 100, count += (1 - 0 + 1) = 3

        // {10} || new add {10,5}, {5}

        // i = 0, j = 2 -> product = 10 \* 5 \* 2 >= 100, product = 100 / 10 = 10, i = 1

        // count += (2 - 1 + 1) = 5

        // {10}, {10,5}, {5} || new add {2}, {5,2}

        // i = 1, j = 3 -> product = 5 \* 2 \* 6 < 100, count += (3 - 1 + 1) = 8

        // {10}, {10,5}, {5}, {2}, {5,2} || new add {5,2,6}, {2,6}, {6}

        // ------------------------------------------------------------

        // e.g nums = {1,2,3}, k = 0

        // i = 0, j = 0 -> product = 1 >= 0, product = 1 / 1 = 1, i = 1

        // count += (0 - 1 + 1) = 0

        // i = 1, j = 1 -> product = 2 >= 0, product = 2 / 2 = 1, i = 2

        // count += (1 - 2 + 1) = 0

        // i = 2, j = 2 -> product = 3 >= 0, product = 3 / 3 = 1, i = 3

        // count += (2 - 3 + 1) = 0

        for(int j = 0; j < len; j++) {

            product \*= nums[j];

            // Must include i <= j, test out by nums = {1,2,3}, k = 0

            // if no i <= j, for product /= nums[i], i will increase to 3

            // and index out of boundary as product >= k always satisfied

            // and loop forever result into i keep increasing

            while(i <= j && product >= k) {

                product /= nums[i];

                i++;

            }

            // The distance between left and right end pointer equal to newly

            // add number of subarrays

            count += (j - i + 1);

        }

        return count;

    }

}

Space Complexity: O(1)

Time Complexity: O(n)