6299. Minimum Cost to Split an Array

My Submissions (/contest/weekly-contest-329/problems/minimum-cost-to-split-an-array/submissions/) Back to Contest (/contest/weekly-contest-329/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

You are given an integer array `nums` and an integer `k`.

Split the array into some number of non-empty subarrays. The **cost** of a split is the sum of the **importance value** of each subarray in the split.

Let `trimmed(subarray)` be the version of the subarray where all numbers which appear only once are removed.

- For example, `trimmed([3,1,2,4,3,4]) = [3,4,3,4].`

The **importance value** of a subarray is `k + trimmed(subarray).length`.

- For example, if a subarray is `[1,2,3,3,3,4,4]`, then `trimmed( [1,2,3,3,3,4,4]) = [3,3,3,4,4].` The importance value of this subarray will be `k + 5`.

Return *the minimum possible cost of a split of* `nums`.

A **subarray** is a contiguous **non-empty** sequence of elements within an array.

Example 1:

```
Input: nums = [1,2,1,2,1,3,3], k = 2
Output: 8
Explanation: We split nums to have two subarrays: [1,2], [1,2,1,3,3].
The importance value of [1,2] is 2 + (0) = 2.
The importance value of [1,2,1,3,3] is 2 + (2 + 2) = 6.
The cost of the split is 2 + 6 = 8. It can be shown that this is the minimum possible cost among all the possible splits.
```

Example 2:

```
Input: nums = [1,2,1,2,1], k = 2
Output: 6
Explanation: We split nums to have two subarrays: [1,2], [1,2,1].
The importance value of [1,2] is 2 + (0) = 2.
The importance value of [1,2,1] is 2 + (2) = 4.
The cost of the split is 2 + 4 = 6. It can be shown that this is the minimum possible cost among all the possible splits.
```

Example 3:

```
Input: nums = [1,2,1,2,1], k = 5
Output: 10
Explanation: We split nums to have one subarray: [1,2,1,2,1].
The importance value of [1,2,1,2,1] is 5 + (3 + 2) = 10.
The cost of the split is 10. It can be shown that this is the minimum possible cost among all the possible splits.
```

Constraints:

- `1 <= nums.length <= 1000`
- `0 <= nums[i] < nums.length`
- `1 <= k <= 10^9`

JavaScript

```
const minCost = (a, k) => {
    let n = a.length, max = Math.max(...a), dp = Array(n + 1).fill(Number.MAX_SAFE_INTEGER);
    dp[0] = 0;
    for (let i = 0; i < n; i++) {
        let f = Array(max + 1).fill(0);
        for (let j = i; j < n; j++) {
            f[a[j]]++;
            let cost = 0;
```

```
            for (let x = 0; x <= max; x++) cost += (f[x] == 1) ? 0 : f[x];
            dp[j + 1] = Math.min(dp[i] + cost + k, dp[j + 1]);
        }
    }
    return dp[n];
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/882833304/) | More Details > (/submissions/detail/882833304/)

Share your acceptance!