(/) Explore(/explore/) Problems(/problemset/all/) Interview New (/contest/) Contest Discuss(/discuss/) Store ☆ Premium (/subscribe?ref=nb_npl) 0 if-there-is-a-valid-partition-for-the-array/)

7023. Apply Operations to Maximize Score

My Submissions (/contest/weekly-contest-358/problems/apply-operations-to-maximize-score/submissions/) Back to Contest (/contest/weekly-contest-358/)

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

You are given an array `nums` of `n` positive integers and an integer `k`.

Initially, you start with a score of `1`. You have to maximize your score by applying the following operation at most `k` times:

- Choose any **non-empty** subarray `nums[l, ..., r]` that you haven't chosen previously.
- Choose an element `x` of `nums[l, ..., r]` with the highest **prime score**. If multiple such elements exist, choose the one with the smallest index.
- Multiply your score by `x`.

Here, `nums[l, ..., r]` denotes the subarray of `nums` starting at index `l` and ending at the index `r`, both ends being inclusive.

The **prime score** of an integer `x` is equal to the number of distinct prime factors of `x`. For example, the prime score of `300` is `3` since `300 = 2 * 2 * 3 * 5 * 5`.

Return *the **maximum possible score** after applying at most* `k` *operations*.

Since the answer may be large, return it modulo $10^9 + 7$.

Example 1:

```
Input: nums = [8,3,9,3,8], k = 2
Output: 81
Explanation: To get a score of 81, we can apply the following operations:
- Choose subarray nums[2, ..., 2]. nums[2] is the only element in this subarray. Hence, we multiply the score by nums[2]. The
- Choose subarray nums[2, ..., 3]. Both nums[2] and nums[3] have a prime score of 1, but nums[2] has the smaller index. Hence,
It can be proven that 81 is the highest score one can obtain.
```

Example 2:

```
Input: nums = [19,12,14,6,10,18], k = 3
Output: 4788
Explanation: To get a score of 4788, we can apply the following operations:
- Choose subarray nums[0, ..., 0]. nums[0] is the only element in this subarray. Hence, we multiply the score by nums[0]. The
- Choose subarray nums[5, ..., 5]. nums[5] is the only element in this subarray. Hence, we multiply the score by nums[5]. The
- Choose subarray nums[2, ..., 3]. Both nums[2] and nums[3] have a prime score of 2, but nums[2] has the smaller index. Hence,
It can be proven that 4788 is the highest score one can obtain.
```

Constraints:

- $1 <= nums.length == n <= 10^5$
- $1 <= nums[i] <= 10^5$
- $1 <= k <= min(n * (n + 1) / 2, 10^9)$

JavaScript

```javascript
const multi_mod = (x, y, mod) => Number(ll(x) * ll(y) % ll(mod));
// const multi_mod = (x, y, mod) => x * y % mod;

const ll = BigInt;
const powmod = (a, b, mod) => { let r = 1; while (b > 0) { if (b & 1) r = multi_mod(r, a, mod); b >>= 1; a =
multi_mod(a, a, mod); } return r; };

const mod = 1e9 + 7;
const maximumScore = (a, k) => {
    let max = Math.max(...a), cnt = [], d = [], lpf = LeastPrimeFactors(max + 1);
    a.map(x => {
        let f = factorizationLPF(x, lpf);
        cnt.push(f.length);
    });
    let L = stockPlan(cnt), R = stockPlanRev(cnt), res = 1;
    a.map((x, i) => d.push([cnt[i], i, (i - L[i]) * (R[i] - i), x]));
    d.sort((x, y) => y[3] - x[3]);
    for (const [, , t, x] of d) {
        let use = Math.min(t, k);
        k -= use;
        let pow = powmod(x, use, mod);
        res = multi_mod(res, pow, mod);
    }
    return res;
};

const LeastPrimeFactors = (n) => {
    let lpf = Array(n + 1).fill(0);
    for (let i = 2; i <= n; i++) {
        if (lpf[i] == 0) {
            lpf[i] = i;
            for (let j = i * i; j <= n; j += i) {
                if (lpf[j] == 0) lpf[j] = i;
            }
        }
    }
    return lpf;
};

const factorizationLPF = (n, lpf) => {
    let f = Array(9), i = 0;
    while (lpf[n] > 0) {
        let p = lpf[n];
        if (i == 0 || p != f[i - 1][0]) {
            f[i++] = [p, 1];
        } else {
            f[i - 1][1]++;
        }
        n /= p;
    }
    return f.slice(0, i);
};

const stockPlan = (a) => {
    let n = a.length, span = Array(n).fill(0);
    for (let i = 0; i < n; i++) {
        span[i] = i - 1;
        while (span[i] >= 0 && a[span[i]] < a[i]) span[i] = span[span[i]];
    }
    return span;
};

const stockPlanRev = (a) => {
    let n = a.length, span = Array(n).fill(0);
    for (let i = n - 1; i >= 0; i--) {
        span[i] = i + 1;
        while (span[i] < n && a[span[i]] <= a[i]) span[i] = span[span[i]];
    }
    return span;
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/1021265662/) More Details (/submissions/detail/1021265662/)

Share your acceptance!

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)