Codeforces 1167E (2100)

KH4RG0SH

21 January 2026

https://codeforces.com/contest/1167/problem/E
Accepted: https://codeforces.com/contest/1167/submission/359105151

§1 Solution

§1.1 Explanation

Definition 1.1. Denote a pair (l, r) as *valid* if deleting all values such that

$$l \leq a_i \leq r$$

from the array results in a non-decreasing array.

Definition 1.2. Denote a number p as *prefix-good* if the pair $(p+1, x)$ is *valid*.

Lemma 1.3
If a pair (l, r) is *valid* then, $(l-1, r)$ and $(l, r+1)$ are *valid* too.

Proof. If (l, r) is valid then the remainder sequence of the array looks something like

$$1 \cdots 1 \,|\, 2 \cdots 2 \,|\, \cdots \,(l-1) \,|\, (r+1) \cdots \,|\, x \cdots x$$

Hence, if we remove either of $(l-1)$ or $(r+1)$ from the remaining sequence, it is still non-decreasing. This implies that $(l-1, r)$ and $(l, r+1)$ are valid too. □

Lemma 1.4
If p is *prefix-good* then for any $q < p$, we will have q as *prefix-good*. Similarly, if p is not *prefix-good* then for any $q > p$, we cannot have q as *prefix-good*.

Proof. For the first part, if p is prefix-good then all the numbers from 1 to p occur in a non-decreasing manner in the array. Hence any prefix of it will also be non-decreasing.

For the second part, if p is not prefix-good then the numbers 1 to p do not occur in a non-decreasing manner. So for any $q > p$, the numbers from 1 to q cannot be non-decreasing. The key fact used here is that any prefix of a non-decreasing sequence is also non-decreasing, and if a prefix of a sequence is not non-decreasing then the sequence cannot be non-decreasing. □

Lemma 1.5

If p is not *prefix-good* then for any $p < l \leq r$, the pair (l, r) can never be *valid*.

Proof. If (l, r) is valid then $(l-1)$ has to be prefix-good, but $l - 1 \geq p$ is not prefix-good, which is a contradiction. □

¶ Back to the Problem

1. Compute the first and last occurrence for each number $1 \leq i \leq x$. This can be done in a simple $\mathcal{O}(n)$ scan.
2. Compute the max value p such that p is prefix-good. To do this, we just have to keep comparing if
$$\text{last}[i-1] < \text{first}[i]$$
3. Similarly, we want to compute the minimum value s such that s is prefix-good on the reversed array. So basically this is the notion of suffix-good, which can be defined similarly. We just have to compute
$$\text{last}[i] < \text{first}[i+1]$$
4. if $p \geq s$, it should mean that $p = x$ and $s = 1$. If this does not hold then $p \geq s$ is impossible. We will always otherwise have $p < s$. If $p \geq s$ then all (l, r) are valid. Otherwise, we proceed to the next step.
5. Now we have to understand that if (l, r) is valid then $1 \leq l \leq p+1$ and $s - 1 \leq r \leq x$. If any of this is violated, then the remaining sequence cannot be non-decreasing. Note how $p + 1 \leq s - 1 \implies l \leq r$ which looks fine.
6. Now for each l we want to find the minimum r such that (l, r) is valid. If suppose g is a function that tells us the minimum r for l (that is $g(l) = r$), then the answer would be
$$\sum_{i=1}^{p+1} (x - (g(i) - 1))$$
7. To compute $g(i)$ we need to realise that if suppose m is the maximum number occurring in $[1, \text{last}[i-1]]$, then r has to be atleast m. So $r \geq \max(i, s-1, m)$. So we have a way to compute $g(i)$
$$g(i) = \max(i, s-1, \max([1, \text{last}[i-1]]))$$

§1.2 Code

```
void solve() {
    // 1. input of the problem
    ll n, x;
    std::cin >> n >> x;

    ll a[n + 1];
    for (ll i = 1; i <= n; i++) {
        std::cin >> a[i];
    }
```

```
    // 2. compute the first and last
    std::vector<ll> first(x + 2, 0LL), last(x + 2, 0LL);
    for (ll i = 1; i <= n; i++) {
        if (first[a[i]] == 0) {
            first[a[i]] = i;
            last[a[i]] = i;
        } else {
            last[a[i]] = i;
        }
    }

    // 3. compute p and s
    ll b[n + 1];
    for (ll i = 1; i <= n; i++) {
        b[i] = a[i];
    }
    std::sort(b + 1, b + n + 1);
    ll p = b[1], s = b[n];

    // compute p here
    for (ll i = 2; i <= n; i++) {
        if (b[i - 1] == b[i]) {
            if (i == n) {
                p = x;
            }
            continue;
        }
        if (last[b[i - 1]] < first[b[i]]) {
            p = b[i];
            if (i == n) {
                p = x;
            }
        } else {
            p = b[i] - 1;
            break;
        }
    }

    // compute s here
    for (ll i = n - 1; i > 0; i--) {
        if (b[i] == b[i + 1]) {
            if (i == 1) {
                s = 1;
            }
            continue;
        }
        if (last[b[i]] < first[b[i + 1]]) {
            s = b[i];
            if (i == 1) {
                s = 1;
            }
        } else {
            s = b[i] + 1;
            break;
        }
    }

    // 4. compute g(i) for each i and get final ans
```

```
    ll ans = 0;
    ll mx[n + 1];
    mx[0] = 0LL;

    for (ll i = 1; i <= n; i++) {
        mx[i] = std::max(mx[i - 1], a[i]);
    }

    for (ll i = 1; i <= x; i++) {
        if (last[i] == 0) {
            last[i] = last[i - 1];
        }
    }

    for (ll i = 1; i <= std::min(x, p + 1); i++) {
        ans += (x + 1 - std::max({i, s - 1, mx[last[i - 1]]}));
    }

    std::cout << ans << '\n';
}
```