Good Array 1 - Upsolve

znzryb

April 16, 2026

我发现，并不是每道题目都是要去对拍的。
不过，其实，我们可以想一下
采用一个这个贪心做法是显然可以的：
就是只要这个前缀不是好数组，我们就把这个前缀的最大值加一，改成一个新的数。

```
bool check(const vector<ll> &a) {
    for (int l = 0; l < SZ(a); ++l) {
        for (int r = l; r < SZ(a); ++r) {
            map<ll, ll> mp;
            for (int i = l; i <= r; ++i) {
                mp[a[i]]++;
            }
            bool ok = false;
            for (auto [u,cnt]: mp) {
                if (cnt == 1) {
                    ok = true;
                }
            }
            if (!ok) {
                return false;
            }
        }
    }
    return true;
}
Solve() {
    cin >> N;
    A.resize(N);
    for (int i = 0; i < N; ++i) {
        cin >> A[i];
    }
    ll cur = *max_element(all(A));
    ll ans = 0;
    for (int i = 0; i < N; ++i) {
        if (!check(vector<ll>(A.begin(), A.begin() + i + 1))) {
            ++cur;
            A[i] = cur;
            ++ans;
        }
    }
    cout << ans << "\n";
}
```

不过这个东西比较慢，
TLE 提交记录
当然，我们可以稍微优化一下我们的这个 check 函数，**我们不需要每次都从头检查到尾**，我们只需要检查这个前缀的最后一个数就好了。
当然，他好像卡了 log，最后一个这个两个点还是过不了。
TLE 提交记录 2

```
        bool check(const vector<ll> &a) {
```

```
        for (int l = 0; l < SZ(a); ++l) {
            int r = SZ(a) - 1;
            map<ll, ll> mp;
            for (int i = l; i <= r; ++i) {
                mp[a[i]]++;
            }
            bool ok = false;
            for (auto [u,cnt]: mp) {
                if (cnt == 1) {
                    ok = true;
                }
            }
            if (!ok) {
                return false;
            }
        }
        return true;
    }
```

当然，这个 check 函数还可以继续优化，这个 map 没有必要每次都重建，可以继续进行增量维护。
AC 提交记录

```
    bool check(const vector<ll> &a) {
        mp.clear();
        // 反方向遍历
        for (int l = SZ(a) - 1; l >= 0; --l) {
            // int r = SZ(a) - 1;
            // mp 进行增量维护
            mp[a[l]]++;
            bool ok = false;
            for (auto [u,cnt]: mp) {
                if (cnt == 1) {
                    ok = true;
                }
            }
            if (!ok) {
                return false;
            }
        }
        return true;
    }
```