**Test 1:**

**Input**

5

1 **2 3 3 2**

[1, 2]: constant

[2, 3]: constant

[1, 5]: Not constant tại vì M = 3, m = 1 -=> M - n > 1

[2, 5]: constant

⇒ **4**

**Test 2:**

**Input**

11

5 4 5 5 6 **7 8 8 8 7** 6

[1, 4]: M - m = 5 - 4 = 1 ⇒ len = 4

[5, 6]: M - m = 7 - 6 = 1 ⇒ len = 2

[6, 10]: M - m = 8 - 7 = 1 ⇒ len = 5

⇒ **5**

[L, R]: M - m <= 1

Th1: M == m

Th2: M == m + 1

Mảng đã cho có độ chênh lệch giữa 2 phần tử liên tiếp <= 1

⇒ Nếu đoạn từ [L, R] chỉ có tối đa là 2 giá trị khác nhau, thì đoạn L, R sẽ thỏa mãn

**Two-pointer: giúp chúng ta kiểm soát 2 điểm đầu cuối của một đoạn con liên tiếp**

**Kiểm soát số lượng giá trị khác nhau:**

f: mảng đếm phân phối

f[x] là số lượng phần tử mang giá trị x trong đoạn từ i cho tới j đang xét

**Cập nhật số lượng phần tử khác nhau:**

**Khi chúng ta thêm a[j] vào trong đoạn từ i tới j:**

* Tăng giá trị của f[a[j]] lên 1
* Nếu f[a[j]] == 1: Tức là a[j] chỉ vừa được thêm vào → tăng số lượng giá trị khác nhau

**Khi chúng ta xóa a[i] khỏi đoạn từ i tới j:**

* Giảm giá trị của f[a[i]] đi 1
* Nếu f[a[i]] == 0: Tức là các phần tử mang giá trị a[i] đã bị xóa đi hoàn toàn khỏi đoạn từ i → j ⇒ Giảm số lượng giá trị khác nhau

**Giải thuật:**

**Bước 1:** Nhập dữ liệu.

**Bước 2:** Tạo mảng f[] (có thể để giới hạn là 10^5 + 1 phần tử hoặc max(a) + 1 phần tử) là mảng đếm phân phối, unique (số lượng các giá trị khác nhau trong đoạn i, j mà đang xét)

**Bước 3:** Khởi tạo cho giá trị i = 1.

**Bước 4:** Cho j chạy từ đầu mảng tới cuối mảng.

* Nếu phần tử hiện tại ở vị trí j lần đầu tiên xuất hiện (giá trị trong mảng phân phối bằng 0) thì tăng biến đếm số lượng phân biệt lên 1 (tăng unique lên 1).
* Nếu số lượng phần tử phân biệt lúc này lớn hơn 2, thực hiện rút ngắn đoạn bằng biến i. Đồng thời cập nhật số lượng phần tử phân biệt.
* So sánh đoạn hiện tại [i, j] có kích thước j - i + 1 với độ dài lớn nhất hiện có và tiến hành cập nhật. // Cập nhập kết quả.

**Bước 5:** In ra kết quả, kết thúc chương trình.

**Mã giả:**

**read(n);**

**for i = 1 to n:**

**read(a[i]);**

**f = []; // 10^5 + 1 hoặc max(a) + 1 phần tử = 0**

**unique = 0;**

**i = 1;**

**result = 1;**

**for j = 1 to n: {**

**f[a[j]] += 1;**

**if f[a[j]] == 1:**

**unique += 1;**

**while unique > 2: {**

**f[a[i]] -= 1;**

**if f[a[i]] == 0:**

**unique -= 1;**

**i += 1;**

**}**

**result = max(result, j - i + 1);**

**}**

**print(result);**

**Đánh giá:**

Không gian: **O(n + max\_a)**

Thời gian: **O(3 \* n) = O(n)**