**Ví dụ**

**Input**

4 2

1 2 2 3

**Output**

1 2

→ 3 4 → Cũng là một đáp án được chấp nhận

→ 2 4 → Không thể coi là đáp án

**Input**

8 3

1 **1 2 2 3** 3 4 5

**Output**

2 5

→ 6 8 cũng là một kết quả

→ 5 8 → Không thể coi là đáp án

**Input**

7 4

4 7 7 4 7 4 7

**Output**

-1 -1

**Cách đơn giản nhất:**

* Cách 1: Duyệt từng đoạn i → j, rồi đếm số lượng phần tử phân biệt trong đoạn từ i → j
  + Duyệt i \* duyệt j \* đếm: O(N^3)
* Duyệt i, rồi duyệt j, và dùng cái vòng for duyệt j để đếm số lượng phần tử phân biệt.
  + Duyệt i \* duyệt j: O(N^2)

⇒ Sử dụng **Two pointer**

* Tìm vị trí j đầu tiên mà đoạn từ 1 → j có đúng K phần tử phân biệt

→ Tìm đoạn i = 1 → j đang có K phần tử phân biệt

→ Tăng i lên, tới khi nào mà đoạn từ i → j (đoạn i → j chứa đúng K phần tử phân biệt) không chứa một đoạn con nào có đúng K phần tử phân biệt

**Mảng đếm phân phối**

f[x] là số lượng phần tử x trong đoạn từ i → j

* Khi f[x] == 0: thì tức là trong đoạn từ i → j không có phần tử x
* Khi f[x] > 0: thì tức là trong đoạn từ i → j có tồn tồn tại x

→ Số lượng phần tử phân biệt trong đoạn từ i → j = số lượng giá trị x mà f[x] > 0

**Khi tăng j lên → kết nạp a[j] vào**

* f[a[j]] += 1
* Nếu f[a[j]] == 1: tức là a[j] chỉ vừa được thêm vào → số lượng phần tử phân biệt tăng lên 1

**Khi tăng i lên: → loại bỏ a[i] đi**

* f[a[i]] -= 1
* Nếu f[a[i]] == 0: tức là a[i] chỉ vừa bị bỏ đi

→ số lượng phần tử phân biệt giảm đi 1

**Giải thuật**

**Bước 1:** Nhập dữ liệu.

**Bước 2:** Sử dụng biến chạy j để chạy tìm đoạn đầu tiên chứa đủ K số phân biệt.

**Bước 3:** Sử dụng biến chạy i chạy từ dưới lên nhằm rút ngắn đoạn tìm được, đảm bảo trong [i, j] không còn bất cứ đoạn con nào cũng thỏa yêu cầu.

**Bước 4:** In ra kết quả bài toán nằm trong hai biến i và j, nếu không có kết quả thì in ra -1 -1

**Mã giả**

**read(n, k);**

**for i = 1 to n:**

**read(a[i]); //O(n)**

**f = []; // 10^5 + 1 phần tử, ban đầu khởi tạo = 0**

**unique = 0**

**i = 1**

**for j = 1 to n: { // O(N)**

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

**unique += 1;**

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

**while unique == k: {**

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

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

**print(i, j);**

**exit();**

**}**

**i += 1;**

**}**

**}**

**print(-1, -1);**

**Độ phức tạp:**

**Thời gian: O(N + N) = O(2\*N) = O(N)**

**Không gian: O(max(a) + N) // max(a) có thể lên tới 10^5**