

Array

Big-O Blue - Lecture 02: **Algorithmic Complexity**

Tóm tắt đề bài

Tóm tắt đề bài

Cho một mảng **A**, gồm **N** số nguyên. Tìm một đoạn con tối thiểu **[L,R]** thỏa:

- 1. Mảng con **[L,R]** của **A** chứa đúng **K** phần tử phân biệt.
- 2. Là đoạn con tối thiểu: trong đoạn **[L,R]** không có đoạn con nào cũng chứa **K** phần tử phân biệt.

Lưu ý: Output là đoạn **[L,R]** bất kỳ thỏa 2 điều kiện trên, đoạn **[L,R]** không nhất thiết có độ dài **M = R-L+1** nhỏ nhất trên toàn mảng.

Mô tả Input/Output

Input

N K

$A_1 A_2 \dots A_N$

$1 \leq N, K \leq 10^5$

$1 \leq A[i] \leq 10^5$

Output

L R

trong đó $[L, R]$ là đoạn con tối thiểu

chứa đúng K phần tử phân biệt, $1 \leq L \leq R \leq N$

nếu không tồn tại đoạn thỏa mãn điều kiện
→ “-1 -1”

Giải thích ví dụ

Ví dụ 1

Input:

4 2

1 2 2 3

Output:

1 2

[L,R]	Mảng con	Thỏa mãn
[1,2]	[1,2]	✓
[1,3]	[1,2,2]	✗ : chứa đoạn [1,2] chứa 2 phần tử phân biệt
[2,4]	[2,2,3]	✗ : chứa đoạn [3,4] chứa 2 phần tử phân biệt
[3,4]	[2,3]	✓

Output = {1 2, 3 4}

Ví dụ 2

Input:

8 3

1 1 2 2 3 3 4 5

Output:

2 5

[L,R]	Mảng con	Thỏa mãn
[2,5]	[1,2,2,3]	✓ 1. Chứa đúng 3 phần tử phân biệt. 2. Không chứa đoạn con chứa 3 phần tử phân biệt.
[4,7]	[2,3,3,4]	
[6,8]	[3,4,5]	

Output = {2 5, 4 7, 6 8}

Ví dụ 3

Input:

7 4

4 7 7 4 7 4 7

Output:

-1 -1

Toàn bộ mảng chỉ chứa 2 phần tử phân biệt

→ Không tồn tại mảng con nào chứa 4 phần tử phân biệt.

Hướng dẫn giải

Ý tưởng

Output là bất kỳ đoạn con nào thỏa đủ 2 điều kiện.
→ Chọn in đoạn con đầu tiên thỏa.

Sử dụng two pointers, 1 con trỏ **i** chạy từ trái sang để kiểm được đoạn có đúng **k** phần tử phân biệt → khi kiểm đủ ta sẽ dùng 1 con trỏ **j** sẽ chạy để rút ngắn đoạn đó (nếu có thể)

Điều kiện 1: Đoạn con chứa đúng **k** phần tử phân biệt.

→ Dùng mảng tần số **count[x]**: đếm số lần xuất hiện của **x** trong đoạn **[0,i]**

* Nếu **count[x] > 0**: giá trị **x** đã xuất hiện trước đó.

* ngược lại: **count[x] == 0** gặp một giá trị mới.

→ Tăng **i** cho đến khi gặp **k** lần **count[x] == 0**.

Minh họa Input 2

Input:

8 3

1 1 2 2 3 3 4 5

A

1	1	2	2	3	3	4	5
0	1	2	3	4	5	6	7

count

0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 0

Minh họa Input 2

Input:

8 3

1 1 2 2 3 3 4 5

• **count[A[i]] = count[1] = 0:**
1 chưa xuất hiện

→ Tăng **unique** = **0 + 1 = 1**

• Tăng **count[A[i]]** lên **1**

A

i								
	↓							
	1	1	2	2	3	3	4	5
	0	1	2	3	4	5	6	7

count

0	0 → 1	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 1

Minh họa Input 2

Input:

8 3
1 1 2 2 3 3 4 5

• **count[A[i]] = count[1] = 1:**
1 đã xuất hiện
→ Giữ nguyên **unique = 1**

• Tăng **count[A[i]]** lên 1

A

i	i								
⋮	↓								
		1	1	2	2	3	3	4	5
		0	1	2	3	4	5	6	7

count

0	1→2	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 1

Minh họa Input 2

Input:

8 3
1 1 2 2 3 3 4 5

• **count[A[i]] = count[2] = 0:**
2 chưa xuất hiện
→ Tăng **unique** = 1 + 1 = 2

• Tăng **count[A[i]]** lên 1

A

1	1	2	2	3	3	4	5
0	1	2	3	4	5	6	7

count

0	2	0→1	0	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 2

Minh họa Input 2

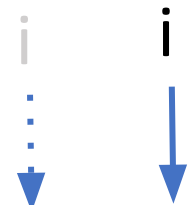
Input:

8 3
1 1 2 2 3 3 4 5

• **count[A[i]] = count[2] = 1:**
2 đã xuất hiện
→ Giữ nguyên **unique = 2**

• Tăng **count[A[i]]** lên 1

A



1	1	2	2	3	3	4	5
0	1	2	3	4	5	6	7

count

0	2	1→2	0	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 2

Minh họa Input 2

Input:

8 3

1 1 2 2 3 3 4 5

- **count[A[i]] = count[3] = 0:**
3 chưa xuất hiện
→ Tăng **unique** = 2 + 1 = 3

- Tăng **count[A[i]]** lên 1

Tại **i = 4**, **unique** = 3: thỏa điều kiện 1
Còn điều kiện 2: là đoạn con tối thiểu

A

1	1	2	2	3	3	4	5
0	1	2	3	4	5	6	7

count

0	2	2	0→1	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 3

Ý tưởng (tt)

Khi **unique == K**, thu hẹp đầu phía bên trái:

1. Gọi **j = 0** là đầu mút bên trái.
2. Nếu **count[A[j]] > 1** : trong đoạn **[j+1, i]** vẫn còn chứa **A[j]**
→ Thu hẹp đoạn thành **[j+1, i]**
3. Tiếp tục thu hẹp cho đến khi không thu hẹp được nữa, tức gặp **count[A[j]] == 1**

Minh họa Input 2

Input:

8 3
1 1 2 2 3 3 4 5

- **count[A[j]] = count[1] = 2**: phía sau vẫn còn chứa A[j]
→ Thu hẹp [j+1,i]

- Giảm **count[A[j]]** đi 1

A

j					i			
	↓				↓			
	1	1	2	2	3	3	4	5
	0	1	2	3	4	5	6	7

count

0	2→1	2	1	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 3

Minh họa Input 2

Input:

8 3
1 1 2 2 3 3 4 5

• **count[A[j]] = count[1] = 1**: phía sau không còn chứa **A[j]**
→ Không thu hẹp nữa.

→ **[j,i]** là đoạn cần tìm.

A

j	j			i			
⋮	↓			↓			
1	1	2	2	3	3	4	5
0	1	2	3	4	5	6	7

count

0	1	2	1	0	0	0	0	0
0	1	2	3	4	5	6	7	...

unique = 3

Các bước giải

Bước 1: Đọc **N**, **K**, mảng **A**,
khởi tạo mảng tần số **count** có $10^5 + 1$ phần tử với giá trị 0.

Bước 2: Khởi tạo **unique** = 0, **i** = 0, **j** = 0

- Nếu **count[A[i]]** == 0: **unique** += 1

Cập nhật mảng tần số: **count[A[i]]** += 1

- Nếu **unique** == **K**, thu hẹp đầu bên trái:
 - * Nếu **count[A[j]]** > 1: **count[A[j]]** -= 1, **j** += 1
 - * ngược lại, không thể thu hẹp nữa: in ra (**j+1**, **i+1**)
- Tăng **i** để tiếp tục kết nạp thêm phần tử mới.

Độ phức tạp: **O(N)**

Mã giả

Mã giả

```
read N, K
read array A
count = [0]*(1000000 + 1) ~ lưu tần số các số trong mảng [j,i]
unique = 0
j = 0
for i = 0 → N-1:
    if count[A[i]] == 0:
        unique += 1
    count[A[i]] += 1
    if unique == K: ~Biên phải i đã gọn nhất
        while True:
            if count[A[j]] > 1:
                count[A[j]] -= 1
                j += 1
            else: ~ A[j] là giá trị duy nhất trong đoạn mảng [j,i]
                print(j + 1, i + 1)
                exit()
print("-1 -1")
```

Thank you