DELEGAME - Lời Giải

Nhóm 1

January 2026

1 Phát biểu bài toán

Cho hai dãy số nguyên: A độ dài n và B độ dài m. Với mỗi giá trị x, ký hiệu $\text{first}(x)$ là **vị trí xuất hiện sớm nhất** của x trong A (đánh số từ 0), và coi $\text{first}(x) = +\infty$ nếu x không xuất hiện trong A.

Yêu cầu: xác định người thắng trong trò chơi **DELEGAME** theo quy tắc sau:

- Nếu tồn tại phần tử x trong B mà x **không xuất hiện** trong A (tức $\text{first}(x) = +\infty$), **Bob thắng**.
- Ngược lại, xét dãy chỉ số $\big(\text{first}(B_1), \text{first}(B_2), \ldots, \text{first}(B_m)\big)$. Nếu dãy này **không giảm** (tức là $\text{first}(B_i) \le \text{first}(B_{i+1})$ với mọi i), **Alice thắng**; nếu có chỉ số vi phạm (tồn tại i sao cho $\text{first}(B_i) > \text{first}(B_{i+1})$), **Bob thắng**.

Diễn giải quan Trực giác: B "tôn trọng" thứ tự xuất hiện sớm nhất trong A thì Alice thắng; chỉ cần một cặp ngược thứ tự (hoặc phần tử không có trong A), Bob thắng.

2 Phân tích và đánh giá phương pháp

2.1 Ý tưởng thuật toán

Thuật toán sử dụng hai bước rất tuyến tính:

1. Duyệt A từ trái sang phải, lưu **vị trí xuất hiện đầu tiên** cho mỗi giá trị x vào bảng băm `first`.
2. Duyệt B từ **phải sang trái** và duy trì một ngưỡng `limit` (ban đầu là n). Với mỗi phần tử x:
   - Nếu x không có trong `first`: đặt `limit` lại $= n$ (coi như chuỗi điều kiện bị *reset*).

- Nếu có $p = \text{first}(x)$:
  - Nếu $p \le$ `limit`: cập nhật `limit` $\leftarrow p$.
  - Nếu $p >$ `limit`: đặt `limit` $= n$ (vi phạm tính không giảm khi nhìn theo chiều trái sang phải).

Sau khi quét xong, nếu `limit` $< n$ nghĩa là phần tử đầu tiên của B "gắn được" vào một chuỗi không giảm hợp lệ của các chỉ số xuất hiện đầu tiên; kết luận **Alice thắng**. Ngược lại, **Bob thắng**.

2.2 Tính đúng đắn (phác thảo)

Bất biến Khi duyệt B từ phải sang trái, biến `limit` luôn giữ giá trị **chỉ số nhỏ nhất có thể** của phần tử kế tiếp về bên trái để chuỗi chỉ số $\text{first}(\cdot)$ còn **không tăng** khi nhìn từ trái sang phải.
Nếu tại phần tử $x = B_i$ có $p = \text{first}(x) \le$ `limit`, ta có thể đặt chỉ số của B_i về p mà vẫn duy trì được tính không giảm cho đoạn $B_i, B_{i+1}, \ldots, B_m$. Ngược lại, khi $p >$ `limit` hoặc x không tồn tại trong A, mọi cố gắng "nối" B_i vào một đoạn hợp lệ ở bên phải đều thất bại, nên cần *reset*.
Ở cuối vòng lặp, điều kiện `limit` $< n$ tương đương với việc tồn tại một đoạn hậu tố hợp lệ của B mà **bao trùm** luôn phần tử đầu tiên, tức toàn bộ B tôn trọng thứ tự xuất hiện đầu tiên trong A. Do đó, tiêu chí in `"Alice"`/`"Bob"` là chính xác.

2.3 Đánh giá

- **Đơn giản, tuyến tính**: mỗi phần tử của A và B được xử lý đúng **một lần**.
- **Bộ nhớ nhỏ**: chỉ lưu một bảng băm các lần xuất hiện đầu tiên.
- **Ổn định với trùng lặp**: việc cố định tại *lần xuất hiện đầu tiên* loại bỏ sự mơ hồ do phần tử lặp lại trong A.

3 Phân tích độ phức tạp

Thời gian: $O(n + m)$ — một lần quét A và một lần quét B.
Không gian: $O(u)$ với $u \le n$ là số lượng giá trị phân biệt trong A (kích thước bảng `first`).

4 Giải thích code (Python)

- Đọc n, m; xây `first[x]` = chỉ số xuất hiện đầu tiên của x trong A.
- Đọc dãy B.

- Khởi tạo `limit`$= n$; duyệt B từ phải sang trái:
  - Nếu x không có trong `first` hoặc `first`$[x] >$ `limit`: `limit`$\leftarrow n$.
  - Ngược lại: `limit`$\leftarrow$ `first`$[x]$.
- In `"Alice"` nếu `limit`$< n$, ngược lại in `"Bob"`.

5 Cài đặt (Python)

```python
import sys

def solve():
    it = iter(sys.stdin.buffer.read().split())
    try:
        n = int(next(it)); m = int(next(it))
    except StopIteration:
        return

    first = {}
    for i in range(n):
        x = int(next(it))
        first.setdefault(x, i)

    b = [int(next(it)) for _ in range(m)]

    limit = n
    for x in reversed(b):
        p = first.get(x)
        if p is None or p > limit:
            limit = n
        else:
            limit = p

    sys.stdout.write("Alice\n" if limit < n else "Bob\n")

if __name__ == "__main__":
    solve()
```