Dễ thấy việc thực hiện các truy vấn theo thứ tự như nào thì không quan trọng.

Và ta có thể đổi lại đề bài một chút thành: Ban đầu cho mảng A có N phần tử hỏi tốn bao nhiêu thao tác trừ đoạn để cho mảng A chuyển thành toàn bộ về 0 với ít thao tác nhất có thể.

Vậy vì hãy cùng kiểm tra test đề bài là:

1 2 1 4

Khi đó dễ thấy các thao tác trừ chúng ta sẽ tìm cách ưu tiên các thao tác có **vùng ảnh hưởng là to nhất** vì khi đó kết quả sẽ luôn là số lượng thao tác ít nhất.

Là ưu tiên thực hiện [1;4] rồi [2;2] và [4;4].

Hãy cùng kiểm tra ví dụ:

4 6 4 5 4

Vậy thì bước này chúng ta sẽ thấy việc giảm cả đoạn xuống 4 đơn vị là một bước đi tối ưu.

Vậy thì mỗi khi chúng ta trừ cả đoạn tất cả các phần tử có giá trị ngang bằng và **không có phần tử nhỏ hơn ở giữa** ví dụ: 4 3 4 thì sẽ không thể bị trừ đi trong thao tác đó.

Tức với mỗi phần tử A_i tạo nên một **đỉnh** mới trong toàn bộ các bước đang bằng nhau thì chúng sẽ cần một thao tác riêng để loại bỏ ví dụ: 1 2 1 4 thì 2 và 4 là đỉnh trong trường hợp này.

Nhưng chúng ta cũng sẽ kiểm tra điều kiện là chúng có thực sự cần một thao tác riêng nữa là bắt buộc phải có phần tử **bằng giá trị** với phần tử đang xét hiện tại và cũng phải thỏa mãn việc ở giữa chúng không có phần tử nào bé hơn cả hai phần tử đang xét hiện tại ví dụ: 4 3 4 là phải thêm thao tác, nhưng 4 5 4 thì không cần thêm thao tác cho số 4 lần 2.

Subtask 1 và 2

Với việc subtask 1 và 2 có số lượng nhỏ của N thì chúng ta sẽ dùng các cấu trúc dữ liệu để kiểm tra điều kiện xem A_i đang xét có phải là:

- Tạo nên một đỉnh mới.
- Không bị chồng lấn trong thao tác cộng của một giá trị tương ứng đứng trước nó.

Vậy để kiểm tra chúng ta có thể sử dụng các cấu trúc dữ liệu quản lý như IT và BIT.

Subtask 3

Với việc subtask 3 là tối đa 2×10^6 nên chúng ta không thể nào dùng được các cấu trúc dữ liệu như IT và BIT vì sẽ không đủ thời gian mà chúng ta sẽ dùng kỹ thuật stack đơn điệu để lưu trữ các phần tử tăng dần nhưng $\leq A_i$ rồi kiểm tra với điều kiện ban đầu.

Solution mẫu