SORTSUB

Gọi $B_{i,j}$ là giá trị của a_j sau i truy vấn đầu tiên.

Ta sẽ chặt nhị phân để tìm đáp án. Để làm được điều này, ta cần kiểm tra xem $B_{q,i} \geq x$ sau mọi truy vấn hay không.

Dễ dàng nhận thấy ta có thể chia các số nguyên trong dãy a thành hai loại: các số nguyên nhỏ hơn x và các số nguyên lớn hơn hoặc bằng x. Hãy mô phỏng lại việc sắp xếp, nhưng thay vì sắp xếp $B_{i,j}$, ta hãy sắp xếp $C_{i,j} = (B_{i,j} \geq x)$. Bây giờ, để kiểm tra xem $B_{i,j} \geq x$ hay không, ta sẽ kiểm tra $C_{i,j}$ mang giá trị $true$ hay $false$.

Chuyện gì sẽ xảy ra với C_i khi ta thực hiện truy vấn $i + 1$? Hãy nhìn vào dãy $C_{i,l_{i+1}}, C_{i,l_{i+1}+1}, ..., C_{i,r_{i+1}}$. Ta sẽ thấy các giá trị $false$ sẽ dịch chuyển sang bên trái, còn các giá trị $true$ sẽ dịch chuyển sang bên phải.

Hãy lưu C_i vào một cây IT (hay còn gọi là Segment Tree). Cây này giúp ta đếm được số giá trị $false$ ở trong một đoạn nào đó. Để thực hiện truy vấn i, ta làm như sau:

- Gọi cnt là số lượng giá trị $false$ trong đoạn $[l_i; r_i]$.
- Đặt giá trị $false$ cho mọi phần tử trong đoạn $[l_i; l_i + cnt - 1]$.
- Đặt giá trị $true$ cho mọi phần tử trong đoạn $[l_i + cnt; r_i]$.

Với sự hỗ trợ của cấu trúc dữ liệu này, ta có thể chặt nhị phân tìm kết quả một cách dễ dàng trong thời gian cho phép.

Độ phức tạp: $O(n.log^2(n))$.