Fixing array

Không mất tính tổng quát của bài toán, ta rời rạc hóa giá trị của mảng a, lúc này giá trị của các phần tử a_i sẽ nằm trong khoảng $[1, k]$ với k là số giá trị khác nhau trong mảng a.

Giả sử ta sẽ chuyển L phần tử về đầu dãy, thì thứ tự chuyển là ta sẽ lấy theo thứ tự giảm dần của các phần tử được chuyển về đầu dãy. Và chuyển R phần tử về cuối dãy, thì thứ tự chuyển là ta sẽ lấy theo thứ tự tăng dần của các phần tử được chuyển về cuối. Và với những tử còn lại không cần chuyển phải có dạng sau:

- Gọi x là phần tử nhỏ nhất trong các phần tử không cần chuyển
- Khi loại bỏ các phần tử phải chuyển đi và giữ nguyên vị trí của các phần tử còn lại thì chúng sẽ có dạng:
$$x, x, x, x+1, x+1, ..., x+2, x+2, ..., x+t, t+t, x+t$$
- giá trị của các phần tử trong L phần tử phải chuyển về đầu dãy không được vượt quá x
- giá trị của các phần tử trong R phần tử phải chuyển về cuối dãy không được nhỏ hơn $x+t$.

Như vậy, bài toàn giờ cần giải quyết là làm sao tìm được các đoạn $[x+1, x+t-1]$ thỏa mãn các phần tử thuộc đoạn này trong dãy a có giá trị không giảm theo thứ tự. Để tìm được các đoạn này ta có thể sử dụng cách sau:

- Lưu vị trí của các phần tử có giá trị là i trong mảng a và một vector $P[i]$, các vị trí này được sắp xếp tăng dần trong $P[i]$.
- Vì khi rời rạc hóa xong thì ta đảm bảo rằng với mọi giá trị $x < k$ luôn tồn tại phần tử có giá trị là $x+1$, do đó các đoạn $[x+1, x+t-1]$ sẽ thỏa mãn $P[i].front() \leq P[i-1].back()$ với $x+1 < i \leq x+t-1$.
- Dễ dàng tìm được các đoạn $[x+1, x+t-1]$ trong thời gian O(n).

Bây giờ, với mỗi đoạn $[x+1, x+t-1]$ ta tìm được ta cần đếm xem có bao nhiêu phần tử có giá trị x có vị trí lớn hơn vị trí đầu tiên của phần tử có giá trị $x+1$ vì những phần tử này sẽ cần phải chuyển về đầu. Tương tự cũng cần phải tìm số lượng các phần tử có giá trị là $x+t$ có vị trí nhỏ hơn vị trí cuối cùng của giá trị $x+t-1$.

- để đếm các phần tử có giá trị bằng x cần chuyển về đầu, ta biết rằng đây là những phẩn tử có vị trí lớn hơn vị trí đầu tiên của phần tử có giá trị $x+1$, do đó chỉ cần sử dụng tìm kiếm nhị phân trong tập $P[x]$ là có thể đếm được L_x (số lượng giá trị x cần chuyển về đầu):
$$L_x = P[x].end() - upper\_bound(P[x].begin(), P[x].end(), P[x+1].front())$$
- tương tự ta đếm số lượng các phần tử có giá trị là $x+t$ có vị trí nhỏ hơn vị trí cuối cùng của phần tử có giá trị $x+t-1$ gọi là R_{x+t}:
$$R_{x+t} = upper\_bound(P[x+t].begin(), P[x+t].end(), P[x+t-1].back()) - P[x+t].begin()$$

Gọi số lượng các phần tử có giá trị trong đoạn $[x, x+t]$ là $S_{[x,x+t]}$ thì số lượng các phần tử cần phải chuyển khi ta chọn đoạn $[x+1, x+t-1]$ sẽ là:

$$d_{[x+1,x+t-1]} = n - S_{[x,x+t]} + L_x + R_{x+t}$$

Kết quả sẽ là $\min(d_{[x+1,x+t-1]})$.

Tuy nhiên có một trường hợp đặt biệt có dạng $x, x, x, ..., x+1, x+1, x+1$. Với trường hợp này, ta thấy rằng với mỗi giá trị x có thể có rất nhiều đoạn dạng này, với mỗi vị trí i có giá trị là x ta luôn lấy tất cả những vị trí nhỏ hơn i có giá trị là x, và các phần tử $x+1$ có vị trí lớn hơn vị trí i cũng sẽ được chọn. Như vậy với mỗi giá trị x ta cần xét thêm trường hợp này như sau:

- Với mỗi giá trị x ta duyệt lần lượt i là số lượng phần tử có giá trị là x đầu tiên không bị chuyển, vị trí của phẩn tử giá trị x thứ i sẽ là $P[x][i-1]$.
- Số lượng phần tử có giá trị $x+1$ không phải chuyển sẽ là các phẩn tử có vị trí lớn hơn $P[x][i-1]$, như vậy số lượng các phần tử này sẽ là $\overline{R}^i_{x+1}$:

$$\overline{R}^i_{x+1} = P[x+1].end() - upper\_bound(P[x+1].begin(), P[x+1].end(), P[x][i-1])$$

- Số lượng phẩn tử không phải chuyển sẽ là $i + \overline{R}^i_{x+1} =>$ số lượng cần chuyển là $n - (i + \overline{R}^i_{x+1})$