Hiểu đề:

Có thể hiểu thế này:

- Nếu như trong dãy x tìm đươc 2 vị trí $i < j$ mà $x[i] > x[j]$ thì ta nói cặp vị trí (i, j) là cặp nghịch thế, để thuận tiên, ta gọi đây là nghịch thế gắn với $\boldsymbol{j}$.
- Như vậy, nếu có k nghịch thế gắn với $\boldsymbol{i}$ thì số được lưu tại vị trí i của mảng x sẽ được gắn một giá trị là $t[x[i]] = k$;

✓ Cho biết dãy số x, hãy thành lập dãy t theo cách trên.

✓ Cho biết dãy số t, hãy tìm dãy x có thể lập ra t theo cách trên.

Tư tưởng giải thuật:

a) Tìm T:

- Ta tưởng tượng rằng có 1 dãy các ô còn trống, ta tiến hành điền các số vào đó để sau cùng thu được dãy x. Dĩ nhiên ta phải điền i vào vị trí j thỏa mãn $x[j] = i$.

✓ Ban đầu ta sẽ điền số 1 vào ô i có $x[i] = 1$. Khi đó $t[1] = i - 1$ vì 1 là số bé nhất nên $i - 1$ ô ở bên trái ô i đều sẽ được điền các số > 1. ==> $t[1]$ =số các ô còn trống nằm bên trái $i; = i - 1$;

✓ Tiếp theo ta điền số 2 vào ô j có $x[j] = 2$. Khi đó $t[2]$ =số các ô còn trống nằm bên trái j, vì các ô này sẽ được điền các số > 2.

...... cứ như vậy điền hết các số khác.

- Như vậy ta sẽ tính được lần lượt các t từ $1..n$, và tại thời điểm tính $t[i]$, ***ta phải biết*** vị trí của i trong mảng x ($x[?] = i$) và biết luôn có bao nhiêu ô còn trống ở trước vị trí đó.

Để giải quyết:

1: Gọi $xx[i]$ là vị trí của i trong x. $->$ $xx[x[i]] = i$;

2: Có rất nhiều cách để tính được số ô còn trống trong khoảng $[1..i]$:

✓ Gọi $a[i]$ là số ô còn trống tính từ $1..i - 1$. Hiển nhiên ban đầu $a[i] = i - 1\ \forall\ i$.

✓ Sau khi điền 1 số vào vị trí i, ta giảm $a[i + 1..n]$ đi 1. Có thể sử dụng các cấu trúc như Segment tree, BinDexTree để làm điều này trong $LogN$.

$->$ $t[i] := a[xx[i]]$; ($t[i]$ =số ô trống bên trái vị trí điền i).

Để cài đặt ngắn gọn, ta không cần lưu lại mảng x, mảng t vì không sử dụng trong các truy vấn. Mảng xx có thể thành lập trong khi đọc dữ liệu mà không cần mảng x, trong khi điền các số vào ô ta có thể in ra các $t[i]$ mà không cần lưu trữ.

b) Tìm X:

- Cũng tưởng tượng như trên. Ta điền lần lượt các số $1..n$ vào các ô, sao cho khi điền i vào vị trí j thì số ô còn trống bên trái ô j $(= t[i])$.

Xét việc điền i vào vị trí j nào đó:

- Nếu gọi mảng a có ý nghĩa như trên, Ta phải tìm j sao cho $a[j] = i$. Do mảng a tăng nên có thể chặt nhị phân để tìm j đơn giản. Độ phức tạp để tìm ra j là $LogN$, sau đó độ phức tạp để cập nhật lại mảng a cũng là $LogN$.

Cụ thể:

Cho dãy nghịch thế $t1, t2, .., tn$, tìm dãy hoán vị nhận t là dãy nghịch thế.

Giả sử ta cần điền vào n ô trống n số để được dãy hoán vị cần tìm. Dễ dàng nhận thấy với mỗi giá trị t_i ta cần tìm ô trống thứ $t_i + 1$ từ trái sang để điền vào số i. Để giải quyết vấn đề trong độ phức tạp cho phép có thể dùng cây BIT hoặc IT:

    - Dùng cây BIT. Chặt nhị phân x từ 1 đến n, dùng BIT tính số ô đã có số rồi tính số ô trống từ 1 đến x, nếu số ô trống $\geq t[i] + 1$ thì giảm x, ngược lại tăng x. Sau đó cập nhật ô được chọn. Độ phức tạp $O(n \times \log^2 n)$.