Brute Force.

Generate all permutations & store in a vector, sort the vector to get lexigraphical order.
Find the current sequence & return the next sequence.

Time Complexity : O(N!)
space complexity : O(N!)

Optimised Solution.

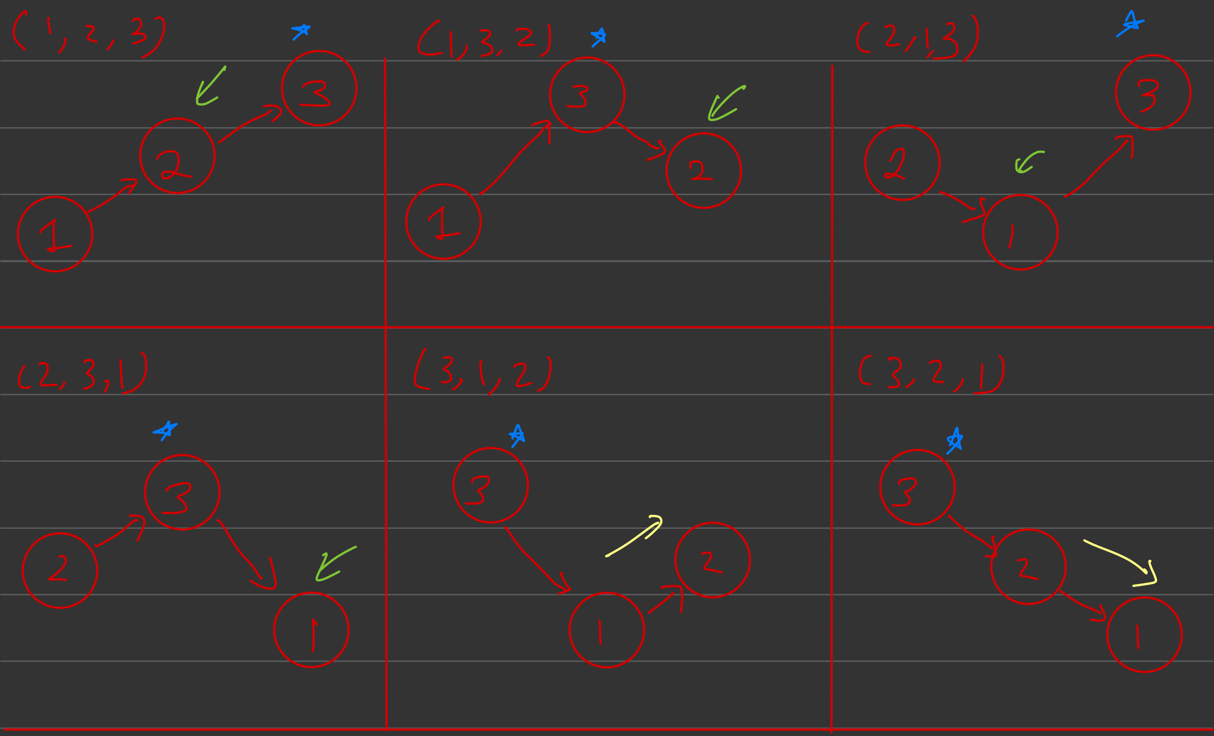
- Lexicographical Order for 1, 2, 3

1, 2, 3
1, 3, 2
2, 1, 3
2, 3, 1
3, 1, 2
3, 2, 1

$\downarrow$ Ascending order

$\downarrow$ Ascending order if first value is same

- If we plot all the values, we can see a pattern being followed.



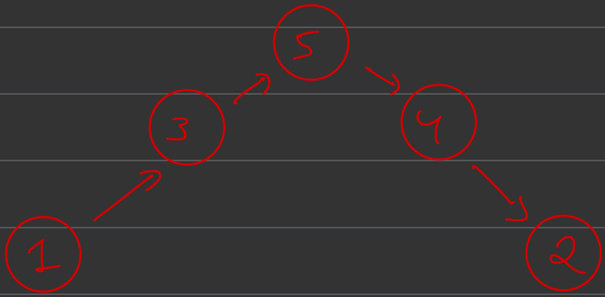
Here, we can see every sequence has an increasing order, might or might not be followed by a decreasing order.

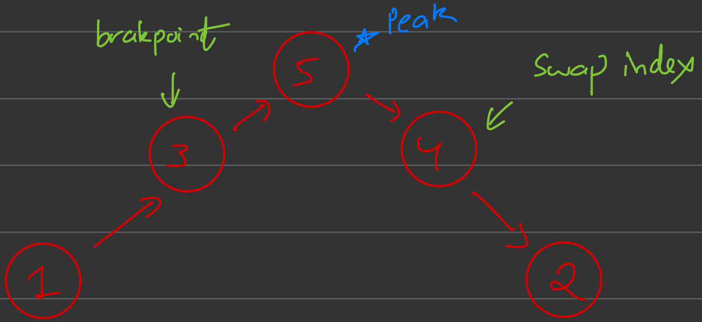
Only thing changing is the nodes on left & right of the peak

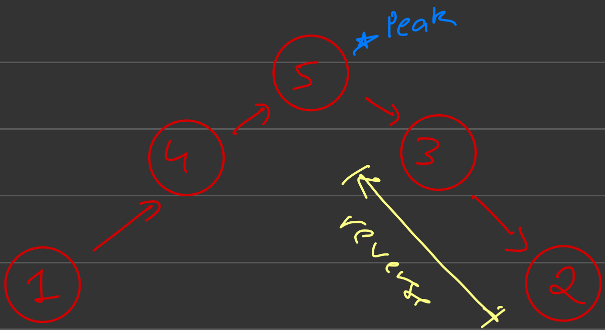
∴ We need to get the peak.

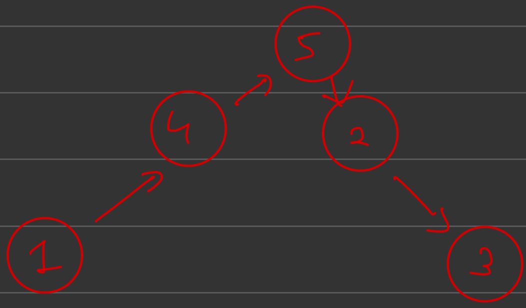
If, there's an element on the left of peak, it goes on the right side
and elements after the peak are reversed (to maintain lexicographical order)

Input :









Output

Algorithm :

1) Traverse from back to find peak
   if(arr[i] < arr[i+1]) peak = i

   If peak is not found, that means it is the last sequence, hence reverse the whole array to get the first sequence
   (321 → 123)

2) Get the breakpoint index
   break_pt_idx = peak - 1

3) Get the swap_idx, traverse the array from back to find an element just higher than a[break_pt_idx]

```
if( arr[i] > arr[break_pt_idx])
    swap_idx = i
  swap(arr[break_pt_idx], arr[swap_idx]);
```

4) Reverse the array from peak + 1 to n-1