

ON A MISSION TO MAKE YOU
LOVE DSA

STRIVER'S SDE SHEET

ARRAY

PB-

Next Permutation

INTERVIEW SIMULATION

MADE BY -



MAYANK SINGH

Statement

- A permutation of an array of integers is an arrangement of its members into a sequence or linear order.
- For example, for `arr = [1,2,3]`, the following are considered permutations of `arr`: `[1,2,3]`, `[1,3,2]`, `[3,1,2]`, `[2,3,1]`.
- The next permutation of an array of integers is the next lexicographically greater permutation of its integer
- If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

i/p

`nums = [1,2,3]`

o/p

`[1,3,2]`

i/p

`nums = [3,2,1]`

o/p

`[1,2,3]`

Brute Force

Bhaiyaa: Ruyank, what is asked in **Problem Statement** ?

Ruyank: Given an integer array `nums[]` we need to re-arrange them in such a way that this new arrangement should be the **Next Permutation** or simply put it should be **just greater** than prev. one

Bhaiyaa: What is the **naive approach** you can think of ?

Ruyank: Bhaiyaa we can try generating all sort of permutations & have a global array which contains our answer & while generating those permutation keep checking which is the Next Permutation

Bhaiyaa: For n elements we would have **$n!$ total permutations** so your **T.C. would be $O(n!)$** & you are having a global array so **S.C. is $O(n)$** , let's come up with something better

1784 1847 1874 8714

Ruyank, out of above 4 combinations which is next(just greater) permutation of **1748** ?

Ruyank: **1784** is next permutation of 1748 because **rightmost** digits have lower weightage in number system so if we want just larger number(not very large) than 1748 then we would start from right and see which number is just greater than 1748 & 1749 is that no

1748

$$1 \times 1000 + 7 \times 100 + 4 \times 10 + 8 \times 1$$

rightmost digit has low weightage

Bhaiyaa: Awesome, so to find next permutation we will be considering **rightmost digits first**

Observations

Bhaiyaa: Now back to problem statement, let's solve few test cases

Give me Next Permutation of 3876

Ruyank: It is 6378

Bhaiyaa: Correct, but how you got it ?

Ruyank: Obviously we want to rearrange the digits of 3876 and as we saw earlier we will start from right side & firstly while iterating from right (last index) I had following numbers

6	These are the numbers, we are getting when we iterate from rightmost index of 3876
76	
876	
3876	


If we look at first 3 numbers there is no way we can rearrange those values to get a smaller number, only while iterating from right I encountered a smaller number 3 which is last value(3876) above, so now in place of 3 I can put some value which is greater than 3 thus we can get just larger number

Bhaiyaa: So from right you first want the index which would be replaced, so if you carefully observe then 6, 76 or 876 in these values the left digit is always greater than right but in 3876 when you encounter 3 the condition becomes like $\text{left digit}(3) < \text{right}(8)$ and this left digit is your index value which would be replaced

Let's call it as targetIdx

Observations

Bhaiyaa: I can also observe 1 pattern while searching our
targetIdx


3 8 7 6

All the values after `nums[targetIdx]` will be in **desc. sorted** order but Ruyank, why is it so ?

Ruyank: It is because from right we iterated till **left**
value(nums[i-1]) is greater than right(nums[i]) & the 1st
value which violates this rule (or simply put ,
terminates the **while(nums[i-1] >= nums[i])**) is our
targetIdx and from here we can see that till we get our
targetIdx all the values **right to it** (here they are 8 7
6) **will be desc. sorted**

Bhaiyaa: `while(nums[i-1] >= nums[i])`
 i--
 targetIdx = i-1 // when while loop breaks i-1 is our
 // targetIdx

Observations

Bhaiyaa: Ruyank, so `targetIdx` is the index that would be replaced but from whom you would replace it ?

Ruyank: Bhaiyaa, we want just larger number so if I see the below number

3 8 7 6
targetIdx ↑

All the digits right of 3(targetIdx) are desc. sorted so I'll iterate from rightmost idx to get the first value(more precisely index as we need to swap the values later) which is greater than 3 & it would also be the **smallest value greater than 3** (targetIdx), let's call that idx as `greaterIdx`

3 8 7 6
targetIdx ↑ ↑ greaterIdx

Now `swap(nums[targetIdx], nums[greaterIdx])`, so we will get the following configuration

6 8 7 3

Bhaiyaa: Is this your final step, are we done ?

Ruyank: No bhaiyaa, one more step is remaining

Observations

Bhaiyaa: And what's that step ?

Ruyank: Bhaiyaa, we wanted just larger number or permutation so we replaced 3 with some greater value which gave us this configuration

`6xxx > 3xxx`

But still 6xxx may not be just larger(`it can very large`) than 3xxx as in previous example after swapping 3 & 6 we had

6 8 7 3, so after 6 we have 873 which can be `minimized` to `378` so finally we can have `6378`, now this is next permutation of `3876`

Bhaiyaa: After your targetIdx (the value is 6) you wanted to minimize the conf. in order to attain just larger combination, how would you do that ?

Ruyank: Bhaiyaa, we have 873 right to 6 so we can `sort()` them to `378` thus we have `6378`

Bhaiyaa: Can't you see that 873, if we just `reverse` it to `378` we still get what we desired thus finally `6378`

And this `reverse()` is also possible only because of that `while loop` we saw earlier which made this configuration such that right to targetIdx all values will be `desc. sorted`, so to minimize them it's better to just reverse them instead of sorting

Algorithm

Bhaiyaa: Ruyank, write down all the steps we discussed, so far !

Ruyank:

- Find the targetIdx

```
i = n - 1
while(nums[i-1] >= nums[i])
    i--
targetIdx = i-1
```

- Find the index which will replace this targetIdx

```
i = n - 1
while(nums[i] <= nums[targetIdx])
    i--
greaterIdx = i
```

- swap(nums[targetIdx], nums[greaterIdx])
- reverse all the values right to targetIdx

Bhaiyaa: What is the time & space complexity for the solution

Ruyank: Space required is obviously $o(1)$ as we aren't using anything

For T.C. we have 1 while loop($o(n)$) (to get targetIdx) + another while loop($o(n)$) (to get greaterIdx) + swapping ($o(1)$) + reverse($o(n)$)

that is $o(3n)$ or $o(n)$

S.C. = $o(1)$

T.C. = $o(n)$

Ruyank: Bhaiyaa, why don't you provide full solution these days as you used to do earlier ?

Bhaiyaa: You are here for building your intuition or just copy pasting my solution ?

Ruyank: Obviously to build intuitions but what if I get wrong answer (WA) or other errors while submitting, they will waste my time !

Bhaiyaa: Yaa, its true that any error while submitting hurts, but the more you get those errors the better your debugging skills would get

And believe me, I really want that you all should get those errors because only when you encounter those errors, you would really get to know when those edge cases occur & how to overcome them

And don't worry, if I see some problem whose solution is not understandable on Leetcode' Discuss or there official solution, I'll make sure to add a slide

BTW, I told you almost 90% of approach but still there may be some cases that I willingly missed so as you can use you mental muscles