33. Search in Rotated Sorted Array

Solved

Medium | Topics | Companies

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`, *or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

```
Input: nums = [1], target = 0
Output: -1
```

Constraints:

- `1 <= nums.length <= 5000`

00:10:25 0 Premium

Code

Java Auto

```java
class Solution {
    public int search(int[] nums, int target) {
      int s=0;int e=nums.length-1;
      while(s<=e){
        int mid=(s+e)/2;
      if(target == nums[mid])   return mid;
       else if(nums[s] <= nums[mid]){
            if((target >= nums[s]) && (target < nums[mid])){
                e=mid-1;
            }
            else{
                s=mid+1;
            }
       }
       else{
            if((target > nums[mid]) && (target <=nums[e])) {
                s=mid+1;
            }
            else{
                e=mid-1;
            }
       }

      }
      return -1;
    }

    }
```

Saved

Ln 25, Col 17

Search in a Rotated Sorted array

We have to solved it in time complexity less than linear O(N). So Using modified Binary Search we will solved it in O(Logn)

```
while(L<=h)
  m=(l+h)/2
  If(target == a[m])  return m
  else If(a[L] <= a[mid])
       If( target >= a[L] && target < a[mid])
            L=mid-1
       else
            h=mid+1
else
       If (target > a[m] && target <= a[h]);
            L=mid+1
       else
            h=m-1
```

Problem List

Run Submit

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted

Azan imtiaz submitted at Jul 23, 2024 10:49

Editorial Solution

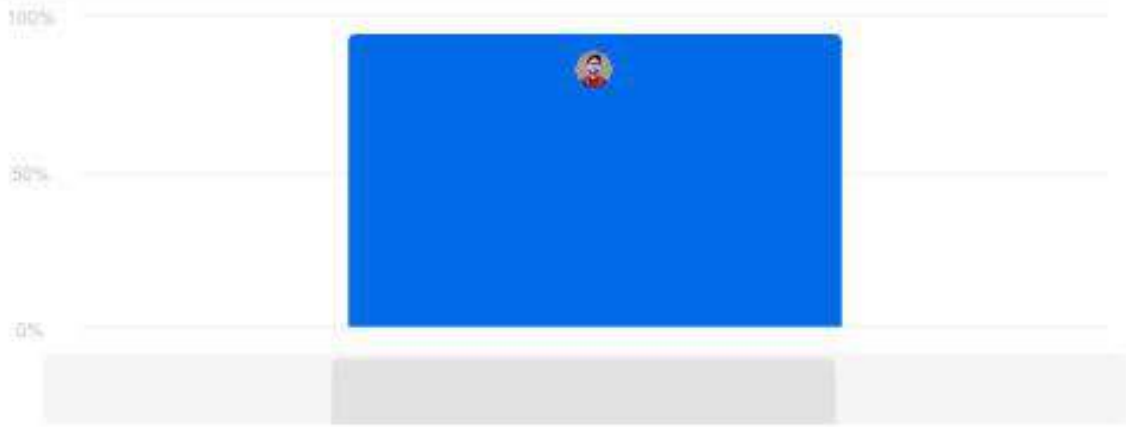
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

41.88 MB Beats 58.15%



Code | Java

```
class Solution {
    public int search(int[] nums, int target) {
      int s=0;int e=nums.length-1;
      while(s<=e){
        int mid=(s+e)/2;
      if(target == nums[mid])   return mid;
       else if(nums[s] <= nums[mid]){
            if((target >= nums[s]) && (target < nums[mid])){
```

View more

Description | Accepted | Editorial | Solutions | Submissions

31. Next Permutation

Solved

Medium | Topics | Companies

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 all the permutations of `arr`: `[1,2,3]`, `[1,3,2]`, `[2, 1, 3]`, `[2, 3, 1]`, `[3,1,2]`, `[3,2,1]`.

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

- For example, the next permutation of `arr = [1,2,3]` is `[1,3,2]`.
- Similarly, the next permutation of `arr = [2,3,1]` is `[3,1,2]`.
- While the next permutation of `arr = [3,2,1]` is `[1,2,3]` because `[3,2,1]` does not have a lexicographical larger rearrangement.

Given an array of integers `nums`, *find the next permutation of* `nums`.

The replacement must be **in place** and use only constant extra memory.

Example 1:

```
Input: nums = [1,2,3]
Output: [1,3,2]
```

Example 2:

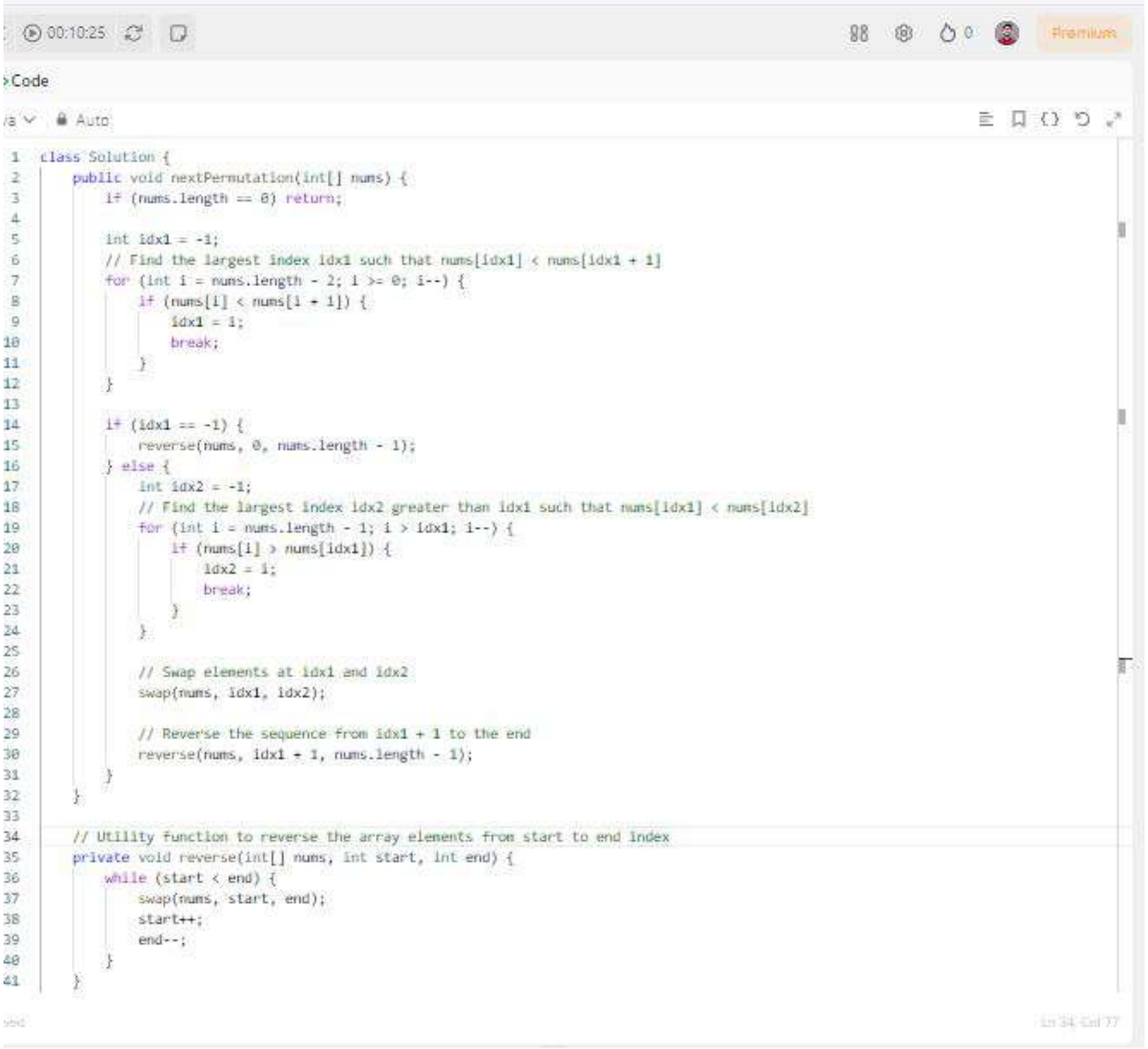
```
Input: nums = [3,2,1]
Output: [1,2,3]
```

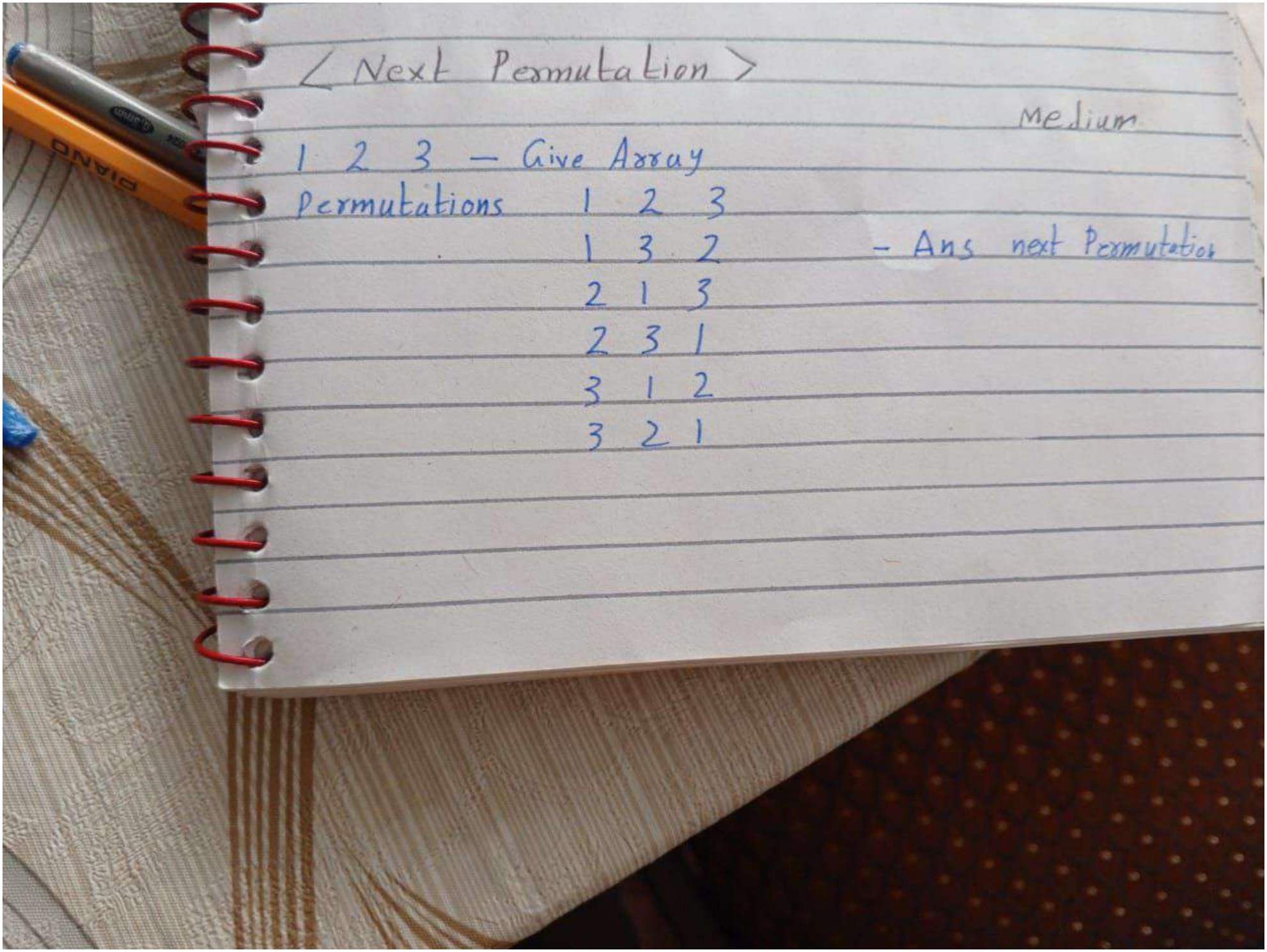
Example 3:

```
Input: nums = [1,1,5]
Output: [1,5,1]
```

Constraints:

18.4K | 194


00:10:25
Premium
Code
Auto
class Solution {
public void nextPermutation(int[] nums) {
if (nums.length == 0) return;
int idx1 = -1;
// Find the largest index idx1 such that nums[idx1] < nums[idx1 + 1]
for (int i = nums.length - 2; i >= 0; i--) {
if (nums[i] < nums[i + 1]) {
idx1 = i;
break;
}
}
if (idx1 == -1) {
reverse(nums, 0, nums.length - 1);
} else {
int idx2 = -1;
// Find the largest index idx2 greater than idx1 such that nums[idx1] < nums[idx2]
for (int i = nums.length - 1; i > idx1; i--) {
if (nums[i] > nums[idx1]) {
idx2 = i;
break;
}
}
// Swap elements at idx1 and idx2
swap(nums, idx1, idx2);
// Reverse the sequence from idx1 + 1 to the end
reverse(nums, idx1 + 1, nums.length - 1);
}
}
// Utility function to reverse the array elements from start to end index
private void reverse(int[] nums, int start, int end) {
while (start < end) {
swap(nums, start, end);
start++;
end--;
}
}


< Next Permutation >
Medium
1 2 3 - Give Array
Permutations
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
- Ans next Permutation

Approach

① Step 1
Find idx1 R → L i = length - 2 i >= 0
arr[i] < arr[i+1]
idx1 = i break;

② Step 2 IF(idx1 == -1) reverse and return array

③ Step 3
Find idx2 R → L i = length - 1 i > idx1
arr[i] > arr[idx1]
idx2 = i; break;

④ Step 4
Swap arr[idx1] and arr[idx2]

⑤ Step 5
Reverse(arr, idx + 1, arr.length - 1)

Problem List

Run Submit

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted

Azan imtiaz submitted at Jul 23, 2024 10:12

Editorial Solution

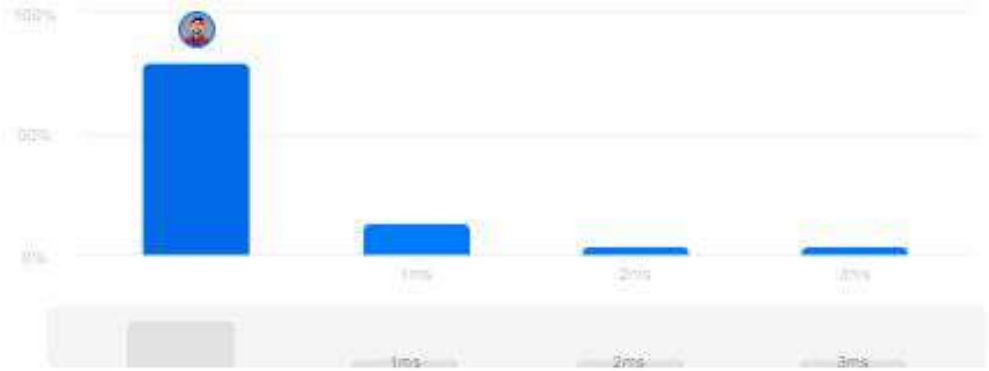
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

42.84 MB | Beats 67.44%



Code | Java

```java
class Solution {
    public void nextPermutation(int[] nums) {
        if (nums.length == 0) return;

        int idx1 = -1;
        // Find the largest index idx1 such that nums[idx1] < nums[idx1 + 1]
        for (int i = nums.length - 2; i >= 0; i--) {
            if (nums[i] < nums[i + 1]) {
```

View more

More challenges

- 46. Permutations
- 47. Permutations II
- 60. Permutation Sequence

Write your notes here