Next Permutation

solved by	Senan
Platform	GeeksForGeeks
difficulty	Medium
tags	Logic
language	C++
solved on	@19/11/2024
link	https://www.geeksforgeeks.org/problems/next-permutation5226/1
Completion	☑

Intuition

To find the next lexicographical permutation of a sequence, the goal is to identify the smallest possible rearrangement of the array that is greater than the current arrangement. This requires finding the "turning point" where the order needs to be adjusted, followed by a swap and reversal to ensure the smallest larger permutation.

Approach

1. **Identify the first decreasing element:** Traverse the array from the end and find the first index (`peak - 1`) where `arr[peak - 1] < arr[peak]`. This is the point where the next permutation can be constructed. If no such point exists, the array is in descending order, and the next permutation is the reverse (smallest permutation).
2. **Find the right element to swap:** From the end of the array, find the smallest element larger than `arr[peak - 1]` and swap it with `arr[peak - 1]`. This ensures that the next permutation is greater.
3. **Reverse the suffix:** Reverse the portion of the array from index `peak` to the end. This guarantees the smallest possible permutation for the rearranged portion, maintaining lexicographical order.

Complexity

Time Complexity:

- Finding the `peak`: **O(n)**, where **n** is the size of the array.
- Finding the element to swap: **O(n)**.
- Reversing the suffix: **O(n)**.

Total: O(n).

Space Complexity:

- **O(1)**, as the solution modifies the array in place without using extra space.

Code

```
class Solution {
public:
    void nextPermutation(vector<int>& arr) {
```

```
        if (arr.size() <= 1) return;

        int peak = -1;
        for (int i = arr.size() - 1; i > 0; i--) {
            if (arr[i - 1] < arr[i]) {
                peak = i;
                break;
            }
        }

        if (peak == -1) {
            reverse(arr.begin(), arr.end());
            return;
        }

        int toSwap = -1;
        for (int i = arr.size() - 1; i >= peak; i--) {
            if (arr[i] > arr[peak - 1]) {
                toSwap = i;
                break;
            }
        }

        swap(arr[peak - 1], arr[toSwap]);
        reverse(arr.begin() + peak, arr.end());
    }
};
```