Given an array arr[] of length n. Find all possible unique permutations of the array in sorted order. A sequence A is greater than sequence B if there is an index i for which Aj = Bj for all j<i and Ai > Bi.

Example 1:

Input:
n = 3
arr[] = {1, 2, 1}
Output:
1 1 2
1 2 1
2 1 1
Explanation:
These are the only possible unique permutations for the given array.

Example 2:

Input:
n = 2
arr[] = {4, 5}
Output:
Only possible 2 unique permutations are

4 5
5 4

Your Task:

You don't need to read input or print anything. You only need to complete the function uniquePerms() that takes an integer n, and an array arr of size n

as input and returns a sorted list of lists containing all unique permutations of the array.

Expected Time Complexity: O(n*n!)

Expected Auxilliary Space: O(n*n!)

Constraints:

1 ≤ n ≤ 9

Solution:

```
//User function Template for Java

class Solution {
    public static ArrayList<ArrayList<Integer>> uniquePerms(ArrayList<Integer> arr, int n) {
        ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
        Collections.sort(arr);
        ans.add(new ArrayList<>(arr));  // Add the first permutation

        while (nextPermutation(arr)) {
            ans.add(new ArrayList<>(arr));  // Add new permutation
        }

        return ans;
    }

    // Helper function to find the next permutation
    private static boolean nextPermutation(ArrayList<Integer> arr) {
        int i = arr.size() - 2;

        // Find the first decreasing element
        while (i >= 0 && arr.get(i) >= arr.get(i + 1)) {
            i--;
        }

        if (i < 0) {
            return false;  // No more permutations
        }

        int j = arr.size() - 1;
```

```
        // Find the element just larger than arr[i]
        while (arr.get(j) <= arr.get(i)) {
            j--;
        }

        // Swap elements at i and j
        Collections.swap(arr, i, j);

        // Reverse the sequence from i+1 to end to get the next smallest lexicographic
permutation
        reverse(arr, i + 1, arr.size() - 1);

        return true;
    }

    // Helper function to reverse a sublist
    private static void reverse(ArrayList<Integer> arr, int start, int end) {
        while (start < end) {
            Collections.swap(arr, start, end);
            start++;
            end--;
        }
    }
}
```