Unique Permutations of Array LeetCode

This C++ program generates all permutations of a given array while avoiding duplicate permutations using two different backtracking approaches.

Approach 1: Function to find all permutations of the array using Backtracking

- The **findPermutations** function generates all permutations of the input array **nums** using recursive backtracking while avoiding duplicate permutations.
- It checks if the current permutation is not already present in the **ans** vector before adding it. This is achieved using the **find** function from the **<algorithm>** library.
- The rest of the approach is the same as in the previous explanation.
- **Time Complexity: O(n!), similar to the basic backtracking approach, but with the overhead of checking for duplicates.**
- **Space Complexity: O(n), similar to the basic backtracking approach.**

Approach 2: Function to find all permutations of the array using Backtracking, excluding duplicates

- The **findPermutationsAlternative** function is an alternative implementation of backtracking to generate permutations while excluding duplicate permutations.
- Before starting the backtracking process, the input array **nums** is sorted.
- At each step, it checks if the current element is the same as the previous one. If yes, it skips that iteration to avoid generating duplicate permutations.
- The rest of the approach is similar to the previous backtracking approach.
- **Time Complexity: O(n!) with some optimization due to duplicate exclusion, similar to the basic backtracking approach.**
- **Space Complexity: O(n), similar to the basic backtracking approach.**