

Permutations II

This problem is similar to the basic permutations problem but requires handling duplicates to return unique permutation.

Key Idea:

- Sort the array first so duplicates are adjacent.
- During backtracking, skip duplicates by checking:
 - If $\text{nums}[i] == \text{nums}[i-1]$ and $\text{nums}[i-1]$ is not used in current recursion level, skip.

Algorithm:

- Sort nums .
- Use a $\text{used}[]$ boolean array to track elements in current permutation.
- For each index i :
 - Skip if $\text{used}[i] == \text{true}$.
 - Skip if $i > 0$ && $\text{nums}[i] == \text{nums}[i-1]$ && $! \text{used}[i-1]$ (avoid duplicates)
- Build permutations recursively.

Complexity:

- Time: $O(n \times n!)$ worst case ($n!$ permutations, each of length n)
- Space: $O(n)$ recursion + $O(n)$ used array