# 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 nums[i] == nums[i-1] and nums[i-1] is not used in current recursion level, skip.

## Algorithm:

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

## Complexity:

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