Problem Description: Subsets II

Problem Statement

The problem "Subsets II" involves finding all possible subsets (the power set) of a given integer array nums which may contain duplicates. The solution set must not include any duplicate subsets, and the result can be returned in any order.

Example 1:

Input: nums = [1, 2, 2]

Output: [[], [1], [1, 2], [1, 2, 2], [2], [2, 2]]

Example 2:

Input: nums = [0]

Output: [[], [0]]

Constraints

- The length of nums is between 1 and 10 (inclusive).
- Each element of nums is an integer between -10 and 10 (inclusive).

Solution Explanation

The solution is implemented using backtracking to generate all possible subsets while ensuring that duplicate subsets are avoided. Here's a step-by-step explanation of the approach:

1. **Sorting the Array:**
   - The input array nums is sorted. Sorting helps to easily skip over duplicates when generating subsets.

2. **Backtracking Function:**
   - A helper function backtrack is defined, which takes two parameters: start (the starting index for generating subsets) and path (the current subset being constructed).
   - The function begins by adding the current subset path to the result list.
   - It then iterates over the elements of nums starting from the start index.

3. **Skipping Duplicates:**
   - Within the loop, a condition checks if the current element is a duplicate of the previous element. If so, the loop continues to the next iteration without including the duplicate element in the subset.
   - If the element is not a duplicate, it is added to the current subset path, and the backtrack function is called recursively with the next starting index (i + 1).

4. **Backtracking:**
   - After the recursive call, the last element added to path is removed (backtracked), allowing for the generation of new subsets without the last element.

5. **Returning the Result:**
   - After the backtracking function completes, the result list contains all unique subsets of nums, which is then returned.