68. Permutations II

Given a collection of numbers, nums, that might contain duplicates, return *all possible unique permutations in any order.*

Example 1:

Input: nums = [1,1,2]

Output:

[[1,1,2],

[1,2,1],

[2,1,1]]

PROGRAM:-

```
def permuteUnique(nums):
    def backtrack(path, used):
        if len(path) == len(nums):
            result.append(path[:])
            return
        for i in range(len(nums)):
            if used[i]:
                continue
            if i > 0 and nums[i] == nums[i - 1] and not used[i - 1]:
                continue
            used[i] = True
            path.append(nums[i])
            backtrack(path, used)
            path.pop()
            used[i] = False

    nums.sort()  # Sort the array to handle duplicates
    result = []
    used = [False] * len(nums)
    backtrack([], used)
    return result

# Example usage and output
```

nums1 = [1, 1, 2]

print(permuteUnique(nums1)) # Output: [[1, 1, 2], [1, 2, 1], [2, 1, 1]]

nums2 = [1, 2, 3]

print(permuteUnique(nums2)) # Output: [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]

OUTPUT:-

```
[[1, 1, 2], [1, 2, 1], [2, 1, 1]]
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]

=== Code Execution Successful ===
```

TIME COMPLEXITY:-O(n!)