PROGRAM 69.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)):
            # Skip used elements and duplicate elements
            if used[i] or (i > 0 and nums[i] == nums[i - 1] and not used[i - 1]):
                continue
            # Mark this element as used
            used[i] = True
            path.append(nums[i])
            backtrack(path, used)
            # Backtrack, remove the element from the current path
            path.pop()
            used[i] = False

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

# Example usage
nums = [1, 1, 2]
print(permuteUnique(nums))
```

Output:

```
"C:\Program Files\Python312\python.exe" "C:\Work Space\DAA\DAA COADS.PYTHON\program 69.py"
[[1, 1, 2], [1, 2, 1], [2, 1, 1]]

Process finished with exit code 0
```

Time complexity:
O(n.n!)