68. Permutations II

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

Code:

```
def permuteUnique(nums):
    nums.sort()
    n = len(nums)
    result = []
    used = [False] * n

    def backtrack(current):
        if len(current) == n:
            result.append(current[:])
            return

        for i in range(n):
            if used[i] or (i > 0 and nums[i] == nums[i - 1] and not used[i - 1])
                continue

            used[i] = True
            current.append(nums[i])
            backtrack(current)
            current.pop()
            used[i] = False

    backtrack([])
    return result
nums = [1, 1, 2]
print(permuteUnique(nums))
```

Output:

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

Time Complexity:

- T(n)= O(n)