47. Permutations II

Created	@July 20, 2020 12:01 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/permutations-ii/
Importance	*****
Tag	Backtrack
Video	

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]]
```

Example 2:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Constraints:

- `1 <= nums.length <= 8`
- `10 <= nums[i] <= 10`

Solution

```python
class Solution:
    def permuteUnique(self, nums: List[int]) -> List[List[int]]:
        res = []
        if not nums:
            return res

        visited = [False] * len(nums)
        nums.sort()
        self.backtrack(nums, [], res, visited)
        return res

    def backtrack(self, nums, subset, res, visited):
        if len(subset) == len(nums):
            res.append(list(subset))
            return

        for i in range(len(nums)):
            if visited[i]:
                continue

            if i > 0 and nums[i] == nums[i - 1] and not visited[i - 1]:
                continue

            subset.append(nums[i])
            visited[i] = True

            self.backtrack(nums, subset, res, visited)

            visited[i] = False
            subset.pop()
```

```java
List<List<Integer>> res = new LinkedList<>();
LinkedList<Integer> track = new LinkedList<>();
boolean[] used;

public List<List<Integer>> permuteUnique(int[] nums) {
    // 先排序，让相同的元素靠在一起
    Arrays.sort(nums);
    used = new boolean[nums.length];
    backtrack(nums);
    return res;
}

void backtrack(int[] nums) {
    if (track.size() == nums.length) {
        res.add(new LinkedList(track));
        return;
    }

    for (int i = 0; i < nums.length; i++) {
```

```
        if (used[i]) {
            continue;
        }
        // 新添加的剪枝逻辑，固定相同的元素在排列中的相对位置
        if (i > 0 && nums[i] == nums[i - 1] && !used[i - 1]) {
            continue;
        }
        track.add(nums[i]);
        used[i] = true;
        backtrack(nums);
        track.removeLast();
        used[i] = false;
    }
}
```