

# 047. Permutations II

## 047 Permutations II

- BackTracking+array

### Description

Given a collection of numbers that might contain duplicates, return all possible unique permutations.

For example,

`[1,1,2]` have the following unique permutations:

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

### 1. Thought line

### 2. BackTracking+array

```
1 class Solution {
2 private:
3     void backTracking_fct(vector<vector<int>>& result, vector<int>& nums, vector<bool>& flag,
4                           vector<int>& temp){
5         if (temp.size()==nums.size()){
6             result.push_back(temp);
7             return;
8         }
9
10        for (int i = 0; i<=flag.size()-1; ++i){
11            if (!flag[i]){
12                temp.push_back(nums[i]);
13                flag[i] = true;
14                backTracking_fct(result, nums, flag, temp);
15                flag[i] = false;
16                temp.pop_back();
17                while(i+1<=flag.size()-1 && nums[i]==nums[i+1])
18                    ++i;
19            }
20        }
21    }
22
23
24 public:
25     vector<vector<int>> permuteUnique(vector<int>& nums) {
26         vector<vector<int>> result;
27         vector<bool> flag(nums.size(),false);
28         vector<int> temp;
29         sort(nums.begin(), nums.end());
30         backTracking_fct(result, nums, flag, temp);
31         return result;
32     }
33 };
```