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

```
class Solution {
private:
    void backTracking_fct(vector<vector<int>>& result, vector<int>& nums, vector<bool>& flag,
                     vector<int>& temp){
        if (temp.size()==nums.size()){
            result.push_back(temp);
            return;
        }

        for (int i = 0; i<=flag.size()-1; ++i){
            if (!flag[i]){
                temp.push_back(nums[i]);
                flag[i] = true;
                backTracking_fct(result, nums, flag, temp);
                flag[i] = false;
                temp.pop_back();
                while(i+1<=flag.size()-1 && nums[i]==nums[i+1])
                    ++i;
            }
        }
    }


public:
    vector<vector<int>> permuteUnique(vector<int>& nums) {
        vector<vector<int>> result;
        vector<bool> flag(nums.size(),false);
        vector<int> temp;
        sort(nums.begin(), nums.end());
        backTracking_fct(result, nums, flag, temp);
        return result;
    }
};
```