# 题目

给定一个可包含重复数字的整数集合 nums ，按任意顺序 返回它所有不重复的全排列。

示例 1：

输入：nums = [1,1,2]

输出：

[[1,1,2],

[1,2,1],

[2,1,1]]

示例 2：

输入：nums = [1,2,3]

输出：[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

提示：

1 <= nums.length <= 8

-10 <= nums[i] <= 10

# 分析

## 方法一：回溯法

这个问题可以使用回溯法来解决。由于输入数组可能包含重复数字，为了避免生成重复的排列，我们可以先对数组进行排序，然后在回溯的过程中，对于每个位置的数字，如果它和前一个数字相同，并且前一个数字还未被使用过，则跳过该数字。

具体步骤如下：

1、对输入数组进行排序，以便相同的数字相邻。

2、使用一个标记数组来记录每个数字是否被使用过。

3、定义一个回溯函数backtrack，用来生成排列。

4、在回溯函数中，遍历数组中的每个数字，如果该数字未被使用过，则将其添加到当前排列中，并标记为已使用；然后递归调用回溯函数；最后在递归结束后，将该数字重新标记为未使用，以便下一次回溯时使用。

下面是具体的实现：

class Solution {

public:

vector<vector<int>> permuteUnique(vector<int>& nums) {

vector<vector<int>> result;

vector<int> path;

vector<bool> used(nums.size(), false);

sort(nums.begin(), nums.end());

backtrack(nums, result, path, used);

return result;

}

private:

void backtrack(vector<int>& nums, vector<vector<int>>& result, vector<int>& path, vector<bool>& used) {

if (path.size() == nums.size()) {

result.push\_back(path);

return;

}

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

if (used[i]) {

continue;

}

if (i > 0 && nums[i] == nums[i - 1] && !used[i - 1]) {

continue;

}

path.push\_back(nums[i]);

used[i] = true;

backtrack(nums, result, path, used);

path.pop\_back();

used[i] = false;

}

}

};

在这个实现中，我们首先对输入数组进行排序，然后定义了一个回溯函数 `backtrack`，用来生成排列。在回溯函数中，我们遍历数组中的每个数字，如果该数字未被使用过，则将其添加到当前排列中，并标记为已使用；然后递归调用回溯函数；最后在递归结束后，将该数字重新标记为未使用，以便下一次回溯时使用。同时，为了避免生成重复的排列，我们在遍历数组时，对于相同的数字，如果前一个数字还未被使用过，则跳过该数字。