1. Permutations II

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

**Example:**

Input: [1,1,2]  
Output:  
[  
 [1,1,2],  
 [1,2,1],  
 [2,1,1]  
]

**解**

先排序，然后判断nums[i-1] == nums[i]是否成立。例如对于[1, 1, 2, 3]

1(0)已经在枚举序列时，1(1)不会再接在后面。所有序列中1(1)出现在1(0)前面

class Solution {  
public:  
 vector<vector<int>> permuteUnique(vector<int>& nums) {  
 vector<vector<int> > ans;  
 vector<int>tmp;  
 sort(nums.begin(), nums.end());  
 bool flag[nums.size()] = {false};  
 dfs(nums, ans, tmp, flag);  
 return ans;  
 }  
 void dfs(vector<int>& nums, vector<vector<int> >&ans, vector<int> &tmp, bool flag[]){  
 if(tmp.size() == nums.size()){  
 ans.push\_back(tmp);  
 return;  
 }  
 for(int i = 0; i < nums.size(); ++i){  
 if(flag[i] == true)continue;  
 if(i > 0 && nums[i - 1] == nums[i] && flag[i-1])continue;  
 flag[i] = true;  
 tmp.push\_back(nums[i]);  
 dfs(nums, ans, tmp, flag);  
 tmp.pop\_back();  
 flag[i] = false;  
 }  
 }  
 bool cmp(vector<int>arr1, vector<int>arr2){  
 int i = 0;  
 while(i < arr1.size()){  
 if(arr1[i] != arr2[i])return false;  
 i++;  
 }  
 return true;  
 }  
};