Permutation II

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]
]
```

【思路】

递归。DFS；

递归的想法最好先从最简单的情况开始考虑，然后在考虑复杂的情况。

当只有两个元素时，只要交换两个就可以得到两个排列；当有很多个元素时，我们可以考虑将多个元素合并成一个元素来考虑，与剩下的另一个元素作最简单的处理。

```
class Solution {

public:
    vector<vector<int>> permuteUnique(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        vector<vector<int>> res;
        DFS(0, nums, res);
        return res;
    }
private:
    void DFS(int order, vector<int> remain, vector<vector<int>>& res){
        if(order>=remain.size()-1){
            res.push_back(remain);
            return;
        }
        for(int i=order;i<remain.size();++i){
            if(i != order && remain[order]==remain[i]) continue;
            swap(remain[order], remain[i]);
            DFS(order+1, remain, res);
            swap(remain[order], remain[i]);
        }
    }
};
```