

47. Permutations II

Description

Hints

Submissions

Discuss

Solution

Pick One

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]
]
```

```
public List<List<Integer>> permuteUnique(int[] nums) {
    List<List<Integer>> res = new ArrayList<List<Integer>>();
    List<Integer> tmp = new ArrayList<Integer>();
    HashSet<Integer> hashSet = new HashSet<Integer>();
    dfs(res, tmp, nums, hashSet);
    return res;
}

public void dfs(List<List<Integer>> res, List<Integer> tmp, int [] nums, HashSet<Integer> hashSet)
    if(tmp.size() == nums.length) {
        if(!res.contains(tmp)) {
            res.add(new ArrayList<Integer>(tmp));
        }
        return;
    }
    /*
     * 因为任意的元素都可以作为开始节点，所以用hashSet记录已经出现过的元素（记住是记录下标）
     * ，如果出现过，不能重复加载进去
     */
    for(int i = 0; i < nums.length; i++) {
        if(!hashSet.contains(i)) {
            hashSet.add(i); //这里不是nums[i]，而是i
            tmp.add(nums[i]);
            dfs(res, tmp, nums, hashSet);
            tmp.remove(tmp.size() - 1);
            hashSet.remove(i);
        }
    }
}
```