给定一**组**候选号码（***C***）**（没有重复）**和一个目标号码（***T***），找出***C***中所有候选号码总和为***T的***唯一组合。

的**相同**重复数目可以选自***Ç***无限次数。

**注意：**

* 所有数字（包括目标）都是正整数。
* 解决方案集不能包含重复的组合。

例如，给定候选集合[2, 3, 6, 7]和目标7，  
解决方案集合是：

[

[7]，

[2，2，3]

]

允许使用相同数据

public List<List<Integer>> combinationSum(int[] nums, int target) {

List<List<Integer>> list = new ArrayList<>();

Arrays.sort(nums);

backtrack(list, new ArrayList<>(), nums, target, 0);

return list;

}

private void backtrack(List<List<Integer>> list, List<Integer> tempList, int [] nums, int remain, int start){

if(remain < 0) return;

else if(remain == 0) list.add(new ArrayList<>(tempList));

else{

for(int i = start; i < nums.length; i++){

tempList.add(nums[i]);

backtrack(list, tempList, nums, remain - nums[i], i); // not i + 1 because we can reuse same elements

tempList.remove(tempList.size() - 1);

}

}

}

给定候选号码（***C***）和目标号码（***T***）的集合，找出***C***中候选号码总和为***T的***所有唯一组合。

***C中的***每个数字只能在组合中使用**一次**。

**注意：**

* 所有数字（包括目标）都是正整数。
* 解决方案集不能包含重复的组合。

例如，给定候选集合[10, 1, 2, 7, 6, 1, 5]和目标8，  
解决方案集合是：

[

[1,7]，

[1,2,5]，

[2，6]，

[1，1，6]

]

不允许使用相同数据

public List<List<Integer>> combinationSum2(int[] nums, int target) {

List<List<Integer>> list = new ArrayList<>();

Arrays.sort(nums);

backtrack(list, new ArrayList<>(), nums, target, 0);

return list;

}

private void backtrack(List<List<Integer>> list, List<Integer> tempList, int [] nums, int remain, int start){

if(remain < 0) return;

else if(remain == 0) list.add(new ArrayList<>(tempList));

else{

for(int i = start; i < nums.length; i++){

if(i > start && nums[i] == nums[i-1]) continue; // skip duplicates

tempList.add(nums[i]);

backtrack(list, tempList, nums, remain - nums[i], i + 1);

tempList.remove(tempList.size() - 1);

}

}

}