Given a collection of candidate numbers (***C***) and a target number (***T***), find all unique combinations in ***C*** where the candidate numbers sums to ***T***.

Each number in ***C*** may only be used **once** in the combination.

**Note:**

* All numbers (including target) will be positive integers.
* The solution set must not contain duplicate combinations.

For example, given candidate set [10, 1, 2, 7, 6, 1, 5] and target 8,   
A solution set is:

[

[1, 7],

[1, 2, 5],

[2, 6],

[1, 1, 6]

]

这又是一道典型的回溯法枚举的变形。注意这里不允许又相同的元素，在水平上等于前一个元素就跳过。

代码

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

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

Arrays.sort(nums);//排序

backtrack(list,new ArrayList<>(),nums,target,0);//递归开始

return list;

}

public static void backtrack(List<List<Integer>> list,ArrayList<Integer> templist,int[] nums,int remain,int start){

if(remain==0){//当减出来的结果等于0时，符合要求

list.add(new ArrayList<>(templist));//添加到结果list中

}

else if(remain<0)//小于0，退出，当前结果不满足要求，失败

return;

else {

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

**if(i>start&&nums[i]==nums[i-1])//水平重复的跳过，记住！！！！**

**continue;**

**//没有这句话，返回的是有重复的子集**

templist.add(nums[i]);

backtrack(list,templist,nums,remain-nums[i],i+1);//递归

templist.remove(templist.size()-1);//回溯

}

}

}