Given a **set** of candidate numbers (***C***) **(without duplicates)** and a target number (***T***), find all unique combinations in ***C*** where the candidate numbers sums to ***T***.

The **same** repeated number may be chosen from ***C*** unlimited number of times.

**Note:**

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

For example, given candidate set [2, 3, 6, 7] and target 7,   
A solution set is:

[

[7],

[2, 2, 3]

]

这到题，是组合求和问题。我们用回溯法就可以解决，但是回溯法时间复杂度比较高。直接代码简洁。

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

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

Arrays.sort(nums);//对nums进行排序

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

return resultlist;

}

public static void backtrack(List<List<Integer>> resultlist,List<Integer> result,int remain,int[] nums,int start){

if(remain<0) //如果当前结果小于0，结果错误，退出

return;

else if(remain == 0)//如果等于0，符合要求，添加到最终结果中

resultlist.add(**new ArrayList<>(result)**);//注意这里一定要new ArrayList(result)

else {

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

result.add(nums[i]); //添加到当前的结果中

//递归

backtrack(resultlist,result,remain-nums[i],nums,i);//注意这里是i，不是i+1.因为可以重复使用数据

result.remove(result.size()-1);//递归完，回溯

}

}

}

其他问题，如排列组合问题，求子集合问题也同样需要回溯法。