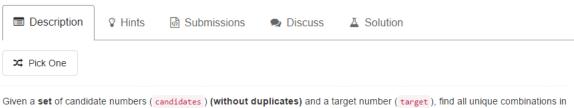
## 39. Combination Sum



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

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

## Note:

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

## Example 1:

```
Input: candidates = [2,3,6,7], target = 7,
A solution set is:
[
  [7],
  [2,2,3]
]
```

## Example 2:

```
Input: candidates = [2,3,5], target = 8,
A solution set is:
[
  [2,2,2,2],
  [2,3,3],
  [3,5]
]
```

```
* @shenh 题目给了我们一个candidates和一个target,让我们从数组中找到所有的组合,它的和是等于target的。
 * 任何组合只要是它的和等于target的话,都需要找到,但是不需要重复的。这道题中是可以重复利用一个数字,那么我们就需要每次
 * 都代入同一个数字,直到它之和达到target或者它超过了target,然后在倒退回去一个数字,继续找下一个数字,这种情况
 * 肯定是需要用递归了。这里是backtracking,每次倒退回一个数字,需要继续递归下去,再倒退,一直重复直到搜索了所有的可能性。
public class L39 {
   public List<List<Integer>> combinationSum(int[] candidates, int target) {
       List<List<Integer>> list = new ArrayList<>();
       Arrays.sort(candidates);
       backtrack(list, new ArrayList<>(), candidates, target, 0);
       return list;
   public static boolean backtrack(List<List<Integer>> list,
           List<Integer> tempList, int[] nums, int remain, int start) {
       //这个是递归的出口
       if (remain < 0)</pre>
           return false;
       else if (remain == 0) {
           list.add(new ArrayList<>(tempList));
           return false;
       } else {
           //这个是递归的展开,递归可以看成是一棵树
           for (int i = start; i < nums.length; i++) {</pre>
              boolean flag;
              tempList.add(nums[i]);
              flag = backtrack(list, tempList, nums, remain - nums[i], i);
               //这里的意思是假如是=0或者是>0,则不需要再深入了,对树进行了剪枝,因为后面的数字都要比这个数字更大
              if (!flag)
                  break;
           return true;
}
```