```
/*
给定一个无重复元素的数组 candidates 和一个目标数 target ，找出 candidates 中所有可以使数字和为 target 的组合。

candidates 中的数字可以无限制重复被选取。

说明：

    所有数字（包括 target）都是正整数。
    解集不能包含重复的组合。

示例 1:

输入: candidates = [2,3,6,7], target = 7,
所求解集为:
[
  [7],
  [2,2,3]
]

示例 2:

输入: candidates = [2,3,5], target = 8,
所求解集为:
[
  [2,2,2,2],
  [2,3,3],
  [3,5]
]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/combination-sum
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 回溯法,为了避免重复，可以先对数组进行排序，进行组合时，只允许从当前位置向更大的索引位置进行相加.

方法一:C++_排序回溯筛选法

```
class Solution
{

    private:

        void helper (   vector<vector<int>>&    vvi             ,
                        vector<int>&            candidates      ,
                        int                     cur_sum         ,
```

```
                        vector<int>&            cur_vi          ,
                        int                     idx             ,
                        int                     target
                    )
        {

            if(cur_sum == target)
            {
                vvi.push_back(cur_vi);
                return;
            }

            /*题目已知全是正整数,不必考虑负数的情况*/

            for(int i = idx ; i < candidates.size() ; i++)
            {
                if(cur_sum+candidates[i] > target)
                {
                    continue;
                }

                cur_vi.push_back(candidates[i]);

 helper(vvi,candidates,cur_sum+candidates[i],cur_vi,i,target);
                cur_vi.pop_back();
            }
        }



    public:
        vector<vector<int>> combinationSum(vector<int>& candidates, int
target)
        {
            vector<vector<int>> vvi;

            if(candidates.size() < 1)
            {
                return vvi;
            }

            vector<int> vi;

            sort(candidates.begin(),candidates.end());
            helper(vvi,candidates,0,vi,0,target);

            return vvi;


        }
};


/*
执行结果:
通过
显示详情
执行用时 :16 ms, 在所有 cpp 提交中击败了85.61% 的用户
```

```
内存消耗 :9.3 MB, 在所有 cpp 提交中击败了93.78%的用户
*/
```

AlimyBreak
2019.11.28