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]
]
```

思路，该题看了下，分类为回溯算法，那么我就用回溯的思路，进行树的深度遍历，每个数给他0组合到n次机会，但其和不能大于目标数，组合后，传递给下一位数，在进行组合，若是等于目标数，将数添加进数组。需注意，当目标数为0时，不要在进行遍历，避免重复添加

```
class Solution {
public:
    void dfs(vector<int> nums,vector<vector<int>> &result,vector<int>
&candidates,int wz,int target){
        if(wz==candidates.size()||target==0){
            return;
        }
        for(int i=0;candidates[wz]*i<=target;i++){
            dfs(nums,result,candidates,wz+1,target-candidates[wz]*i);
            if(candidates[wz]*i==target){
                result.push_back(nums);
            }
            nums.push_back(candidates[wz]);
        }
    }
```

```
    vector<vector<int>> combinationSum(vector<int>& candidates, int
target) {
        sort(candidates.begin(),candidates.end());
        vector<int>nums;
        vector<vector<int>>result;
        dfs(nums,result,candidates,0,target);
        return result;
    }
};
```