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.

Elements in a combination (a1, a2, … , ak) must be in non-descending order. (ie, a1 ≤ a2 ≤ … ≤ ak).

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]

方法1：

class Solution {

public:

void dfs(vector<int>& candidates, int start,int target,int &sum){

int size = candidates.size();

for (int i = start; i < size;){

if (sum + candidates[i] <= target){

temp.push\_back(candidates[i]);

sum += candidates[i];

if (sum == target){

result.push\_back(temp);

temp.pop\_back();

sum -= candidates[i];

return;//剪枝操作

}

dfs(candidates, i + 1, target, sum);

temp.pop\_back();

sum -= candidates[i];

}

int j = i + 1;

while (j < size){

if (candidates[i] == candidates[j])++j;

else break;

}

i = j;

}

}

vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {

int size = candidates.size();

if (size == 0)return result;

int sum = 0;

Sort(candidates.begin(), candidates.end());

dfs(candidates,0,target,sum);

return result;

}

vector<int>temp;

vector<vector<int>>result;

};

方法2：

class Solution {

public:

void dfs(vector<int>& candidates, int start,int target,int &sum){

int size = candidates.size();

for (int i = start; i < size;i++){

if (i!=start&&candidates[i]==candidates[i-1])continue;

if (sum + candidates[i] <= target){

temp.push\_back(candidates[i]);

sum += candidates[i];

if (sum == target){

result.push\_back(temp);

temp.pop\_back();

sum -= candidates[i];

return;//剪枝操作

}

dfs(candidates, i + 1, target, sum);

temp.pop\_back();

sum -= candidates[i];

}

}

}

vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {

int size = candidates.size();

if (size == 0)return result;

int sum = 0;

sort(candidates.begin(), candidates.end());

dfs(candidates,0,target,sum);

return result;

}

vector<int>temp;

vector<vector<int>>result;

};

分析：

题目要求：

1. 所有的combination不能重复;

对于第一点，我们看到下图所示的遍历数有如下特点，如果两个数字有重复，则第二个数字的遍历情况会被前一个数包含(图中的两个1),这种情况下应该直接跳过第二个1,否则如果target=4,就会出现(1,3)和(1,3)的情况.

1. 在任何一个combination里面，同一个数只能被使用一次;

对于第二点我们先将数组进行排序，然后每次DFS遍历比当前index大的结点，那么就可以保证同一个combination内的所有数字只出现一次;

假设有如下数组num=[1,1,2,2,3],target=4利用回溯法构造如下遍历树:

