40. Combination Sum II

Given a collection of candidate numbers (`candidates`) and a target number (`target`), find all unique combinations in `candidates` where the candidate numbers sum to `target`.

Each number in `candidates` may only be used **once** in the combination.

Note: The solution set must not contain duplicate combinations.

Example 1:

```
Input: candidates = [10,1,2,7,6,1,5], target = 8
Output:
[
[1,1,6],
[1,2,5],
[1,7],
[2,6]
]
```

Example 2:

```
Input: candidates = [2,5,2,1,2], target = 5
Output:
[
[1,2,2],
[5]
]
```

Constraints:

- `1 <= candidates.length <= 100`
- `1 <= candidates[i] <= 50`
- `1 <= target <= 30`

40. Combination Sum II

```
/*
   Time compelxity: (n*2^target)
   Space complexity: O(target+n)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
class Solution {
public:
   vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {
      int n=candidates.size();
      std::sort(candidates.begin(),candidates.end());
      vvi ans;
      auto solve=[&](int i,vi& tmp, int sum,auto& self)->void{
         if(sum==0) {
            ans.push_back(tmp);
            return;
         }
         if(i>=n || sum<0) return;

         // Include candidates[i]
         tmp.push_back(candidates[i]);
         // Explore
         self(i+1,tmp,sum-candidates[i],self);
         // Exclude candidates[i]
         tmp.pop_back();

         while(i+1<n&&candidates[i]==candidates[i+1]) i++;
         self(i+1,tmp,sum,self);
      };

      vi tmp;
      solve(0,tmp,target,solve);
      return ans;
   }
};
```

40. Combination Sum II

```
/*
   Time compelxity: (n*2^target)
   Space complexity: O(target+n)
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
public:
   vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {
      int n=candidates.size();

      std::sort(candidates.begin(),candidates.end());
      vvi ans;
      auto solve=[&](int i,vi& tmp, int sum,auto& self)->void{
         if(sum==0) {
            ans.push_back(tmp);
            return;
         }
         if(sum<0) return;
         for(int j=i;j<n;++j){
            if(j>i&&candidates[j]==candidates[j-1]) continue;
            // Include candidates[i]
            tmp.push_back(candidates[j]);
            // Explore
            self(j+1,tmp,sum-candidates[j],self);
            // Exclude candidates[i]
            tmp.pop_back();
         }
      };

      vi tmp;
      solve(0,tmp,target,solve);
      return ans;
   }
};
```