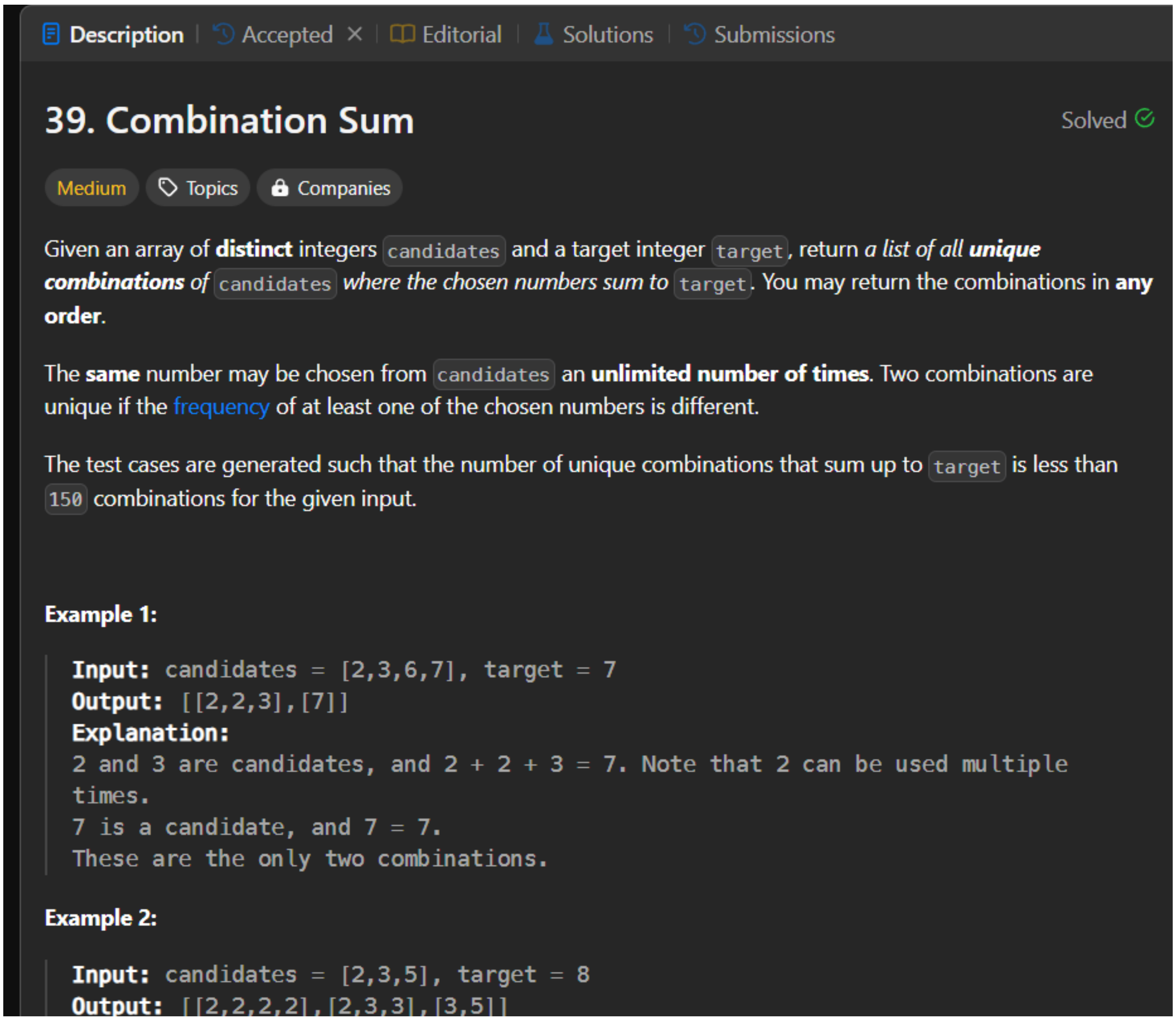

Description | Accepted × | Editorial | Solutions | Submissions

39. Combination Sum

Solved

Medium · Topics · Companies

Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all* ***unique combinations*** *of* `candidates` *where the chosen numbers sum to* `target`*.* You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

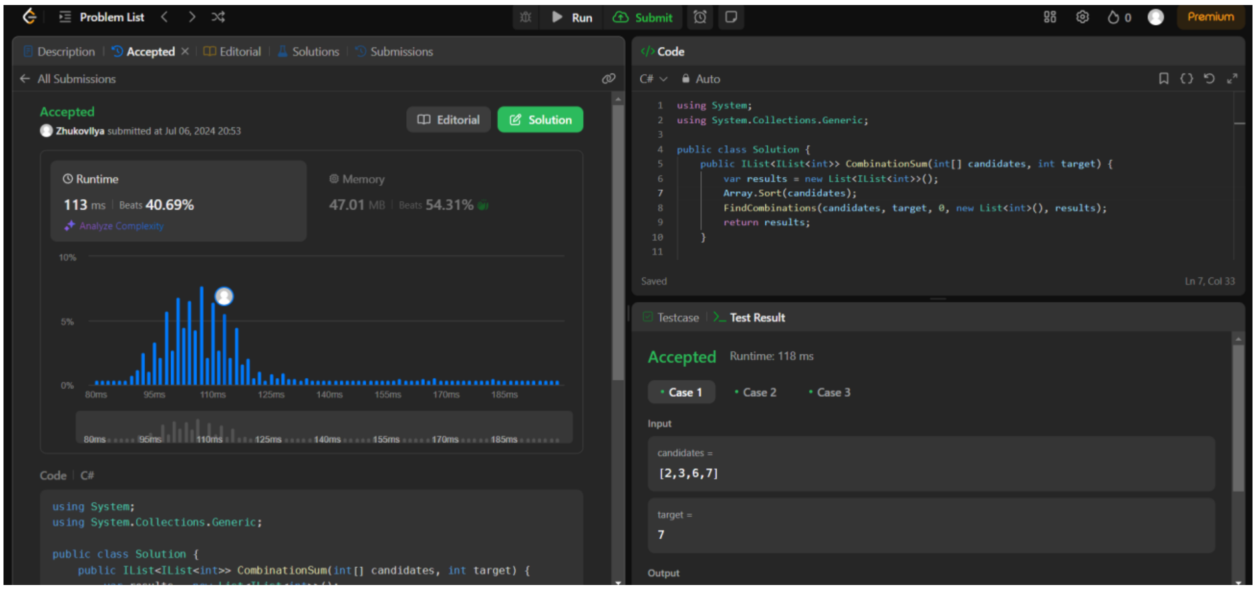
The test cases are generated such that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

Example 1:

```
Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple
times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
```

Example 2:

```
Input: candidates = [2,3,5], target = 8
Output: [[2,2,2,2],[2,3,3],[3,5]]
```



Accepted
Zhukovllya submitted at Jul 06, 2024 20:53

Runtime
113 ms | Beats 40.69%

Memory
47.01 MB | Beats 54.31%

Code | C#

```
using System;
using System.Collections.Generic;

public class Solution {
    public IList<IList<int>> CombinationSum(int[] candidates, int target) {
```

```
using System;
using System.Collections.Generic;

public class Solution {
    public IList<IList<int>> CombinationSum(int[] candidates, int target) {
        var results = new List<IList<int>>();
        Array.Sort(candidates);
        FindCombinations(candidates, target, 0, new List<int>(), results);
        return results;
    }

```

Accepted Runtime: 118 ms

Input

candidates =
[2,3,6,7]

target =
7

Output

Код:

```
using System;
using System.Collections.Generic;

public class Solution
{
    public IList<IList<int>> CombinationSum(int[] candidates, int target)
    {
        var results = new List<IList<int>>();
        Array.Sort(candidates);
        FindCombinations(candidates, target, 0, new List<int>(), results);
        return results;
    }

    private void FindCombinations(int[] candidates, int target, int start, List<int>
current, List<IList<int>> results)
    {
        if (target == 0)
        {
            results.Add(new List<int>(current));
            return;
        }

        for (int i = start; i < candidates.Length; i++)
        {
            if (candidates[i] > target)
            {
                break;
            }
            current.Add(candidates[i]);
            FindCombinations(candidates, target - candidates[i], i, current,
results);
            current.RemoveAt(current.Count - 1);
        }
    }
}
```