CPP SCREENER

**Problem 1: (level : 1 : Expected time 20 mins)**

You are given an unsorted Integer array which consists of positive and negative numbers. You have to find the index where the left side and the right of the index are balanced. At a given index we say the left and right side are balanced, when the sum of elements at lower indexes is equal to the sum of elements at higher indexes.

For example, in an array A:

A[0] = -7, A[1] = 1, A[2] = 5, A[3] = 2, A[4] = -4, A[5] = 3, A[6]=0

The array is balanced at Index 3 (because sum of elements to the Left of A[3] is equal to the sum of elements to the right of A[3])

A[0] + A[1] + A[2] = A[4] + A[5] + A[6]

The array cannot be balanced at an index, which does not have a right or left element.

The Size of the array => 3 <= Size < = 1000

Write a C++ function which takes the array and length as the input and returns the index where the array is balanced (if any) or -1 if no such indexes exist.

Incase there are multiple indexes where the array balances, then the function can return just the first index where the array was balanced.

Use proper boundary conditions. Do not use brute force method. Propose an optimized solution in terms of time and space complexity.

**Solution Hint :**

* Should solve with a time complexity of O(n).
* First calculate the sum of all the elements in the array.
* Then have two variables, RightSum and LeftSum.
* Iterate through the array and calculate Left sum and right sum for each element.
* LeftSum is calculated by adding the value of the current Index to the left sum.
* Right Sum is calculated by subtracting leftSum from the total.
* Whenever LeftSum is equal to RightSum, return the index.

**Problem 2 : (Level – 2 : Expected time 25 mins)**

There are N jars of chocolates. Each Jar has 50 chocolates. The cost of each chocolate in each jar is given in an array of size n. (The value at each index represents the cost of one chocolate) You are given a cash amount of X. The idea here is given this amount X, you have to pick up the fewest number of chocolates that can be got for this amount. The value of the chocolates should exactly match with the amount. Write a function which takes the array and amount as input and returns you the fewest number of chocolates that can be picked for the amount. In case, if you cannot pick the chocolates such that it matches the value of money, then return -1.

The value of N is in the range of 2 to 100.

Example 1 :

**Input:** There are 5 jars and the cost of chocolates in each jar = [1, 2, 5, 11, 9],

Cash given to you = 23

**Output:** 3

**Reason :** 24 = 11 + 11 + 2

Example 2:

**Input:** Chocolates = [5, 7, 2], amount = 3

**Output:** -1

**Solution Hint :**

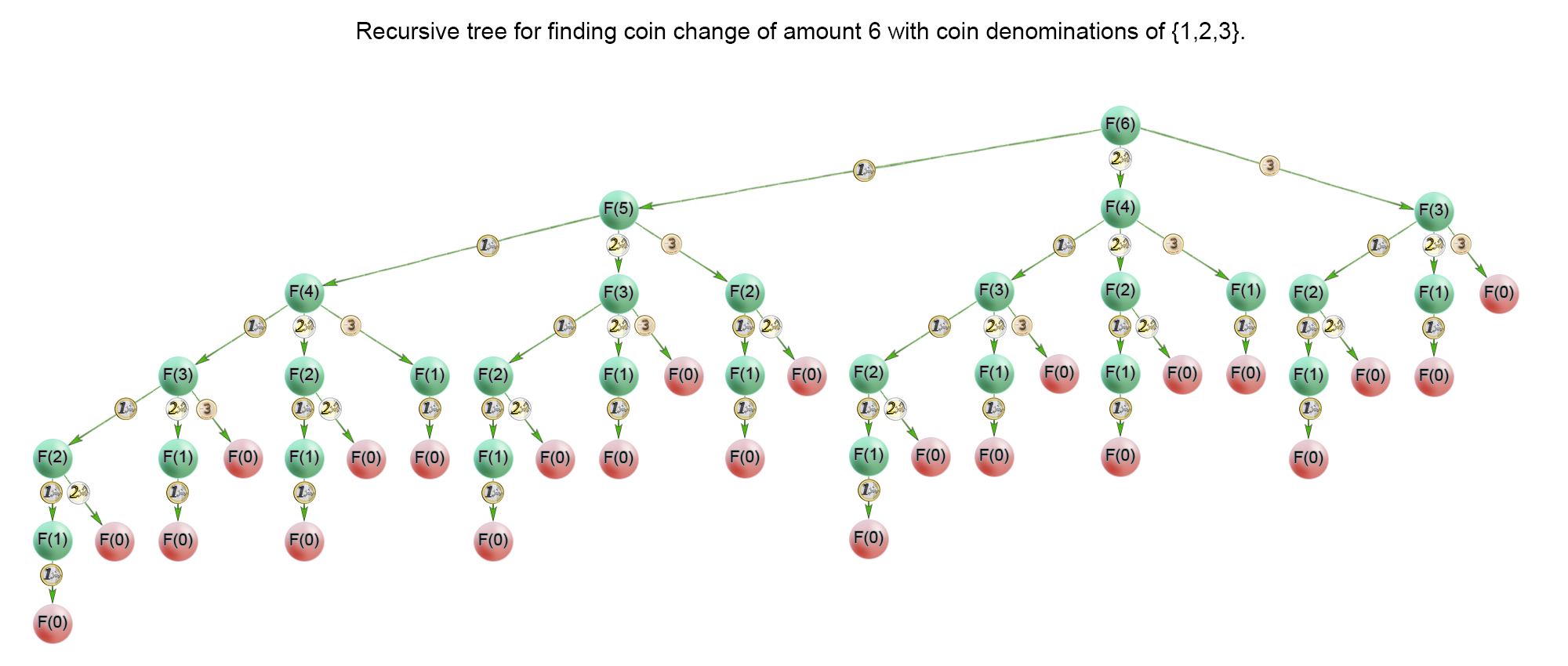
* The Idea here is to use a recursive approach (Other approaches except brute force are welcomes as well)
* If Cash == 0, then 0 Chocolates can be picked.
* If Cash > 0

MinChocolates (Chocolates[0..m-1], V) = min {1 + MinChocolates (V-chocolate[I])}

where i varies from 0 to m-1

and chocolate[I] <= V

Recursive tree When you Cost in each Chocolates Jars = [1, 2, 3], amount = 6



**Problem 3: ( Level : 3 Time : 45 mins)**

Given a list of names, determine the number of names in that list for which a given query string is a prefix.

For example, given the list *names = [jackson, jacques, jack]*, the complete *query* string *jack* is a prefix of *jackson* but not of *jacques* or *jack*. The prefix cannot contain the entire name string.

**Function Description**

Complete the function *findCompletePrefixes* in the editor below. The function must return an array of integers that each denotes the number of *names* strings for which a *query* string is a prefix.

findCompletePrefixes has the following parameter(s):

*names[names[0],...names[n-1]]:*  an array of name strings

*query[query[0],...query[n-1]]:*  an array of query strings

**Constraints**

* *1 ≤ n ≤ 20000*
* *2 ≤ |names[i]| ≤ 30,*
* *1 ≤ sum of all |names[i]| ≤ 5 x 105*
* *1 ≤ q ≤ 200*
* *2 ≤ |query[i]| ≤ 30*

*Example Input*

***Input names given:***

steve

stevens

danny

steves

dan

john

johnny

joe

alex

alexander

**Input Queries Given**

steve

alex

joe

john

dan

**Output**

2

1

0

1

1

**Explanation**

Query 1: steve appears as a prefix in two strings: stevens and steves.

Query 2: alex appears as a prefix in one string: alexander.

Query 3: joe does not appear as a prefix in any string.

Query 4: john appears as a prefix in one string: johnny.

Query 5: dan appears as a prefix in one string: danny.

**Solution Hint :**

Use a **trie** data structure.

Write code to add and retrieve from trie.