DSA - Algorithms

Dynamic Programming 2

Course Planning

Algorithms	Data Structures	Algorithmic Approaches	Interview Practices
1.Introduction	1.Asymptotic Analysis	1.Search Algorithms	1.In-place Reversal
2.Number 1	2.Dynamic Array	2.Sort Algorithms	2.Two Heaps
3.Number 2	3.LinkedList	3.Dac Algorithms	3.Subsets
4.String 1	4.Stack	4.Recursion	4.Modified BS
5.String 2	5.Queue	5.Sliding Window	5.Bitwise XOR
6.Array 1	6.Tree	6.Two Pointers	6.Top 'K' Elements
7.Array 2	7.Heap	7.Fast & Slow	7.K-way Merge
8.Matrix	8.Trie	8.Cyclic Sort	8.Knapsack Problem
9.DP 1	9.Graph	9.Breadth First Search	9.Topological Sort
10.DP 2	10.Undirected Graph	10.Depth First Search	10.Mock Interview

https://kurbanovxurshid.medium.com/data-structures-and-algorithms-course-plan-1ab912ec1e17

Asked by Facebook

Explanation

518. Coin Change 2

Medium 3164 78 Add to List Share

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the number of combinations that make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `0`.

You may assume that you have an infinite number of each kind of coin.

The answer is **guaranteed** to fit into a signed **32-bit** integer.

Example 1:

```
Input: amount = 5, coins = [1,2,5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1
```

Example 2:

```
Input: amount = 3, coins = [2]
Output: 0
Explanation: the amount of 3 cannot be made up just with coins of 2.
```

Coin Change 2

518. Coin Change 2

Medium 3164 78 Add to List Share

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the number of combinations that make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `0`.

You may assume that you have an infinite number of each kind of coin.

The answer is **guaranteed** to fit into a signed **32-bit** integer.

Example 1:

```
Input: amount = 5, coins = [1,2,5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1
```

Example 2:

```
Input: amount = 3, coins = [2]
Output: 0
Explanation: the amount of 3 cannot be made up just with coins of 2.
```

Example 3:

```
class Solution {

    public int change(int amount, int[] coins) {

    }
}

$
```

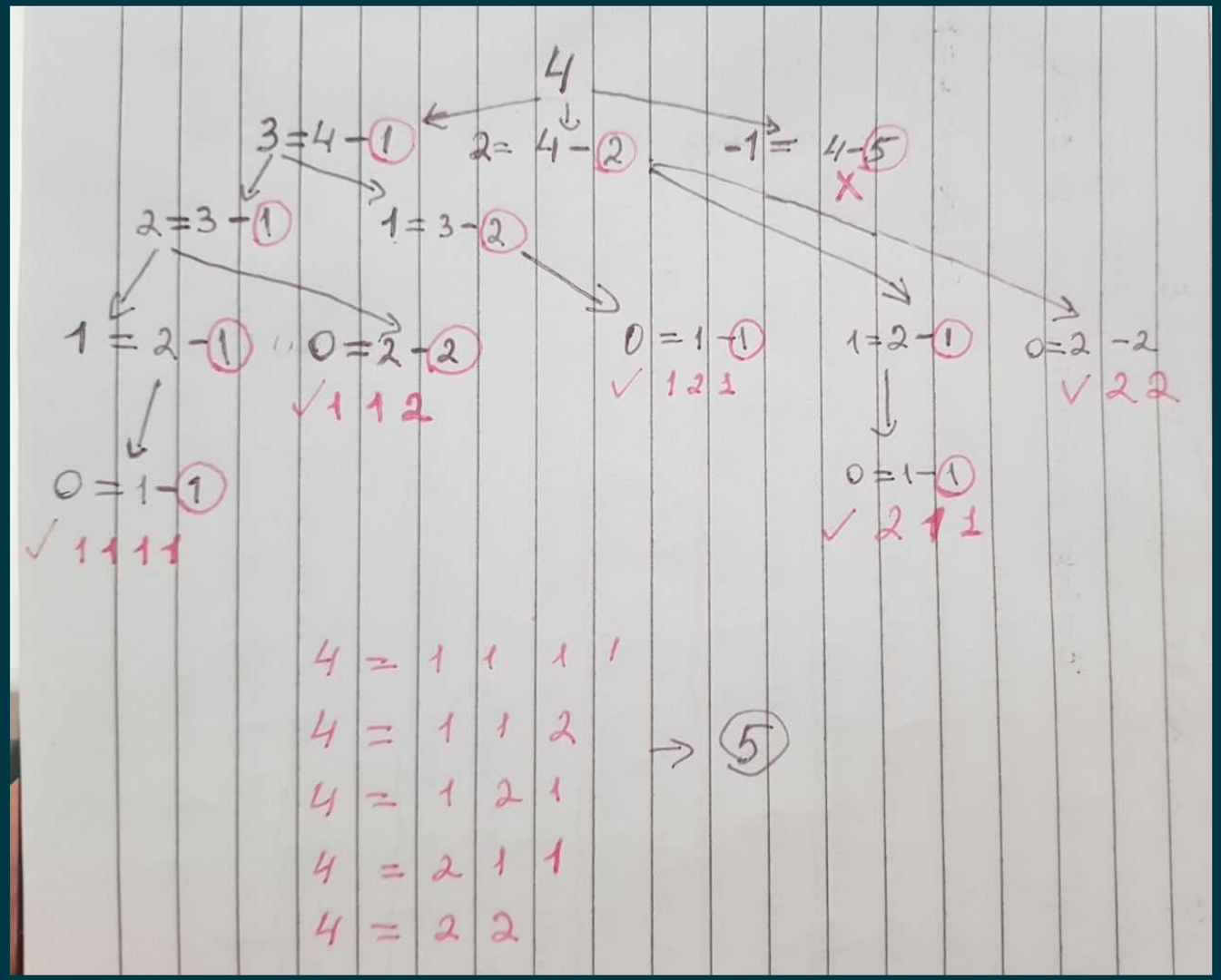
Testcase | Run Code Result | Debugger

```
5
[1,2,5]
```

Problems | Pick One | Prev 518/1873 Next | Console | Use Example Testcases | Run Code | Submit

https://leetcode.com/problems/coin-change-2/

First Theory



result(x) = result(x-1) + result(x-2) + result(x-5)

First Solution

Time Limit Exceeded Details ›

Last executed input

```
500
[3,5,7,8,9,10,11]
```

Time Submitted	Status	Runtime	Memory	Language
05/26/2021 11:14	Time Limit Exceeded	N/A	N/A	java
05/26/2021 11:14	Time Limit Exceeded	N/A	N/A	java
05/26/2021 11:14	Time Limit Exceeded	N/A	N/A	java
05/26/2021 11:13	Time Limit Exceeded	N/A	N/A	java
05/26/2021 10:25	Time Limit Exceeded	N/A	N/A	java

```java
class Solution {
    private int[] coins;

    public int change(int amount, int[] coins) {
        this.coins = coins;
        return result(amount, 0);
    }

    private int result(int amount, int coin) {
        if(amount == 0) {
            return 1;
        }

        if(amount < 0) {
            return 0;
        }

        int res = 0;
        for (int i = coin; i < coins.length; i++) {
            res += result(amount - coins[i], i);
        }
        return res;
    }
}
```

Second Theory

coins = [1,2,5]
target = 5

dp[i] += dp[i - coin];

[1, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1]
[1, 1, 2, 2, 3, 3]
[1, 1, 2, 2, 3, 4]

Second Solution

Success Details ›

Runtime: 2 ms, faster than 100.00% of Java online submissions for Coin Change 2.

Memory Usage: 36.3 MB, less than 76.22% of Java online submissions for Coin Change 2.

Next challenges:

Number of Atoms | Insert into a Sorted Circular Linked List | Capacity To Ship Packages Within D Days

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language

```java
class Solution {

    public int change(int amount, int[] coins) {

        int[] dp = new int[amount + 1];
        dp[0] = 1;

        for(int coin : coins) {
            for(int i = coin; i <= amount; i++) {
                dp[i] += dp[i - coin];
            }
        }

        return dp[amount];
    }
}
```

Task 1 – Coin Change

322. Coin Change

Medium 6990 195 Add to List Share

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the fewest number of coins that you need to make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `-1`.

You may assume that you have an infinite number of each kind of coin.

Example 1:

```
Input: coins = [1,2,5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1
```

Example 2:

```
Input: coins = [2], amount = 3
Output: -1
```

Example 3:

```
Input: coins = [1], amount = 0
Output: 0
```

https://leetcode.com/problems/coin-change/

Task 2 – Maximum Subarray

53. Maximum Subarray

Easy 12093 583 Add to List Share

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

Example 2:

```
Input: nums = [1]
Output: 1
```

Example 3:

```
Input: nums = [5,4,-1,7,8]
Output: 23
```

https://leetcode.com/problems/maximum-subarray/

Task 3 – Is Subsequence

392. Is Subsequence

Easy 2524 240 Add to List Share

Given two strings `s` and `t`, return `true` *if* `s` *is a* ***subsequence*** *of* `t`, *or* `false` *otherwise*.

A **subsequence** of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e., `"ace"` is a subsequence of `"abcde"` while `"aec"` is not).

Example 1:

```
Input: s = "abc", t = "ahbgdc"
Output: true
```

Example 2:

```
Input: s = "axc", t = "ahbgdc"
Output: false
```

Constraints:

- `0 <= s.length <= 100`
- `0 <= t.length <= 10^4`
- `s` and `t` consist only of lowercase English letters.

https://leetcode.com/problems/is-subsequence/