322. Coin Change

Created	@June 10, 2021 1:14 PM
Tags	Medium
link	https://leetcode.com/problems/coin-change/

Description:

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.

Solution:

Approach-

- Formulate the problem as a recursive function (here, X is input and Y is set of coins available)

$$CoinsNeeded(X) = min_{y \in Y}(1 + CoinsNeeded(X - y))$$

- Code the solution for recursive problem. It would get TLE but should be accepted for basic inputs.
- Observe that this problem has `optimal substructure` (i.e. sub-problem is repeated multiple times and its value can be reused).
- Using above formulation, think of implementing top down approach, i.e adding a cache `minCoinsNeeded` to store subproblems.

```
class Solution {
public:
    map<int, int> minCoinsNeeded;

    // give the minimum number of coins needed to change 'amountLeft' value with
```

```
    // coins available in 'coins' vector
    int runCoinChange(vector<int>& coins, int amountLeft) {

        if (amountLeft < 0) return -1;

        if (amountLeft == 0) return 0;

        // using cache to convert the problem to top-down dp
        if (minCoinsNeeded.find(amountLeft) != minCoinsNeeded.end()) {
            return minCoinsNeeded[amountLeft];
        }

        // recursive function implementation
        int minCoins = INT_MAX;
        for (auto i: coins) {
            if (i <= amountLeft) {
                int res = runCoinChange(coins, amountLeft - i);
                if (res == -1) continue;
                else {
                    int val = 1 + res;
                    minCoins = min(minCoins, val);
                }
            }
        }
        minCoinsNeeded[amountLeft] = (minCoins==INT_MAX) ? -1 : minCoins;
        return minCoinsNeeded[amountLeft];
    }

    int coinChange(vector<int>& coins, int amount) {
        return runCoinChange(coins, amount);
    }
};
```