322. Coin Change

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
```

Constraints:

- $1 \leq n \leq 12$
- $1 \leq coins_i \leq 2^{31} - 1$
- $0 \leq amount \leq 10^4$

Solution

$$
\begin{gathered}
f(0) = 0, \\
f(a) = \min_{c \in \text{coins}} \left(1 + f(a - c)\right) \text{ for } a > 0, \\
f(a) = \infty \text{ for } a < 0
\end{gathered}
$$

```
import math
class Solution:
    def rec(self, curr_amount, coins):
        if curr_amount < 0:
            return math.inf
        if curr_amount == 0:
            return 0
        return min([self.rec(curr_amount - coin, coins) + 1 for coin in coins])
    def coinChange(self, coins: List[int], amount: int) → int:
        ans = self.rec(curr_amount = amount, coins = coins)
        return ans if ans != math.inf else -1
```

$$
f(a) = \begin{cases}
M[a] & a \in M \\
0 & a = 0 \\
\infty & a < 0 \\
\min_{c \in \text{coins}}(1 + f(a - c)) & \text{otherwise (store in } M[a]\text{)}
\end{cases}
$$

```
import math
class Solution:
    def mem(self, curr_amount, coins, d):
        if curr_amount < 0:
            return math.inf
        if curr_amount == 0:
            return 0
        if curr_amount not in d:
            d[curr_amount] = min([self.mem(curr_amount - coin, coins, d) +
            1 for coin in coins])
        return d[curr_amount]
    def coinChange(self, coins: List[int], amount: int) → int:
```

```
        d = {}
        ans = self.mem(curr_amount = amount, coins = coins, d = d)
        return ans if ans != math.inf else -1
```

$$dp[i] = \min_{c \in \text{coins},\ c \le i} \big(dp[i-c] + 1\big), \quad \forall i \in [1, \text{amount}]$$

```
import math
class Solution:
    def tab(self, coins, amount):
        dp = [math.inf for _ in range(amount + 1)]
        dp[0] = 0
        for i in range(1, len(dp)):
            dp[i] = 1 + min([dp[i - coin] for coin in coins if i - coin >= 0],
                            default=math.inf)
        return dp[amount]
    def coinChange(self, coins: List[int], amount: int) -> int:
        ans = self.tab(coins = coins, amount = amount)
        return ans if ans != math.inf else -1
```