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 <= coins.length <= 12`
- `1 <= coins[i] <= 2`31` - 1`
- `0 <= amount <= 10`4

322. Coin Change

```
/*
    DFS
    Time complexity: O(n^amount)
    Extra space complexity: O(1)
*/
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        int n=coins.size();

        auto solve=[&](int amount,auto& self)->int{
            if(amount<0) return INT_MAX;
            if(amount==0) return 0;

            int ans=INT_MAX;
            for(int i=0;i<n;++i){
                int count=self(amount-coins[i],self);
                if(count!=INT_MAX) ans=std::min(ans,1+count);
            }
            return ans;
        };

        int ans=solve(amount,solve);
        return ans!=INT_MAX?ans:-1;
    }
};
```

322. Coin Change

```
/*
   DFS+DP memoization
   Time complexity: O(n^2)
   Extra space complexity: O(2n)
*/
class Solution {
   public:
      int minHeightShelves(std::vector<std::vector<int>>& books, int shelfWidth) {
         int n=books.size();
         std::vector<int> memo(n,-1);
         auto solve=[&](int i,auto& self)->int{
            if(i==n) return 0;
            if(memo[i]!=-1) return memo[i];
            int shelf_w=shelfWidth;
            int max_height=INT_MIN;
            int ans=INT_MAX;
            for(int j=i;j<n;++j){
               int w=books[j][0];
               int h=books[j][1];
               if (shelf_w<w) break;
               shelf_w-=w;
               max_height=std::max(max_height,h);
               ans=std::min(ans,self(j+1,self)+max_height);
            }
            return memo[i]=ans;
         };
         return solve(0,solve);
      }
};
```

322. Coin Change

```
/*
    DP: Bottom up
    Time complexity: O(n*amount)
    Extra space complexity: O(amount)
*/
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        std::vector<int> dp(amount+1,INT_MAX);
        dp[0]=0;
        for(int i=1;i<=amount;++i){
            for(auto& coin: coins){
                if(i-coin>=0&&dp[i-coin]!=INT_MAX) dp[i]=std::min(dp[i],1+dp[i-coin]);
            }
        }
            return dp[amount]!=INT_MAX?dp[amount]:-1;
    }
};
```