Dynamic Programming Assignment

Question 1

Problem 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.

Input Description:

An integer array `coins` representing coins of different denominations.

An integer `amount` representing a total amount of money.

Output Description:

The fewest number of coins needed to make up the amount. If the amount cannot be made up, return -1.

Examples:

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

Question 2

Problem Description:

Given a string `s`, return the longest palindromic substring in `s`.

Input Description:

A string `s`.

Output Description:

The longest palindromic substring in `s`.

Examples:

Example 1:

- Input: `s = "babad"`

- Output: `"bab"`

- Explanation: `"aba"` is also a valid answer.

Example 2:

- Input: `s = "cbbd"`

- Output: `"bb"`

Constraints:

1 <= s.length <= 1000

s consists of only digits and English letters.

Question 3

Problem Description:

For a string sequence, a string word is k-repeating if word concatenated k times is a substring of sequence. The word's maximum k-repeating value is the highest value k where word is k-repeating in sequence. If word is not a substring of sequence, word's maximum k-repeating value is 0.

Given strings sequence and word, return the maximum k-repeating value of word in sequence.

Input Description:

Two strings `sequence` and `word`.

Output Description:

The maximum k-repeating value of `word` in `sequence`.

Examples:

Example 1:

- Input: `sequence = "ababc"`, `word = "ab"`

- Output: `2`

- Explanation: `"abab"` is a substring in `"ababc"`.

Example 2:

- Input: `sequence = "ababc"`, `word = "ba"`

- Output: `1`

- Explanation: `"ba"` is a substring in `"ababc"`. `"baba"` is not a substring in `"ababc"`.

Example 3:

- Input: `sequence = "ababc"`, `word = "ac"`

- Output: `0`

- Explanation: `"ac"` is not a substring in `"ababc"`.

Constraints:

1 <= sequence.length <= 100

1 <= word.length <= 100

sequence and word contain only lowercase English letters.

Question 4

Problem Description:

Given an integer array `nums`, find the subarray with the largest sum, and return its sum.

Input Description:

An integer array `nums`.

Output Description:

The sum of the subarray with the largest sum.

Examples:

Example 1:

- Input: `nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]`

- Output: `6`

- Explanation: The subarray `[4, -1, 2, 1]` has the largest sum `6`.

Example 2:

- Input: `nums = [1]`

- Output: `1`

- Explanation: The subarray `[1]` has the largest sum `1`.

Example 3:

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

- Output: `23`

- Explanation: The subarray `[5, 4, -1, 7, 8]` has the largest sum `23`.

Constraints:

1 <= nums.length <= 10^5

-10^4 <= nums[i] <= 10^4

Question 5

Problem Description:

You are climbing a staircase. It takes `n` steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Input Description:

An integer `n`.

Output Description:

The number of distinct ways to climb to the top.

Examples:

Example 1:

- Input: `n = 2`

- Output: `2`

- Explanation: There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

Example 2:

- Input: `n = 3`

- Output: `3`

- Explanation: There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step

Constraints:

1 <= n <= 45