Assignment 2 – Computational Problem Solving

Prompts Used:

1. Given an unsorted integer array nums of size n containing numbers from 1 to n, find all the numbers that are missing from the array.
2. Given an integer array nums, move all even integers to the beginning of the array followed by all odd integers. Return the array in-place. I also provided the code which I wrote.
3. If I am given an array of integers, and a target. I need to find out that which 2 numbers add up to the target and consider the edge cases.
4. Given an integer array nums, find three numbers whose product is the maximum and return the product.
5. Want to write a function that converts a decimal number to its binary equivalent.
6. Given a sorted array that has been rotated, find the minimum element.
7. Can u help me explain the code of Palindrome
8. Can u help me explain the code of Fibonacci.

Responses Received:

1. You are given an **unsorted integer array** nums of size n, containing numbers from 1 to n. Some numbers appear **once**, some **twice**, and some might be **missing**. Your task is to **find all the numbers** that are **missing** from the array.
2. They told your code is perfectly working fine but since you are using a new array to solve the question it does not meet the constraints given, instead consider using a 2 pointer approach. Time complexity was the same with my code and the code provided but in space for me it was o(n) and for the answer provided it was o (1).
3. The problem you're describing is a common one called **Two Sum**, where the goal is to find two distinct indices in the given array nums such that the elements at those indices add up to the given target. Suggested me to use the dictionary approach.
4. To solve this problem, we need to find three numbers in the given array nums whose product is the maximum and return the product. The naive approach of checking all triplets would take O(n³) time, but we can solve this efficiently with a sorting approach.
5. To convert a decimal number to its binary equivalent, we can repeatedly divide the number by 2 and record the remainders. These remainders, when read in reverse order, will give us the binary representation.
6. To find the minimum element in a **rotated sorted array**, we can use a **modified binary search**. The array is originally sorted, but then rotated, so the array might no longer be sorted in a straightforward manner. However, the key property of the rotated array is that the smallest element will be the only element that is smaller than its previous element.
7. To check if a given integer is a palindrome, we need to determine if the number reads the same backward as forward. A palindrome number is one that is identical when its digits are reversed.
8. **Iterative Approach (Optimized):**

The iterative approach is much more efficient because it only calculates each Fibonacci number once, using a loop instead of recursion

Implementation Details:

1. It provides me the code on how to find the missing number from an unsorted array ranging from 1 to n. I forgot to specify the edge cases.
2. In this case I revamped the code I wrote using a 2-pointer approach.
3. I had to change the approach by using the dictionary.
4. Sort the array.

Compare the product of:

The last three numbers (the largest).

The first two numbers (smallest, possibly negative) and the last number (largest positive).

Return the maximum of the two products.

1. **Repeated Division by 2**: Divide the decimal number by 2, keeping track of the remainders. The remainder will either be 0 or 1, representing a binary digit (bit).

**Continue Until the Number is Zero**: Keep dividing the number by 2 until it becomes zero.

**Build the Binary Number**: Collect the remainders and read them in reverse to form the binary number.

1. Binary Search Algorithm

Identify the Minimum

Termination Condition

1. Initial Check, Reversing the 2nd half and then finally. Comparing the new string with the old one.
2. Base Case: If n is 0 or 1, return n. This handles the simple cases for F(0) and F(1).

**Iterative Calculation**: We initialize two variables, a = 0 and b = 1.

In the loop, starting from 2 up to n, we:

Calculate the next Fibonacci number as the sum of a and b.

Shift a to b and update b to the new Fibonacci number.

**Final Return**:

Adjustments:

1. The edge cases were not considered, since we did not explicitly mention so I manually entered the edge cases that if the nums is blank or length of the nums = 0 then just return the array directly.
2. As mentioned above I understood why they are using the 2-pointer approach instead of creating a new array and creating extra space and not meeting the constraints.
3. Implemented the dictionary approach rather than just using the normal 2-point approach.
4. Wrote the prompt again to consider the edge cases.
5. Considered edge cases if the decimal number is 0.
6. Tested the logic on 1-element, multi-sorted and sorted arrays too.
7. Got the idea and then wrote my own code to check for palindrome. For odd numbers it was not working properly then checked again and modified my code for odd numbers too.
8. Got the idea and wrote my own code to calculate the Fibonacci.