Assignment 1: Problem Solving (CLO-2)
Name: Hammad Haroon Shah, Huzaifa Khawar
Enrollment Number: 01-131232-102, 01-131232-034
Link:

1. **Imagine you are developing a GPS navigation system. You are given a map with various locations and the roads connecting them.**

   We can solve this problem using an algorithm called Dijkstra's algorithm. As input, we will take a list of all the locations and the distance between each pair of locations, can also be referred to as the road network. We will output the shortest path from the starting location to the destination location and the total distance of that path.

   1. Create a list that will keep track of the distance from the starting location, which can be set to 0, to each location, the distance for which can be set to infinity.
   2. Also create a list that keeps track of all visited locations. Initially, all locations are unvisited so the list will be empty.
   3. While there are unvisited locations:
      - Find the unvisited location that has the smallest known distance from the starting location, this will be the "current location".
      - For the current location, consider all of the locations connected to it by roads, called these locations "neighbors":
        - Calculate the distance to each "neighbor" through the current location.
        - If the calculated distance is less than the known distance, update the distance.
      - Mark the location as visited.
   4. Once all the locations have been visited, we will have the shortest distance from the starting point to all other locations. To find the shortest path, we will work backwards from the destination location to the starting location by following the path with the smallest distance.
   5. Return the shortest path, which is a sequence of locations, and the total distance from the starting location to the destination location.

2. **You are working on a project where you need to sort a list of numbers in ascending order. Design an algorithm to efficiently sort a list of integers. You should consider various sorting algorithms, evaluate their time complexity, and choose the most suitable one for the task.**

   To sort a list of integers in an efficient manner, we can use a number of sorting algorithms, depending on the size of the list of unsorted integers. If the list is small, we can opt for:

   - bubble sort
   - insertion sort
   - selection sort

   Likewise, to efficiently sort larger lists, we can utilize:

   - merge sort
   - quick sort

   The process of sorting through all these algorithms remains roughly the same though:

   1. Receive a list of unsorted integers as input.
   2. Based on the size of the input list, choose one of the sorting algorithms (bubble sort, insertion sort, selection sort, merge sort, or quick sort).
   3. Implement the chosen sorting algorithm:
      - For smaller lists: a. In bubble sort, compare adjacent values and swap them if they are in the wrong order. Repeat this process for the entire list until no further swaps are required. b. In insertion sort, compare elements and expand the sorted portion of the list as you sort values. c. In selection sort, find the smallest element in the unsorted part and swap it with the first element.
      - For larger lists: a. In merge sort, divide the list into halves, recursively sort each half, and merge them into a single sorted list. b. In quick sort, choose a pivot element, divide the list into sublists, sort the sublists recursively, and concatenate them around the pivot.
   4. Return the sorted list in ascending order.

3. **The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones (e.g., 0, 1, 1, 2, 3, 5, 8, 13, ...). Write an algorithm to calculate the nth Fibonacci number. Your algorithm should be efficient and capable of handling large values of n.**

   To calculate the nth Fibonacci number:
   1. Start with two variables, “a” and “b”, and initialize them to 0 and 1, respectively.
   2. If n is 0 return 0. Else if n is 1, return 1.
   3. For i from 2 to n (inclusive):
      a. Calculate the sum of “a” and “b” and store it in a temporary variable “d”.
      b. Update “a” to be the current value of “b”.
      c. Update “b” to be the current value of “d”.
   4. Once the loop is finished, “b” will contain the nth Fibonacci number.
   5. Return the value of "b" as the nth Fibonacci number.

4. **You are tasked with creating an algorithm for a store's inventory management system. Your algorithm should be able to add and remove items from the inventory, update the quantity of existing items, and generate reports of the items and their quantities.**

   1. Create an empty inventory list to store items and their respective quantities.
   2. Define specific functions for the following actions:
      a. Add item:
         - Accept item name and quantity as input and check if the item already exists in the inventory.
         - If it does, update the quantity of the existing item. If it doesn’t, add the item name and quantity to the inventory list.
      b. Remove item:
         - Accept item name and quantity as input and check if the item already exists in the inventory.
         - If it does, remove the item from the inventory list. If it doesn’t, display an error message to the user.
      c. Update quantity
         - Accept item name and new quantity as input and check if the item already exists in the inventory.
         - If it does, update the quantity of the item with new quantity. If it doesn’t, display an error message to the user.
      d. Generate a report
         - Display a list of all items in the inventory along with their quantities.
   3. Implement a loop that concurrently prompts the user for aforementioned actions, until these choose to exit.
   4. For each action, call the corresponding function to perform the desired operation.