

Question 1: Finding the Shortest Path

Imagine you are developing a GPS navigation system. You are given a map with various locations and the roads connecting them. Your task is to write an algorithm to find the shortest path from one location to another. You can assume that you have a list of locations and the distance between each pair of locations. Your algorithm should output the shortest path and the total distance.

Algorithm:

- Start at the initial location.
- Set the distance to the initial location as 0 and the distance to all other locations as infinity.
- Visit the current location and consider all its unvisited neighbours.
- For each unvisited neighbour, calculate their tentative distance from the start location (the distance to the current location plus the distance from the current location to this neighbour).
- If this tentative distance is less than the currently recorded distance for this neighbour, update the record with this smaller distance.
- Mark the current location as visited. We won't visit it again since we've found the shortest path to it.
- If we've visited all locations or if the smallest tentative distance among the locations, we haven't yet visited is infinity (which means there's no path to the end location), stop. Otherwise, set the unvisited location with the smallest tentative distance as our new current location and go back to step 3.
- Once we've visited all locations or found that there's no path to the end location, we've got our shortest path.

Question 2: Sorting a List of Numbers

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.

Algorithm:

- Start with an array called arr.
- Find the size of the array and store it in a variable called size.
- Begin a loop from $i = 0$ to $\text{size} - 1$.
- Within the above loop, start another loop from $j = 0$ to $\text{size} - 1 - i$.
- Compare $\text{arr}[j]$ with $\text{arr}[j + 1]$. If $\text{arr}[j]$ is greater than $\text{arr}[j + 1]$, then:
- Swap the values of $\text{arr}[j]$ and $\text{arr}[j + 1]$.
- Repeat steps 4 and 5 until the inner loop completes.
- Repeat steps 3 to 6 until the outer loop completes.
- The array arr is now sorted in ascending order.
- Display the sorted array arr.

Question 3: Calculating Fibonacci Numbers

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.

Algorithm:

- Start by declaring three integers a, b, c, n.
- Let a=0, b=1.
- Take value of n from the user.
- Run loop from zero to n.
- If i is 0 or 1.
 - Then print i.
- For i>1;
 - Print (a+b)
 - c=a ,a=b, b=c+b
- Stop.

Question 4: Inventory Management

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. Design an algorithm that efficiently manages the store's inventory based on these requirements.

Algorithm:

- Start by declaring dictionary called dict.
- Now show the menu.
- Using loop while by adding condition (i is not equals to 0).
- If input is equal to 1.
 - Take product name and quantity.
 - If the name is already in dictionary then update or add the quantity.
 - Else add new name and quantity to the dictionary.
- If input is equal to 2.
 - If the name is in the dictionary then remove the quantity from the dictionary.
 - Show dictionary.
- Stop if the input is less than 1.