MODULE 2: ALGORITHMIC THINKING

Session 1: Algorithmic Thinking 1

Problem Identification and Analysis

What is algorithm thinking?

Algorithmic thinking is a step-by-step approach to solving problems by creating a sequence of instructions, or algorithms, that lead to a solution.

It involves breaking down complex tasks into smaller, manageable parts and designing logical steps to accomplish each part efficiently

What is algorithm thinking? Cont.…

Algorithmic thinking is somehow a pool of abilities that are connected to constructing and understanding algorithms:

- The ability to analyze given problems - the ability to specify a problem precisely.
- The ability to find the basic actions that are adequate to the given problem.
- The ability to construct a correct algorithm to a given problem using the basic actions.
- The ability to think about all possible special and normal cases of a problem.
- The ability to improve the efficiency of an algorithm.

Problem Identification and Analysis

Problem Identification and Analysis involves understanding and defining the problem, including identifying the inputs and outputs, and recognizing any constraints. It requires breaking the problem into smaller parts and analyzing its structure to determine the best approach.

Steps in Problem Identification and Analysis

1. Define the Problem

The first step in solving any problem, algorithmic or otherwise, is to define the problem. This involves identifying the problem's inputs, expected outputs, requirements and constraints.

2. Break Down the Problem

Once you understand the problem, the next step is to decompose the problem into smaller, manageable sub-problems.
Create a flowchart or pseudocode to visualize the problem.

3. Design the Algorithm

Decide on the algorithms and data structures to be used. Outline the steps needed to solve each sub-problem

4. Implement the Algorithm

After designing the algorithm, you must implement it in Python. This involves writing Python code that follows the steps outlined in your algorithm.

5. Test the Algorithm

After implementing the algorithm, you should test it to ensure it works correctly. This involves running your Python code and checking that it produces the expected output for various input values.

6. Analyze the Algorithm

You should analyze your algorithm to understand its efficiency. This involves calculating your algorithm's time and space complexity, which are measures of how the algorithm's resource usage scales with the input size.

Example:

Problem: Given a list of numbers, find the two numbers that add up to a specific target.

1. **Define the Problem:**

Input: List of numbers and a target number.
Output: Indices of the two numbers that add up to the target.

2. Break Down the Problem:

Iterate through the list.

For each number, find another number in the list that, when added to the first, equals the target.

3. Design the Algorithm:

Consider cases where no two numbers add up to the target.

Ensure the solution is efficient for large lists.

3. Implement the Algorithm:

```
def two_sum(nums, target):
    num_dict = {}
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_dict:
            return [num_dict[complement], i]
        num_dict[num] = i
    return None

# Example usage
numbers = [2, 7, 11, 15]
target = 9
result = two_sum(numbers, target)
print("Indices of numbers adding up to
target:", result)
```

Explanation:

This function accepts an array of numbers and a target value.
Then through each pair of numbers in the array checks when added sums up to the target value

Activity

You are tasked with using heuristics and different problem-solving approaches to find the shortest route for visiting multiple cities. This activity helps develop intuition for solving optimization problems using heuristic methods.

PAUSE 5 mins

Activity Feedback

- Understand how heuristics provide practical solutions for complex optimization problems.
- Learn to apply different problem-solving approaches and compare their effectiveness.
- Gain experience balancing solution quality with computational efficiency.
- Develop critical thinking and algorithmic intuition in a hands-on environment.

- This marks the end of this lesson.
- In the next lesson we will learn about Problem Solving Approaches and Heuristics

THANK YOU