Algorithms & Scientific Computing

Day 3: February 11

Agenda

- Morning Session (9:00 – 12:00)
    - Algorithm Fundamentals
    - Time and Space Complexity
    - Common Algorithms
    - Data Structures
    - Problem-Solving Strategies
    - Breaking Down Problems
    - Algorithm Design
    - LeetCode Approaches
- Lunch Break (12:00 – 13:30)

Agenda

- Afternoon Session (13:30 – 15:30)
  - NumPy Introduction
  - Array Operations
  - Vectorization
  - Performance Optimization

Algorithm Fundamentals

- What is an Algorithm?
  - A step-by-step procedure for solving a problem.
  - Characteristics: Finite, well-defined, and effective.
- Why Algorithms Matter:
  - Enable automation, problem solving, and data processing.
- Examples:
  - Sorting, searching, and graph traversal algorithms.r

Time and Space Complexity

- Understanding Complexity:
  - Time Complexity: How the running time grows with input size.
  - Space Complexity: How the memory usage grows with input size.
- Big O Notation:
  - $O(1)$, $O(n)$, $O(n^2)$, etc.
- Examples:
  - Linear search: $O(n)$
  - Binary search: $O(\log n)$

Common Algorithms

- Sorting Algorithms:
  - Bubble Sort, Merge Sort, Quick Sort
- Searching Algorithms:
  - Linear Search, Binary Search
- Graph Algorithms:
  - Depth-First Search (DFS), Breadth-First Search (BFS)
- Other Algorithms:
  - Dynamic programming, Greedy algorithms

Data Structures

- Fundamental Data Structures:
  - Arrays, Linked Lists, Stacks, Queues
- Hierarchical Structures:
  - Trees, Graphs
- Use Cases:
  - Choosing the right data structure for efficient algorithm implementation

Problem-Solving Strategies

- Understanding the Problem:
  - Read carefully, identify inputs and outputs
- Choosing an Approach:
  - Brute force vs. optimized algorithms
- Tools & Techniques:
  - Pseudocode, flowcharts, and whiteboard brainstorming

Breaking Down Problems

- Decomposition:
  - Divide a large problem into smaller, manageable sub-problems.
- Modular Design:
  - Build independent modules or functions.
- Examples:
  - Divide and Conquer algorithms

Algorithm Design

- Design Techniques:
  - Recursion, Iteration, Greedy Methods, Dynamic Programming
- Considerations:
  - Correctness, efficiency, and maintainability
- Real-World Example:
  - Designing an algorithm for route planning or scheduling

LeetCode Approaches

- Platforms for Practice:
  - LeetCode, HackerRank, CodeSignal
- Approach Tips:
  - Read problem statements carefully.
  - Start with simple test cases.
  - Optimize iteratively.
- Benefits:
  - Enhances coding skills, algorithmic thinking, and interview preparation(Hopefully not anytime soon!)

Example: Two Sum

- Link
  - Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
  - You may assume that each input would have exactly one solution, and you may not use the same element twice.
  - You can return the answer in any order.
- Brute force solution:
  - $O(n^2)$ time complexity
  - $O(1)$ space complexity
- Best Solution(I could come up with):
  - $O(n)$ time complexity
  - $O(n)$ space complexity

Lunch Break

- Have lunch in building 13, the diner
- Come back by 13:30(1:30PM)

NumPy Introduction

- What is NumPy?
  - A powerful Python library for numerical computing.
- Key Features:
  - Efficient array operations, linear algebra, random number generation.
- Why Use NumPy?
  - Enhances performance in scientific computing and data analysis.

Array Operations

- Creating Arrays:
  - Using np.array(), np.zeros(), np.ones(), etc.
- Basic Operations:
  - Arithmetic, slicing, and reshaping arrays.

Vectorization

- Concept:
  - Performing operations on entire arrays without explicit loops.
- Benefits:
  - Improved performance and cleaner code.
- Numpy is built entirely on the low-level language C!

Performance Optimization

- Optimizing with NumPy:
  - Use vectorized operations and avoid Python loops.
- Memory Management:
  - Understand array broadcasting and in-place operations.
- Example:
  - Compare loop-based vs. vectorized solutions for a computational task.