****Course Code:** DSA101**

****Course Title:** Data Structure and Analysis**

****Pre-requisite:** Basic programming knowledge (e.g., familiarity with at least one programming language like Python or Java). High school algebra.**

****Syllabus Version:** 1.0**

****Total Lecture Hours:** 45 hours**

****Course Objectives:** Upon successful completion of this course, students will be able to:**

* Understand fundamental data structures and their applications.

* Analyze the time and space complexity of algorithms.

* Implement common data structures using a programming language.

* Select appropriate data structures for specific problem-solving scenarios.

* Apply data analysis techniques to solve real-world problems.

****Course Outcomes:** Students will be able to:**

* Design and implement efficient algorithms using various data structures.

* Analyze the performance of algorithms and data structures.

* Critically evaluate different data structures for given tasks.

* Solve problems using appropriate data structures and algorithms.

* Communicate effectively about data structures and algorithms.

Module Structure:

- **Module 1: Introduction to Data Structures and Algorithms - 5 hours**

  - What are Data Structures?

  - What are Algorithms?

  - Algorithm Analysis: Big O Notation (Introduction)

  - Time and Space Complexity

  - Introduction to Python for Data Structures (if not pre-requisite)

- **Module 2: Arrays and Linked Lists - 6 hours**
  - Arrays: Declaration, Initialization, Operations
  - Static vs. Dynamic Arrays
  - Singly Linked Lists: Creation, Insertion, Deletion, Traversal
  - Doubly Linked Lists: Creation, Insertion, Deletion, Traversal
  - Circular Linked Lists
  - Applications of Linked Lists

- **Module 3: Stacks and Queues - 5 hours**
  - Stacks: LIFO principle, Operations (push, pop, peek), Applications (e.g., function calls, undo/redo)
  - Queues: FIFO principle, Operations (enqueue, dequeue), Applications (e.g., buffering, task scheduling)
  - Priority Queues
  - Implementation of Stacks and Queues using Arrays and Linked Lists

- **Module 4: Trees - 7 hours**
  - Introduction to Trees: Terminology (root, node, leaf, parent, child, sibling)
  - Binary Trees: Properties, Traversals (inorder, preorder, postorder), Implementation
  - Binary Search Trees (BSTs): Search, Insertion, Deletion, Operations
  - Balanced Trees (brief overview AVL trees, Red-Black trees)
  - Tree Applications (e.g., file systems, decision trees)

- **Module 5: Graphs - 8 hours**
  - Introduction to Graphs: Terminology (vertices, edges, directed/undirected graphs)

    - Graph Representations (adjacency matrix, adjacency list)
    - Graph Traversal Algorithms: Breadth-First Search (BFS), Depth-First Search (DFS)
    - Shortest Path Algorithms: Dijkstra's Algorithm (brief overview)
    - Minimum Spanning Trees: Prim's Algorithm (brief overview)
    - Graph Applications (e.g., social networks, route planning)

- **Module 6: Hash Tables - 6 hours**
    - Introduction to Hashing
    - Hash Functions
    - Collision Handling Techniques (separate chaining, open addressing)
    - Hash Table Implementations
    - Applications of Hash Tables (e.g., dictionaries, symbol tables)

- **Module 7: Algorithm Analysis and Design - 8 hours**
    - Big O Notation: Detailed analysis
    - Time and Space Complexity Analysis of various algorithms and data structures covered in previous modules.
    - Algorithm Design Techniques: Divide and Conquer, Greedy Algorithms, Dynamic Programming (introductory concepts)
    - Case Studies: Applying data structures and algorithms to solve real-world problems.

****Textbooks:****

* [Insert relevant textbook here, including author and publication details] Example: "Data Structures and Algorithm Analysis in C++" by Mark Allen Weiss

****Reference Books:****

* [Insert relevant reference books here, including author and publication details] Example:

"Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein

Note: The time allocation for each module is a suggestion and can be adjusted based on the specific needs of the course and the students' pace of learning. The "brief overview" notes indicate topics that might be covered less extensively at a beginner level. Practical exercises, assignments, and quizzes should be integrated throughout the course to reinforce learning.