📘**Data Structures & Algorithms (DSA) Notes – AetherCode (Notion Template)**

🖥️Cover Page

Course Title: Data Structures & Algorithms – Build Logic, Solve Problems\ **Prepared By:** AetherCode Team\ **Website:** www.aethercode.com

> *"Code. Notes. Clarity."*

🕐Table of Contents

Introduction

DSA is the foundation of problem-solving in computer science. Efficient use of memory and computation is key.

📏Complexity Analysis

- Time & Space complexity
- Big-O, Big-Theta, Big-Omega
- Example:

```
for(int i = 0; i < n; i++) {
  cout << i; // O(n)
}
```

📦Arrays

- Static size, index-based access
- Operations: insertion, deletion, traversal
- 1D, 2D, Multidimensional

🔗Linked Lists

- Singly, Doubly, Circular
- Dynamic memory allocation

```
struct Node {
  int data;
  Node* next;
};
```

🚽Stacks & Queues

- Stack: LIFO (push, pop, top)
- Queue: FIFO (enqueue, dequeue)
- Applications in recursion, OS scheduling

🗜️Trees

- Binary Tree, BST, AVL
- Traversals: inorder, preorder, postorder
- Height, depth, balanced trees

🩲Graphs

- Directed/Undirected
- Representations: adjacency matrix/list
- Traversal: BFS, DFS

🔍Searching Algorithms

- Linear Search
- Binary Search (O(log n))

🔮 Sorting Algorithms

- Bubble, Selection, Insertion, Merge, Quick Sort
- Time complexity comparisons

🔦 Recursion

- Base case & recursive case
- Stack memory behavior

```
int fact(int n) {
  if(n == 0) return 1;
  return n * fact(n - 1);
}
```

🧰 Hashing

- Hash functions, collision handling
- Applications: password storage, indexing

💸 Greedy Algorithms

- Huffman Coding, Kruskal's MST
- Makes locally optimal choices

🔧 Dynamic Programming

- Overlapping subproblems
- Memoization vs Tabulation
- Examples: Fibonacci, Knapsack

🕐 Backtracking

- Try all possibilities, revert if needed
- Examples: N-Queens, Sudoku

🔗 Summary

Understanding DSA leads to efficient and optimized problem solving. It is the core of coding interviews and logic building.

Next: Explore Computer Organization & Operating System fundamentals.