JOB SHARATHI | KNOWLEDGE HUB

DATA STRUCTURES AND ALGORITHMS

Complete Study Notes for Beginners, Students and Technical Interview Preparation

LEARN SMARTER. PREPARE BETTER. BUILD YOUR FUTURE.

A structured introduction to core data structures, searching, sorting, complexity analysis and interview fundamentals.

About These Notes

These notes introduce the foundational concepts of Data Structures and Algorithms (DSA). They are organized for learners who want a clear starting point for computer science studies, programming practice and technical interview preparation.

Learning objective: Understand how data is organized, how common structures work, and how algorithm efficiency is described.

Contents

- 1. What is Data Structure?
- 2. Types of Data Structures
- 3. Array
- 4. Linked List
- 5. Stack
- 6. Queue
- 7. Tree Data Structure
- 8. Graph Data Structure
- 9. Searching Algorithms
- 10. Sorting Algorithms
- 11. Time and Space Complexity
- 12. Important DSA Interview Questions
- 13. Conclusion

1. What is Data Structure?

A **Data Structure** is a way of organizing, storing and managing data in a computer so that it can be accessed and modified efficiently.

> **Example:** A list of student names, employee records, product details and bank transactions can be stored using different data structures.

Why are Data Structures important?

- Efficient data storage and management.
- Faster searching and sorting.
- Better memory utilization.
- Improved program performance.
- Helps solve complex programming problems.

2. Types of Data Structures

Data structures are broadly classified into two categories:

A. Linear Data Structures

In linear data structures, elements are arranged sequentially.

- Array
- Linked List
- Stack
- Queue

B. Non-Linear Data Structures

In non-linear data structures, elements are organized in hierarchical or interconnected forms.

- Tree
- Graph
- Heap
- Trie

3. Array

An **Array** is a collection of elements stored in contiguous memory locations. Elements are generally accessed using an index.

Example: `int numbers[5] = {10, 20, 30, 40, 50};`

Index	Value
0	10
1	20
2	30
3	40

Index	Value
4	50

Advantages

- Fast access using an index.
- Easy to traverse.
- Simple data organization.

Disadvantages

- Fixed size in many languages.
- Insertion and deletion can be expensive.

4. Linked List

A **Linked List** is a linear data structure where elements are stored in nodes. Each node contains data and a reference to another node.

Types of Linked Lists

- Singly Linked List
- Doubly Linked List
- Circular Linked List

Advantages

- Dynamic size.
- Easy insertion and deletion when the position is known.

Disadvantages

- Extra memory required for pointers.
- No direct index-based access.

5. Stack

A **Stack** is a linear data structure that follows the **LIFO (Last In, First Out)** principle. The last element inserted is the first element removed.

Main Operations

- Push: Insert an element.
- Pop: Remove the top element.
- Peek: View the top element.
- IsEmpty: Check whether the stack is empty.

Real-life example: A stack of plates.

Applications

- Undo and redo operations.
- Function calls.

- Expression evaluation.
- Browser history.

6. Queue

A **Queue** is a linear data structure that follows the **FIFO (First In, First Out)** principle. The first element inserted is the first element removed.

Main Operations

- Enqueue: Insert an element.
- Dequeue: Remove an element.
- Front: View the first element.
- Rear: View the last element.

Types of Queue

- Simple Queue
- Circular Queue
- Priority Queue
- Double-Ended Queue (Deque)

> **Real-life example:** People waiting in a ticket counter line.

7. Tree Data Structure

A **Tree** is a non-linear hierarchical data structure consisting of nodes connected by edges.

Important terms

- Root: The topmost node.
- Parent: A node that has child nodes.
- Child: A node connected below another node.
- Leaf: A node without children.
- Height: The longest downward path from a node to a leaf.

Types of Trees

- Binary Tree
- Binary Search Tree (BST)
- AVL Tree
- Heap
- B-Tree

Applications: File systems, databases and hierarchical data management.

8. Graph Data Structure

A **Graph** consists of vertices (nodes) and edges (connections).

Types of Graphs

- Directed Graph
- Undirected Graph
- Weighted Graph
- Unweighted Graph
- Cyclic Graph
- Acyclic Graph

Applications

- Social networks.
- Google Maps and route planning.
- Computer networks.
- Recommendation systems.

9. Searching Algorithms

Searching algorithms help find a specific element in a collection of data.

A. Linear Search

Linear Search checks elements one by one until the required element is found or the collection ends.

Example: Searching for 30 in [10, 20, 30, 40, 50].

Case	Time Complexity
Best	O(1)
Average	O(n)
Worst	O(n)

B. Binary Search

Binary Search repeatedly divides a sorted search range into two halves.

> **Important:** Binary Search requires sorted data.

Case	Time Complexity
Best	O(1)
Average	O(log n)
Worst	O(log n)

10. Sorting Algorithms

Sorting means arranging data in a particular order, such as ascending or descending.

Common sorting algorithms

- Bubble Sort
- Selection Sort

- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort

Sorting Complexity Comparison

Algorithm	Average Time
Bubble Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Insertion Sort	$O(n^2)$
Merge Sort	O(n log n)
Quick Sort	O(n log n)
Heap Sort	O(n log n)

11. Time and Space Complexity

Complexity analysis helps measure the efficiency of an algorithm.

Big O Notation

Big O describes how an algorithm's resource requirements grow as input size increases, commonly used for upper-bound analysis.

Complexity	Meaning
O(1)	Constant
O(log n)	Logarithmic
O(n)	Linear
O(n log n)	Linearithmic
$O(n^2)$	Quadratic
O(2■)	Exponential

Space Complexity

Space Complexity measures the additional memory required by an algorithm as input size grows.

12. Important DSA Interview Questions

Q1. What is Data Structure?

A way of organizing and storing data for efficient access and modification.

Q2. What is an Array?

A collection of elements stored in contiguous memory locations and accessed using indices.

Q3. What is a Stack?

A linear data structure that follows Last In, First Out (LIFO).

Q4. What is a Queue?

A linear data structure that follows First In, First Out (FIFO).

Q5. What is a Linked List?

A collection of nodes connected through references or pointers.

Q6. What is Binary Search?

A searching algorithm that repeatedly divides a sorted search range into halves.

Q7. What is the difference between Stack and Queue?

Stack follows LIFO, while Queue follows FIFO.

Q8. What is Big O Notation?

A mathematical notation used to describe how an algorithm's resource usage grows with input size.

Q9. What is Recursion?

A programming technique in which a function calls itself to solve a problem.

Q10. What is a Graph?

A non-linear data structure consisting of vertices and edges.

13. Conclusion

Data Structures and Algorithms are fundamental concepts in computer science and software development. Learning DSA helps students improve logical thinking, solve programming problems and prepare for technical interviews.

> **Recommended Learning Path:** Arrays → Strings → Linked Lists → Stack → Queue → Searching → Sorting → Trees → Graphs → Dynamic Programming.

JOB SHARATHI – KNOWLEDGE HUB

Learn Smarter. Prepare Better. Build Your Future.

Website: https://todaybihar.in/