**List all the data structures in dsa**

There are various data structures in DSA, broadly categorized into Linear and Non-Linear data structures:

1. Linear Data Structures

These data structures store data sequentially, and elements are accessed in a linear order.

* Array – Fixed-size data structure that stores elements of the same type.
* Linked List – A dynamic data structure consisting of nodes where each node points to the next one. (Types: Singly, Doubly, and Circular Linked List)
* Stack – Follows LIFO (Last In, First Out) principle; used in recursion, undo operations, and expression evaluation.
* Queue – Follows FIFO (First In, First Out) principle; used in scheduling and buffering. (Types: Simple Queue, Circular Queue, Priority Queue, Deque)

2. Non-Linear Data Structures

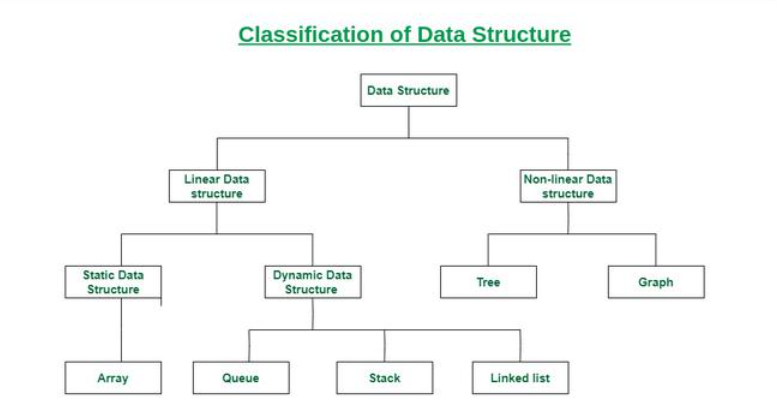
These data structures do not store elements in a sequential manner.

* Tree – A hierarchical structure where nodes have parent-child relationships. (Types: Binary Tree, Binary Search Tree, AVL Tree, B-Trees, etc.)
* Graph – A set of nodes (vertices) connected by edges. (Types: Directed, Undirected, Weighted, Unweighted, etc.)
* Heap – A special tree-based structure that satisfies the heap property. (Types: Min-Heap, Max-Heap)
* Trie (Prefix Tree) – A tree used for searching words efficiently, mainly used in dictionaries and autocomplete features.

3. Hashing-based Data Structure

* Hash Table / Hash Map – A key-value pair data structure that provides fast lookups using a hash function.

Each of these data structures has specific use cases, time complexity, and advantages depending on the problem statement.



* 1. **Array**
  2. **Matrix/Grid**
  3. **String**
  4. **Stack**
  5. **Queue**
  6. **Linked list**
  7. **Hash**
  8. **Tree**
  9. **Binary Tree**
  10. **Binary Search tree**
  11. **Heap**
  12. **Graph**
  13. **Advanced data structures**

**Link -** [**https://www.geeksforgeeks.org/data-structures/**](https://www.geeksforgeeks.org/data-structures/)

**list all the algorithms in dsa –**

There are several important algorithms in Data Structures and Algorithms (DSA), categorized based on their functionality.

1. Searching Algorithms

These are used to find an element in a data structure.

* Linear Search – Iterates through each element; O(n) time complexity.
* Binary Search – Efficient search for sorted arrays; O(log n) time complexity.
* Interpolation Search – Improved binary search for uniformly distributed data; O(log log n) average case.

2. Sorting Algorithms

These are used to arrange elements in a specific order (ascending/descending).

* Bubble Sort – Repeated swapping; O(n²).
* Selection Sort – Finds the smallest element and places it at the correct position; O(n²).
* Insertion Sort – Builds a sorted array one element at a time; O(n²).
* Merge Sort – Divide-and-conquer sorting; O(n log n).
* Quick Sort – Selects a pivot and partitions elements; O(n log n) average case.
* Heap Sort – Uses a heap data structure; O(n log n).
* Radix Sort – Digit-by-digit sorting; O(nk) where k is the number of digits.

3. Recursion & Backtracking Algorithms

Used for problems involving decision trees and exploration.

* Tower of Hanoi – Recursive problem-solving.
* N-Queens Problem – Places queens on a chessboard using backtracking.
* Sudoku Solver – Fills a Sudoku grid using backtracking.
* Rat in a Maze – Pathfinding using recursion.

4. Divide and Conquer Algorithms

These break down a problem into subproblems and solve recursively.

* Merge Sort – Sorts by dividing the array.
* Quick Sort – Uses partitioning to sort.
* Binary Search – Searches by dividing.
* Strassen’s Matrix Multiplication – Efficient matrix multiplication.

5. Dynamic Programming (DP) Algorithms

Solves problems by storing solutions to subproblems (memoization/tabulation).

* Fibonacci Sequence – Recursion with memoization.
* Knapsack Problem (0/1 and Fractional) – Optimizes weight in a bag.
* Longest Common Subsequence (LCS) – Finds the longest sequence in two strings.
* Longest Increasing Subsequence (LIS) – Finds increasing sequence in an array.
* Matrix Chain Multiplication – Optimizes multiplication order.
* Coin Change Problem – Finds the minimum number of coins for an amount.

6. Greedy Algorithms

Builds a solution step-by-step, choosing the best option at each step.

* Huffman Coding – Data compression using a priority queue.
* Kruskal’s Algorithm – Minimum spanning tree using union-find.
* Prim’s Algorithm – Minimum spanning tree using priority queues.
* Dijkstra’s Algorithm – Finds the shortest path from a source.
* Fractional Knapsack – Optimizes value per weight unit.

7. Graph Algorithms

Used for solving graph-based problems.

* Depth-First Search (DFS) – Explores depth-first; O(V+E).
* Breadth-First Search (BFS) – Explores breadth-first; O(V+E).
* Dijkstra’s Algorithm – Shortest path in a weighted graph.
* Floyd-Warshall Algorithm – All-pairs shortest path.
* Bellman-Ford Algorithm – Shortest path with negative weights.
* Kruskal’s Algorithm – Minimum spanning tree.
* Prim’s Algorithm – Minimum spanning tree.
* Topological Sorting – Used in directed acyclic graphs (DAGs).

8. String Algorithms

Used for pattern matching and string manipulation.

* KMP Algorithm (Knuth-Morris-Pratt) – Efficient pattern matching; O(n).
* Rabin-Karp Algorithm – Uses hashing for pattern searching.
* Z Algorithm – Finds substrings efficiently.
* Suffix Array & Suffix Tree – Optimized substring search.

9. Bit Manipulation Algorithms

Efficient operations using bitwise operators.

* Check if a number is power of 2 – Bitwise AND.
* Count set bits in an integer – Brian Kernighan’s Algorithm.
* XOR-based problems – Finding missing numbers in an array.

10. Mathematical Algorithms

Used in numerical and combinatorial problems.

* GCD (Greatest Common Divisor) – Euclidean Algorithm
* LCM (Least Common Multiple)
* Sieve of Eratosthenes – Finds prime numbers efficiently.
* Exponentiation by Squaring – Fast power calculation.
* Modular Exponentiation – Used in cryptography.