
Ashish Sahu
@ashsau

Data Structure

CheatSheet

Tree — Hierarchical structure with nodes.

Stack — LIFO (Last In, First Out) collection.

Array — Fixed-size, indexed collection.

Queue — FIFO (First In, First Out) collection.

Linked List — Dynamic, node-based sequence.

NEXT ➡


Ashish Sahu
@ashsau

Array

A contiguous block of memory consisting of elements of the same type, accessed via indices.

Example:

- int array[5] = {1, 2, 3, 4, 5};

Use Case: Used for situations requiring constant-time access to elements, such as storing a list of known size.

Advantages:

- Fast access to elements via index.
- Memory-efficient for fixed-size data.

Disadvantages:

- Fixed size; can't dynamically grow.
- Insertion and deletion can be costly.

NEXT

Ashish Sahu
@ashsau

Linked List

A sequence of nodes where each node contains data and a reference to the next node.

Example:

```
class Node {
    int data;
    Node next;
}
```

Use Case: Suitable for scenarios where dynamic memory allocation and frequent insertion/deletion are required, like implementing a music playlist.

Advantages:

- Dynamic size.
- Efficient insertion and deletion.

Disadvantages:

- Higher memory usage due to pointers.
- Sequential access; slower than arrays for indexing.

NEXT


Ashish Sahu
@ashsau

Tree

A hierarchical data structure with nodes connected by edges, with one node designated as the root.

Example:

```
class TreeNode {
    int data;
    TreeNode left, right;
}
```

Use Case: Used in hierarchical data representation like file systems, databases, and organizing information with parent-child relationships.

Advantages:

- Reflects hierarchical relationships naturally.
- Efficient searching, insertion, and deletion (in balanced trees).

Disadvantages:

- Can become unbalanced, affecting performance.
- More complex to implement and maintain.

NEXT

Ashish Sahu
@ashsau

Queue

A collection that follows the First In, First Out (FIFO) principle.

Example:

```
Queue<Integer> queue = new LinkedList<>();
queue.add(1);
queue.add(2);
queue.remove(); // 1
```

Use Case: Ideal for scheduling tasks, managing resources in operating systems, and breadth-first search (BFS) in graph algorithms.

Advantages:

- Simple to implement.
- Ensures order of processing.

Disadvantages:

- Fixed size in some implementations.
- Sequential access; can't randomly access elements.

NEXT


Ashish Sahu
@ashsau

Stack

A collection that follows the Last In, First Out (LIFO) principle.

Example:

```
Stack<Integer> stack = new Stack<>();
stack.push(1);
stack.push(2);
stack.pop(); // 2
```

Use Case: Used in function call management, expression evaluation, and backtracking algorithms.

Advantages:

- Simple to implement.
- Efficient for managing function calls and recursive algorithms.

Disadvantages:

- Fixed size in some implementations.
- Sequential access; can't randomly access elements.