Data Structure Algorithm	Operation	Time Complexity	Description and Common Usage
Arrays	Access (by index)	O(1)	Directly access elements by index.
	Search (unsorted)	O(N)	Linear search through an unsorted array.
	Search (sorted)	O(log N)	Binary search in a sorted array.
	Insert/Delete (end)	O(1)	Inserting or deleting elements at the end.
	Insert/Delete (middle)	O(N)	Inserting or deleting elements in the middle.
Linked Lists	Access (singly linked)	O(N)	Traversing linked list sequentially.
	Access (doubly linked)	O(N)	Traversing doubly linked list sequentially.
	Search	O(N)	Searching for an element.
	Insert/Delete (beginning)	O(1)	Efficient insertion or deletion at the beginning.

	Insert/Delete (middle, known location)	O(1)	Efficient insertion or deletion with a known location.
	Insert/Delete (end, with tail reference)	O(1)	Efficient insertion or deletion at the end.

Data Structure Algorithm	Operation	Time Complexity	Description and Common Usage
Stacks	Push (Insert)	O(1)	Inserting an element at the top.
	Pop (Remove)	O(1)	Removing an element from the top.
	Peek (View)	O(1)	Viewing the top element without removal.
Queues	Enqueue (Insert)	O(1)	Inserting an element at the end.
	Dequeue (Remove)	O(1)	Removing an element from the front.
	Peek (View)	O(1)	Viewing the front element without removal.
Hash Tables	Insert	O(1) avg., O(N) worst-case	Efficient insertion with hashing
	Search	O(1) avg., O(N) worst-case	Efficient search with hashing.
	Delete	O(1) avg., O(N) worst-case	Efficient deletion with hashing.
BST	Insert (avg.)	O(log N)	Efficient insertion in balanced trees.
	Search (avg.)	O(log N)	Efficient search in balanced trees.

	Delete (avg.)	O(log N)	Efficient deletion in balanced trees.

Data Structure Algorithm	Operation	Time Complexity	Description and Common Usage
Heaps	Insert	O(log N)	Efficient insertion with a heap data structure.
	Extract Min/Max	O(log N)	Efficient extraction of minimum/maximum element.
	Peek Min/Max	O(1)	Viewing the minimum/maximum element.
Graphs	BFS (Traversal)	O(V + E)	Breadth-First Search traversal.
	DFS (Traversal)	O(V + E)	Depth-First Search traversal.
Priority Queues	Insert	O(log N)	Efficient insertion in a priority queue.
	Retrive Min/Max	O(log N)	Efficient extraction of minimum/maximum element.
	Peek Min/Max	O(1)	Viewing the minimum/maximum element.
Trie (Prefix Tree)	Insert	O(m)	Insertion of elements in a Trie.
	Search	O(m)	Searching for elements in a Trie.
	Delete	O(m)	Deletion of elements in a Trie.

Data Structure Algorithm	Operation	Time Complexity	Description and Common Usage
Hash Map	Insert	O(1) avg., O(N) worst-case	Efficient key-value pair insertion.
	Search	O(1) avg., O(N) worst-case	Efficient key-based search.
	Delete	O(1) avg., O(N) worst-case	Efficient key-based deletion.
Hash Set	Insert	O(1) avg., O(N) worst-case	Efficient element insertion.
	Search	O(1) avg., O(N) worst-case	Efficient element-based search.
	Delete	O(1) avg., O(N) worst-case	Efficient element-based deletion.
Tree Map	Insert	O(log N)	Efficient key-value pair insertion.
	Search	O(log N)	Efficient key-based search.
	Delete	O(log N)	Efficient key-based deletion.
Tree Set	Insert	O(log N)	Efficient element insertion.

	Search	O(log N)	Efficient element-based search.
	Delete	O(log N)	Efficient element-based deletion.

Data Structure Algorithm	Operation	Time Complexity	Description and Common Usage
O(N Log N)	• Merge Sort • Quick Sort • Heap Sort	O(N log N)	Sorting algorithms that divide data into smaller parts, sort them, and then merge.
			Divide and conquer: Divides input into smaller parts.
			Sorting: Sorts the smaller parts.
			Merge: Combines the sorted parts.
			Commonly used for sorting a list of items efficiently.

O(N^2)	Bubble Sort	O(N^2)	A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. It is often used for educational purposes and small datasets.
	Selection Sort		A simple sorting algorithm that sorts an array by repeatedly finding the minimum element and moving it to the beginning of the array. It has limited practical use due to its inefficiency on larger datasets.
	Insertion Sort		A simple sorting algorithm that builds the final sorted array one item at a time. It is efficient for small datasets or nearly sorted data.

	Quadratic Operations		Algorithms involving nested loops where comparisons or operations are performed for all pairs of elements. Common in educational contexts to illustrate inefficient algorithms.