|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Data Structure | Access | Insert | Delete | Use Case | | Array | O(1) | O(n) | O(n) | Fast access | | Linked List | O(n) | O(1) | O(1) | Fast insertion/deletion | | Stack | O(1) | O(1) | O(1) | Backtracking, recursion | | Queue | O(1) | O(1) | O(1) | BFS, resource scheduling | | HashMap | O(1)\* | O(1)\* | O(1)\* | Fast lookup | | Set | O(1)\* | O(1)\* | O(1)\* | No duplicates, membership check | | Tree (BST) | O(log n) | O(log n) | O(log n) | Sorted data, fast search | | Heap | N/A | O(log n) | O(log n) | Priority queue | | Trie | O(m) | O(m) | O(m) | Word storage, prefix search (m=word length) | | Graph | O(1) per edge | Varies | Varies | Route finding, dependencies | |

**1. Arrays**

* **Definition:** Fixed-size, contiguous memory blocks that store elements of the same type.
* **Operations:** Access (O(1)), Insert/Delete (O(n))
* **Example:** int[] nums = {1, 2, 3, 4};
* **Use Cases:** Fast indexed access, storing collections of data.

**2. Linked List**

* **Definition:** A series of nodes where each node contains data and a reference to the next (or previous) node.
* **Types:** Singly, Doubly, Circular
* **Operations:** Insert/Delete (O(1) at head), Search (O(n))
* **Use Cases:** Dynamic memory allocation, efficient insertions/deletions.

**🔷 3. Stack (LIFO)**

* **Definition:** A linear structure that follows "Last In First Out".
* **Operations:** push(), pop(), peek() (all O(1))
* **Use Cases:** Recursion, undo functionality, expression evaluation.

**🔷 4. Queue (FIFO)**

* **Definition:** A linear structure that follows "First In First Out".
* **Operations:** enqueue(), dequeue()
* **Types:** Simple Queue, Circular Queue, Priority Queue, Deque
* **Use Cases:** Scheduling, BFS, messaging systems.

**🔷 5. HashMap / HashTable**

* **Definition:** Stores key-value pairs using a hash function for fast lookup.
* **Operations:** Insert, Delete, Search (Avg O(1), Worst O(n))
* **Use Cases:** Caching, indexing, frequency counting.
* **Java Example:** HashMap<String, Integer> map = new HashMap<>();

**🔷 6. Set**

* **Definition:** A collection of unique elements (no duplicates).
* **Operations:** Add, Remove, Search (O(1) avg)
* **Java Example:** HashSet<Integer> set = new HashSet<>();

**🔷 7. Trees**

* **Definition:** A hierarchical data structure with nodes connected by edges.
* **Types:** Binary Tree, Binary Search Tree (BST), AVL Tree, Heap, Trie
* **Use Cases:** Searching (BST), priority queues (Heap), dictionaries (Trie)

**🔷 8. Graphs**

* **Definition:** A set of nodes (vertices) connected by edges.
* **Types:** Directed/Undirected, Weighted/Unweighted
* **Representations:** Adjacency List, Matrix
* **Use Cases:** Social networks, navigation (maps), dependency resolution.

**🔷 9. Heaps (Priority Queue)**

* **Definition:** A binary tree where the parent is either ≥ (max-heap) or ≤ (min-heap) than children.
* **Operations:** Insert, Delete (O(log n))
* **Use Cases:** Scheduling, finding k largest/smallest elements.