Linear Data Structures

- Arrays
  - **Access**: Time O(1), Space O(1)
  - **Search**: Time O(n), Space O(1)
  - **Insertion / Deletion (at end)**: Time O(1) (amortized for dynamic), Space O(1)
  - **Insertion / Deletion (at middle)**: Time O(n), Space O(1)
- Linked Lists (Singly, Doubly, Circular)
  - **Access / Search**: Time O(n), Space O(1)
  - **Insertion / Deletion (at beginning)**: Time O(1), Space O(1)
  - **Insertion / Deletion (at end)**: Time O(1) (with tail pointer), Space O(1)
- Stacks (Array/List implementation)
  - **Push / Pop / Peek**: Time O(1), Space O(1)
- Queues (Array/List implementation)
  - **Enqueue / Dequeue**: Time O(1), Space O(1)
- Hash Table / Hash Map / Hash Set
  - **Insert / Delete / Search**: Average Time O(1), Worst Time O(n). Space O(n) for storage.

Trees

- Binary Tree Traversal (Inorder, Preorder, Postorder)
    - **Time**: O(n), **Space**: O(h) where h is the height of the tree (O(n) in worst-case, O(logn) in best-case).
- Binary Search Tree (BST)
    - **Search / Insert / Delete**: Average Time O(logn), Worst Time O(n).
- Balanced BSTs (AVL, Red-Black)
    - **Search / Insert / Delete**: Time O(logn), Space O(logn).
- Heap (Min-Heap, Max-Heap)
    - **Insert / Delete**: Time O(logn)
    - **Get Min/Max**: Time O(1)
    - **Build Heap**: Time O(n)
- **Trie** (where M is the length of the key)
    - **Insert / Search / Delete**: Time O(M), Space O(N · M) where N is the number of keys.

Graphs

Let V be the number of vertices and E be the number of edges.

- **DFS / BFS**:

  Time O(V+E), Space O(V).

- **Dijkstra's Algorithm**:

  Time O(ElogV) with a binary heap. Space O(V).

- **Bellman-Ford Algorithm**:

  Time O(V · E), Space O(V).

- **Floyd-Warshall Algorithm**:

  Time O(V3), Space O(V2).

- **Kruskal's Algorithm**:

  Time O(ElogE), Space O(V+E).

- **Prim's Algorithm**:

  Time O(ElogV) with a binary heap. Space O(V).

- **Topological Sort**:

  Time O(V+E), Space O(V).

- **Strongly Connected Components (Tarjan's, Kosaraju's)**:

  Time O(V+E), Space O(V).

Sorting & Searching Algorithms

- **Linear Search**:

  Time O(n), Space O(1).

- **Binary Search**:

  Time O(logn), Space O(1).

- **Merge Sort**:

  Time O(nlogn), Space O(n).

- **Quick Sort**:

  Average Time O(nlogn), Worst Time O(n2). Space O(logn).

- **Heap Sort**:

  Time O(nlogn), Space O(1).

- **Bubble / Selection / Insertion Sort**:

  Time O(n2), Space O(1).

- **Counting Sort**:

  Time O(n+k), Space O(n+k) where k is the range of elements.

- **Radix Sort**:

  Time O(d · (n+k)), Space O(n+k) where d is the number of digits.

String Algorithms

Let n be the length of the text and m be the length of the pattern.

- **Naive String Matching**: Time O(n⋅m), Space O(1).
- **Knuth-Morris-Pratt (KMP)**: Time O(n+m), Space O(m).
- **Rabin-Karp**: Average Time O(n+m), Worst Time O(n⋅m). Space O(1).
- **Longest Common Subsequence (LCS)**: Time O(n⋅m), Space O(n⋅m).
- **Edit Distance**: Time O(n⋅m), Space O(n⋅m).

Dynamic Programming & Recursion Examples

- **Fibonacci (Memoized/Tabulation)**: Time O(n), Space O(n).
- **0/1 Knapsack**: Time O(N⋅W), Space O(W) where N is items and W is capacity.
- **Longest Increasing Subsequence (LIS)**: Time O(nlogn), Space O(n).
- **Matrix Chain Multiplication**: Time O(n3), Space O(n2).

Greedy Algorithms Examples

- **Activity Selection**: Time O(nlogn) (if not sorted), Space O(1).
- **Fractional Knapsack**: Time O(nlogn), Space O(1).
- **Huffman Coding**: Time O(nlogn), Space O(n).

Divide & Conquer Examples

- **Closest Pair of Points**: Time O(nlogn), Space O(n).
- **Strassen's Matrix Multiplication**: Time O(nlog27), Space O(n2).

Advanced & Specialized

- **Union-Find / Disjoint Set Union (DSU)**: Time $O(\alpha(n))$ (Amortized, nearly constant), Space $O(n)$.