Algorithm Time & Space Complexity Cheat Sheet

1. Data Structures

Array (static): Access O(1), Search O(n), Insert O(n), Delete O(n), Space O(n)

List (dynamic): Access O(1), Search O(n), Insert O(1), Delete O(n), Space O(n)

HashMap: Search/Insert/Delete O(1) avg, O(n) worst, Space O(n)

Stack/Queue: Insert/Delete O(1), Search O(n), Space O(n)

BST (Balanced): Search/Insert/Delete O(log n), Space O(n)

Heap: Insert/Delete O(log n), Peek O(1), Space O(n)

2. Sorting Algorithms

Bubble Sort: Best O(n), Avg/Worst O(n^2), Space O(1)

Insertion Sort: Best O(n), Avg/Worst O(n^2), Space O(1)

Merge Sort: O(n log n) all cases, Space O(n)

Quick Sort: Avg O(n log n), Worst O(n^2), Space O(log n)

Heap Sort: O(n log n) all cases, Space O(1)

Counting Sort: O(n + k), Space O(k)

Radix Sort: O(nk), Space O(n + k)

3. Searching Algorithms

Linear Search: O(n) time, O(1) space

Binary Search: O(log n) time, O(1) space

Hashing Search: O(1) avg, O(n) worst, Space O(n)

4. Recursion & DP

Recursive Fibonacci: O(2^n) time, O(n) space

DP Fibonacci: O(n) time, O(n) or O(1) space

0/1 Knapsack: O(nW) time, O(nW) space

LCS: O(m×n) time and space

5. Graph Algorithms

Algorithm Time & Space Complexity Cheat Sheet

BFS/DFS: O(V + E) time, O(V) space

Dijkstra: O((V + E) log V) time, O(V) space

Bellman-Ford: O(VE) time

Floyd-Warshall: O(V^3) time, O(V^2) space

Kruskal/Prim: O(E log V) time

6. Tree Algorithms

BST (Balanced): Insert/Search/Delete O(log n)

BST (Unbalanced): O(n)

AVL/Red-Black: O(log n) for all operations

Traversal: O(n) time

7. Heap Operations

Insert/Delete Min-Max: O(log n)

Peek: O(1)

Build Heap: O(n)

8. String Algorithms

Naive Matching: O(nm) time

KMP: O(n + m) time, O(m) space

Rabin-Karp: O(n + m) avg

Trie: O(k) time, O(k·n) space