GUIDE : HOW TO MEASURE TIME COMPLEXITY

This guide should help you quickly analyze time complexity by recognizing common patterns in code! 🚀

1. Loop-Based Patterns

Pattern	Code Example	Time Complexity
Single Loop	for (int i = 0; i < n; i++)	O(n)
Nested Loops	for (int i = 0; i < n; i++) for (int j = 0; j < n; j++)	$O(n^2)$

Triple Nested Loops	for (int i = 0; i < n; i++)
	for (int j = 0; j < n; j++)
	for (int k = 0; k < n; k++)
		$O(n^3)$

**Loop with Increment (i = 2, i += constant)*	for (int i = 1; i < n; i *= 2)	O(log n)
Loop with Decrement (i /= 2)	for (int i = n; i > 0; i /= 2)	O(log n)

2. Recursive Patterns

Pattern	Recurrence Relation	Time Complexity
Linear Recursion	T(n) = T(n-1) + O(1)	O(n)
Binary Recursion	T(n) = 2T(n/2) + O(1)	O(n)
Divide & Conquer (Merge Sort, Quick Sort Worst Case)	T(n) = 2T(n/2) + O(n)	O(n log n)
Exponential Recursion (Fibonacci, Brute Force DFS)	T(n) = T(n-1) + T(n-2)	$O(2^n)$

3. Divide and Conquer Patterns

Algorithm	Recurrence	Complexity
Binary Search	$T(n) = T(n/2) + O(1)$	$O(\log n)$
Merge Sort	$T(n) = 2T(n/2) + O(n)$	$O(n \log n)$
Quick Sort (Best & Avg)	$T(n) = T(n/2) + O(n)$	$O(n \log n)$
Quick Sort (Worst Case - sorted array)	$T(n) = T(n-1) + O(n)$	$O(n^2)$

4. Dynamic Programming Patterns

Pattern	Example	Time Complexity
Memoization (Top-Down Recursion with Cache)	Fibonacci DP	$O(n)$
Bottom-Up Iterative DP	Knapsack, LIS	$O(n^2)$ or $O(n^3)$
Matrix Chain Multiplication	$T(n) = O(n^3)$	$O(n^3)$

5. Graph Algorithms

Algorithm	Complexity
BFS / DFS (Adjacency List)	$O(V + E)$
Dijkstra (Min Heap)	$O((V + E) \log V)$
Bellman-Ford	$O(VE)$
Floyd Warshall (All-Pairs Shortest Path)	$O(V^3)$
Prim’s / Kruskal’s MST	$O(E \log V)$

6. Sorting Algorithms

Algorithm	Best Case	Worst Case
Bubble Sort / Insertion Sort	O(n)	$O(n^2)$
Merge Sort	O(n log n)	O(n log n)
Quick Sort	O(n log n)	$O(n^2)$
Heap Sort	O(n log n)	O(n log n)

7. Logarithmic and Amortized Complexities

Pattern	Example	Complexity
Binary Search	Search in sorted array	O(log n)
Heap Operations (Insert/Delete)	Priority Queue, Dijkstra	O(log n)
Balanced BST (Insertion, Deletion, Search)	AVL, Red-Black Tree	O(log n)
Union-Find (Path Compression & Rank)	DSU operations	$O(\alpha(n))$ (inverse Ackermann)

8. Special Cases

Case	Example
Iterating All Subsets	$O(2^n)$
Iterating All Permutations	O(n!)
Brute Force Checking All Pairs	$O(n^2)$