

TIME and SPACE COMPLEXITY:

Notation	Name	Example	Description
$O(1)$	Constant	Accessing array element	Same time regardless of input size
$O(\log n)$	Logarithmic	Binary search	Divides problem in half each time
$O(n)$	Linear	Loop through array	Grows directly with input size
$O(n \log n)$	Linearithmic	Merge sort, Quick sort	Efficient sorting algorithms
$O(n^2)$	Quadratic	Nested loops	Loop inside a loop
$O(n^3)$	Cubic	Triple nested loops	Three loops nested
$O(2^n)$	Exponential	Fibonacci recursion	Doubles with each input increase
$O(n!)$	Factorial	Generating all permutations	Extremely slow for large inputs

Big-O Time & Space Complexity Cheat Sheet

1 Time Complexity Cheat Sheet

Notation	Name	Typical Example	When You See This
$O(1)$	Constant	Array index access	No loops, fixed operations
$O(\log n)$	Logarithmic	Binary search	Halving input each step
$O(n)$	Linear	Single loop	Iterating once over input
$O(n \log n)$	Linearithmic	Merge sort, Quick sort (avg)	Divide + process
$O(n^2)$	Quadratic	Nested loops	Loop inside loop
$O(n^3)$	Cubic	Triple nested loops	3 levels of iteration
$O(2^n)$	Exponential	Recursive Fibonacci	Try all combinations
$O(n!)$	Factorial	Permutations	Try all orderings

2 Space Complexity Cheat Sheet

Notation	Name	Typical Example	Memory Usage
$O(1)$	Constant Space	Few variables	No extra memory
$O(\log n)$	Logarithmic Space	Recursive binary search	Call stack depth
$O(n)$	Linear Space	Array, list, hashmap	One extra structure
$O(n + m)$	Linear (multi-input)	Graphs	Depends on inputs
$O(n^2)$	Quadratic Space	2D DP table	Matrix storage
$O(n^3)$	Cubic Space	3D DP	Advanced DP
$O(2^n)$	Exponential Space	All subsets	Power set
$O(n!)$	Factorial Space	All permutations	Extremely large

3 Common Algorithms: Time vs Space

Algorithm	Time	Space
Binary Search	$O(\log n)$	$O(1)$
Linear Search	$O(n)$	$O(1)$
Merge Sort	$O(n \log n)$	$O(n)$
Quick Sort (avg)	$O(n \log n)$	$O(\log n)$
Bubble Sort	$O(n^2)$	$O(1)$
BFS / DFS	$O(V + E)$	$O(V)$
HashMap lookup	$O(1)$ avg	$O(n)$

4 Quick Rules

- Loops

- 1 loop $\rightarrow O(n)$
- 2 nested loops $\rightarrow O(n^2)$

- **Recursion**
 - Time = number of calls
 - Space = recursion depth
 - **Sorting**
 - Efficient sorts $\rightarrow O(n \log n)$
 - **In-place algorithm**
 - Space $\rightarrow O(1)$
 - **Graphs**
 - Time $\rightarrow O(V + E)$
 - Space $\rightarrow O(V + E)$
-

5 Golden Line

“Time complexity measures how runtime grows with input size, while space complexity measures the extra memory used excluding the input itself.”
