

Constraints	Worst time complexity	Algorithmic solution	Examples
$n \leq 12$	$O(n!)$	Recursion & Backtracking	Permutation 1....n
$n \leq 25$	$O(2^n)$	Recursion , Backtracking & Bit Manipulation	All subsets of an array of size n
$n \leq 100$	$O(n^4)$	Dynamic Programming	4Sum
$n \leq 500$	$O(n^3)$	Dynamic Programming	All triangles with side length less than n
$n \leq 10^4$	$O(n^2)$	Dynamic Programming, Graphs , Trees	Bubble Sort (Slow comparison-based sorting)
$n \leq 10^6$	$O(n \log n)$	Sorting , Binary Search , Divide and Conquer	Merge Sort (Fast comparison-based sorting)
$n \leq 10^8$	$O(n)$	Mathematical , Greedy	Min and max of element
$n > 10^8$	$O(\log n)$ or $O(1)$	Mathematical, Greedy	Binary Search

max ops = 10^7

Solve LeetCode Questions with This One Pattern (Constraints to Time Complexity)

$n < 20$	$2^n, n!$ e.g. brute force, backtracking
$n < 3000$	n^2 e.g. dynamic programming
$3000 < n < 10^6$	$O(n), O(n \log n)$ e.g. 2 pointers, greedy, heap, sorting
$n > 10^6$	$O(\log n), O(1)$ e.g. binary search, math

Common time complexities

Let n be the main variable in the problem.

- If $n \leq 12$, the time complexity can be $O(n!)$.
- If $n \leq 25$, the time complexity can be $O(2^n)$.
- If $n \leq 100$, the time complexity can be $O(n^4)$.
- If $n \leq 500$, the time complexity can be $O(n^3)$.
- If $n \leq 10^4$, the time complexity can be $O(n^2)$.
- If $n \leq 10^6$, the time complexity can be $O(n \log n)$.
- If $n \leq 10^8$, the time complexity can be $O(n)$.
- If $n > 10^8$, the time complexity can be $O(\log n)$ or $O(1)$.

Examples of each common time complexity

- $O(n!)$ [Factorial time]: Permutations of $1 \dots n$
- $O(2^n)$ [Exponential time]: Exhaust all subsets of an array of size n
- $O(n^3)$ [Cubic time]: Exhaust all triangles with side length less than n
- $O(n^2)$ [Quadratic time]: Slow comparison-based sorting (eg. Bubble Sort, Insertion Sort, Selection Sort)
- $O(n \log n)$ [Linearithmic time]: Fast comparison-based sorting (eg. Merge Sort)
- $O(n)$ [Linear time]: Linear Search (Finding maximum/minimum element in a 1D array), Counting Sort
- $O(\log n)$ [Logarithmic time]: Binary Search, finding GCD (Greatest Common Divisor) using Euclidean Algorithm

- $O(1)$ [Constant time]: Calculation (eg. Solving linear equations in one unknown)