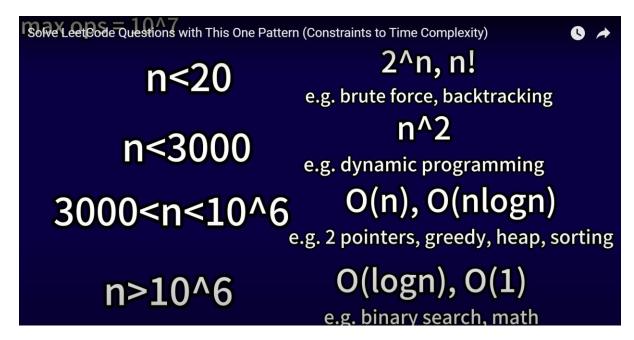
Constraints	Worst time complexity	Algorithmic solution	Examples
<i>n</i> ≤ 12	O(n!)	Recursion & Backtracking	Permutation 1n
n ≤ 25	O(2n)	Recursion , Backtracking & Bit Manipulation	All subsets of an array of size n
<i>n</i> ≤ 100	O(n4)	Dynamic Programming	4Sum
<i>n</i> ≤ 500	O(n3)	Dynamic Programming	All triangles with side length less than n
<i>n</i> ≤ 104	O(n2)	Dynamic Programming, <u>Graphs</u> , <u>Trees</u>	Bubble Sort (Slow comparison-based sorting)
<i>n</i> ≤ 106	O(n log n)	Sorting, Binary Search, Divide and Conquer	Merge Sort (Fast comparison-based sorting)
<i>n</i> ≤ 108	O(n)	Mathematical, Greedy	Min and max of element
n > 108	O(log n) or O(1)	Mathematical, Greedy	Binary Search



Common time complexities

Let n be the main variable in the problem.

- If $n \le 12$, the time complexity can be O(n!).
- If $n \le 25$, the time complexity can be $O(2^n)$.
- If $n \le 100$, the time complexity can be $O(n^4)$.
- If $n \le 500$, the time complexity can be $O(n^3)$.
- If $n \le 10^4$, the time complexity can be $O(n^2)$.
- If $n \le 10^6$, the time complexity can be O(n log n).
- If $n \le 10^8$, 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 ... 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(n²) [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)