**Best to Worst Time Complexities**

1. **O(1) - Constant Time**
   * The runtime does not depend on the input size.
   * Examples:
     + Accessing an element in an array by index.
     + Checking if a number is even or odd.
   * **Best case:** Very efficient.
2. **O(logn) - Logarithmic Time**
   * The runtime grows logarithmically as the input size increases.
   * Examples:
     + Binary search.
     + Operations in balanced trees (e.g., AVL or Red-Black Trees).
   * **Excellent for scalability.**
3. **O(n) - Linear Time**
   * The runtime grows linearly with the input size.
   * Examples:
     + Traversing an array or list.
     + Finding the minimum or maximum in an unsorted array.
   * **Good for moderate input sizes.**
4. **O(nlogn) - Linearithmic Time**
   * Slightly worse than linear time but still very efficient for many problems.
   * Examples:
     + Merge sort, heap sort.
     + Divide-and-conquer algorithms.
   * **Common in efficient sorting algorithms.**
5. **O(n2) - Quadratic Time**
   * The runtime grows quadratically with the input size.
   * Examples:
     + Bubble sort, insertion sort.
     + Algorithms with nested loops over the input.
   * **Only feasible for small input sizes.**
6. **O(n^3) - Cubic Time**
   * The runtime grows cubically with the input size.
   * Examples:
     + Naive matrix multiplication.
     + Algorithms with three nested loops.
   * **Rarely practical except for small inputs.**
7. **O(2^n) - Exponential Time**
   * The runtime doubles with each addition to the input size.
   * Examples:
     + Solving the traveling salesman problem using brute force.
     + Subset sum problem using brute force.
   * **Very slow; only feasible for tiny inputs.**
8. **O(n!) - Factorial Time**
   * The runtime grows factorially with the input size.
   * Examples:
     + Generating all permutations of a list.
     + Brute-force solutions to NP-complete problems.
   * **Worst case:** Almost always impractical.

**Visual Order (Best to Worst)**

* **O(1)** (Constant)
* **O(logn)** (Logarithmic)
* **O(sqrt(n))** (Square root)
* **O(n)** (Linear)
* **O(nlogn)** (Linearithmic)
* **O(n^2) / O(N\*M)** (Quadratic)
* **O(n^3)** (Cubic)
* **O(2^n)** (Exponential)
* **O(n!)** (Factorial)