Steps to Find Time Complexity:

1. **Identify the input size (`n`)**.
2. **Count the number of basic operations (like comparisons, assignments, loops)**.
3. **Focus on the dominant term** (ignore constants and lower-order terms).
4. **Express in Big O notation**.

☑Basic Examples to Understand

□ Example 1: Simple Loop

```
for(int i = 0; i < n; i++) {
    cout << i;
}
```

- **Operation Count**: Runs `n` times.
- **Time Complexity**: `O(n)`

□ Example 2: Nested Loop

```
for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
        cout << i << j;
    }
}
```

- Outer loop runs `n` times.
- Inner loop runs `n` times **for each iteration** of outer loop.
- **Total operations**: `n × n = n²`
- **Time Complexity**: `O(n²)`

□ Example 3: Loop with Divide by 2

```
for(int i = n; i > 0; i /= 2) {
    cout << i;
}
```

- Each time, `i` is divided by 2.
- Number of iterations ≈ $\log_2(n)$
- **Time Complexity**: `O(log n)`

□ Example 4: Combination

```
for(int i = 0; i < n; i++) {              // O(n)
    for(int j = 0; j < n; j++) {          // O(n)
        cout << i << j;
    }
}
for(int k = 0; k < n; k++) {              // O(n)
    cout << k;
}
```

- First part: $O(n^2)$
- Second part: $O(n)$
- **Total Time Complexity**: $O(n^2 + n) \rightarrow O(n^2)$ (since n^2 is dominant)

□ Important Rules:

Code Pattern	Time Complexity
Sequential (one after another)	Add them → $O(n) + O(n^2) = O(n^2)$
Nested loops	Multiply them → $O(n) \times O(n) = O(n^2)$
Divide input each time	$O(\log n)$
Binary Search	$O(\log n)$
Loop through array once	$O(n)$

□ Tips:

- Ignore constants: $O(2n) \rightarrow O(n)$
- Focus on worst-case time.
- Know common complexities:
  $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$