Time and Space Complexity (Beginner to Advanced)

What Are Time and Space Complexity?

When we write code, we often care about:

- How fast it runs   Time Complexity
- How much memory it uses   Space Complexity

Step-by-Step Learning Plan

1. What is Time Complexity (Basics)
2. Big O Notation (Very Important)
3. Common Time Complexities
4. Dry Run and Count Steps
5. What is Space Complexity
6. Examples for Both
7. Nested Loops, Recursion, and Functions
8. Best, Worst, Average Case
9. Advanced Topics (Optional)

What is Time Complexity?

Time complexity is how the number of operations grows as the input size (n) grows.

Example:

```
function printNumbers(n) {
  for (let i = 1; i <= n; i++) {
    console.log(i);
  }
}
```

If n = 5, it prints 5 numbers.

If n = 1,000,000, it prints 1 million numbers.

We write this using Big O Notation, like O(n).

What is Big O Notation?

Big O tells us the upper bound of time as input grows.

Common Big O Complexities:

- O(1) - Constant Time
- O(n) - Linear Time
- O(n) - Quadratic Time
- O(log n) - Logarithmic Time

Common Time Complexities (Examples)

O(1):

```
function getFirstItem(arr) {
  return arr[0];
}
```

O(n):

```
function sum(arr) {
  let total = 0;
  for (let i = 0; i < arr.length; i++) {
    total += arr[i];
  }
}
```

O(n):

```
function printPairs(arr) {
  for (let i = 0; i < arr.length; i++) {
    for (let j = 0; j < arr.length; j++) {
      console.log(arr[i], arr[j]);
    }
  }
}
```

O(log n) - Binary Search:

```
function binarySearch(arr, target) {
  let start = 0, end = arr.length - 1;
  while (start <= end) {
    let mid = Math.floor((start + end) / 2);
    if (arr[mid] === target) return mid;
    else if (arr[mid] < target) start = mid + 1;
    else end = mid - 1;
  }
  return -1;
}
```

Dry Run and Count Steps

Example:

```
function doubleLoop(n) {
  for (let i = 0; i < n; i++) {
    for (let j = 0; j < n; j++) {
      console.log(i, j);
    }
  }
}
```

Time: O(n)

What is Space Complexity?

It tells how much extra memory your algorithm uses.

O(n) example:

```
function createArray(n) {
  let arr = [];
  for (let i = 0; i < n; i++) {
    arr.push(i);
  }
  return arr;
}
```

O(1) example:

```
function sum(n) {
  let total = 0;
  for (let i = 1; i <= n; i++) {
    total += i;
  }
  return total;
}
```

Real Examples (Mix of Time + Space)

```
function multiplyAll(arr1, arr2) {
  let result = [];
  for (let i = 0; i < arr1.length; i++) {
    for (let j = 0; j < arr2.length; j++) {
      result.push(arr1[i] * arr2[j]);
    }
  }
  return result;
}
```

Time: O(n), Space: O(n)

Nested Loops, Recursion, Function Calls

Example (Recursion):

```
function factorial(n) {
  if (n === 1) return 1;
  return n * factorial(n - 1);
}
```

Time: O(n), Space: O(n)

Best, Worst, Average Case

Example:

```
function findInArray(arr, target) {
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] === target) return i;
  }
  return -1;
}
```

Best: O(1), Worst: O(n), Average: O(n)

Advanced Topics (Optional Later)

- Amortized Analysis
- Time/Space tradeoffs
- Master Theorem (for Recursion)
- Comparing algorithms for real-world cases

Your Next Steps

Would you like:

- Quizzes or practice problems?
- Step-by-step walkthrough of dry running some algorithms?
- Visual comparison of time complexities?