Complexity Analysis and Asymptotic Notation

Growth Rate of Functions & Notation Overview

Complexity Analysis Overview

- • Complexity analysis estimates resources required by an algorithm
- • Types of Complexity:
-   - Time Complexity: Time taken by an algorithm
-   - Space Complexity: Memory used by an algorithm

Growth Rate of Functions

- • Constant Time: $O(1)$
- • Logarithmic Time: $O(\log n)$
- • Linear Time: $O(n)$
- • Linearithmic Time: $O(n \log n)$
- • Quadratic Time: $O(n^2)$
- • Exponential Time: $O(2^n)$

Asymptotic Notation

- • Big O (O(f(n))): Upper bound (Worst-case)
- • Big Ω (Ω(f(n))): Lower bound (Best-case)
- • Big Θ (Θ(f(n))): Tight bound (Average-case)

Common Complexity Classes

- • $O(1)$: Hash table lookup
- • $O(\log n)$: Binary search
- • $O(n)$: Linear search
- • $O(n \log n)$: Merge sort
- • $O(n^2)$: Bubble sort
- • $O(2^n)$: Exponential problems

Example Analysis

- Code Example:

```
def example_function(arr):
    for i in range(len(arr)):
        for j in range(len(arr)):
            print(arr[i], arr[j])
```

- • Time Complexity: $O(n^2)$
- • Space Complexity: $O(1)$