Basics: Time/Space Complexity

Time Complexity: measures "how fast" an algo runs

↓

# of operations that an algo makes as a function of input size

Ex: Count # times a^* appears in a list $x_1, x_2 \cdots x_n$

Sequential count: # operations = n

Ex: Check if a^* appears in $x_1 \cdots x_n$

Sequential check: lucky: $x_1 = a^*$ (1 operation)
unlucky: $x_n = a^*$ / a^* not there (n operations)

Time Complexity: "worst-case" # operations

↓

bound ≥ # of operations for any input of size n

↓

n

Big-Oh notation

Time complexity bound as $O(f(n))$

"order" $f(n)$ time

$n, n^2, \log n \cdots$

→ simplified analysis of # of operations

→ focuses on the case when $n \to \infty$

Properties

(1) Ignore constants $\not{2}n$ $O(n)$

② Only care about "fastest" growing terms

$n^2 + n$ $\quad O(n^2)$

n^2 much faster

space $O(n^2)$ ← Loops: n iterations, each loop has n^2 steps $\Big\}$ $n \cdot n^2 = O(n^3)$ time

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^k) < O(2^n)$$

Space Complexity: amount of space (memory) required by an algorithm

① Array of n rows and n columns $\quad O(n^2)$