Order of Growth

Simplifying the abstraction of running time

↳ rate of growth

$T(n) = an^2 + bn + c$

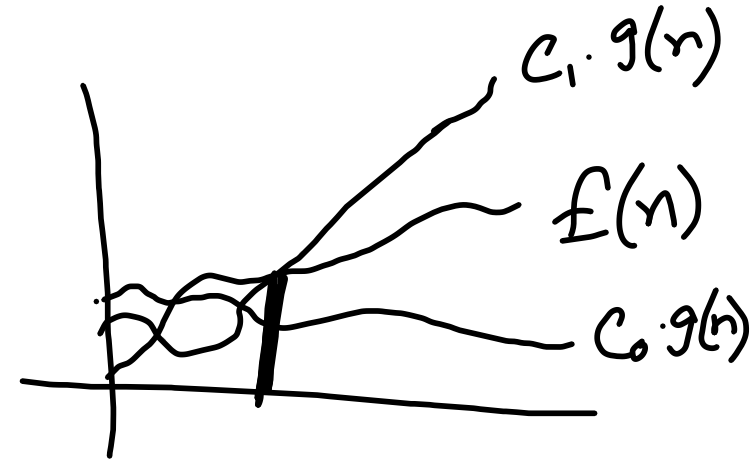
↓

$T(n) = \Theta(n^2)$

- Consider only the leading term.
- Ignore the constant.

Algo A is more efficient than algo B if "worst-case" running time of A has a lower order of growth.

Asymptotic Notations



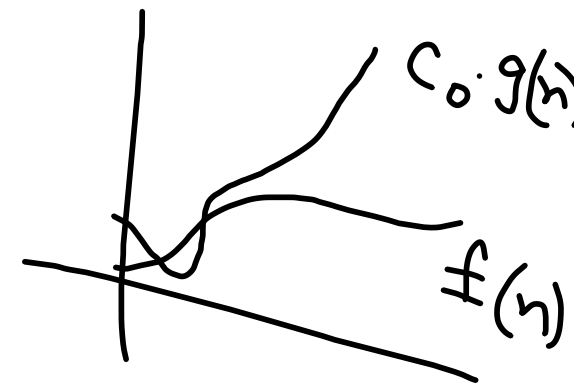
① Theta (Θ)

$$\Theta(g(n)) = \{ f(n) : \exists c_0, c_1, n_0 > 0 \text{ s.t. } 0 \le c_0 \cdot g(n) \le f(n) \le c_1 \cdot g(n) \ \forall n \ge n_0 \}$$

$g(n)$ is asymptotically tight bound for $f(n)$

② Big-Oh (O)

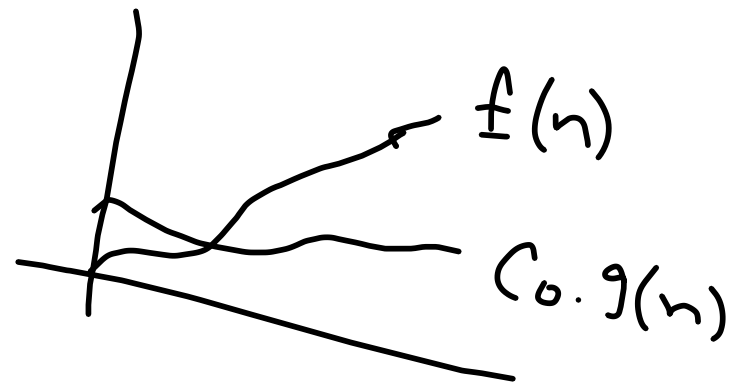
$$O(g(n)) = \{ f(n) : \exists c_0, n_0 > 0 \text{ s.t } 0 \le f(n) \le c_0 \cdot g(n) \ \forall n \ge n_0 \}$$



$g(n)$ is asymptotically upper bound for $f(n)$.

③ Omega (Ω)

$$\Omega(g(n)) = \{ f(n) : \exists\ n_0, c_0 \text{ s.t. } 0 \le c_0 \cdot g(n) \le f(n)\ \forall n \ge n_0 \}$$



Running Time of Insertion Sort

$T(n) = an + b$ (Best Case)

$T(n) = an^2 + bn + c$ (Avg / Worst Case)

$T(n) = an^2 + bn + c$

$\le (a+1)n^2$

$bn + c \le n^2$

$T(n) \ge an^2$

Worst-case running time: $\Theta(n^2)$

running time (irrespective of i/p) : $O(n^2)$

④ Small-oh (o)

$$o(g(n)) = \{ f(n) : \underline{\forall c} , \exists n_0 \text{ s.t. } 0 \leq f(n) \underline{<} c \cdot g(n) \} \quad \forall n \geq n_0$$

$2n = o(n^2)$ | $2n^2 \neq o(n^2)$

⑤ Small-omega (Ω)

$$\Omega(g(n)) = \{ f(n) : \forall c, \exists n_0 \text{ s.t. } 0 \leq c \cdot g(n_0) < f(n) \} \quad \forall n \geq n_0$$

Data Structures

Interfaces

- Specification
  - What data to store
  - Operations on data should be supported
- Sequence $\langle x_0, \ldots, x_{n-1} \rangle$
- Sets

Data Structure

- representation
  - how to store
- Algorithms corresponding to these operations
  - Array
  - Pointer-based (Link-list)

Data Structures

(Problem)

Interfaces

- Specification
  - What data to store
  - Operations on data should be supported
- Sequence $\langle x_0, \ldots, x_{n-1} \rangle$
- Sets

(Solution)

Data Structure

- representation
  - how to store
- Algorithms corresponding to these operations
  - Array
  - Pointer-based (Link-list)

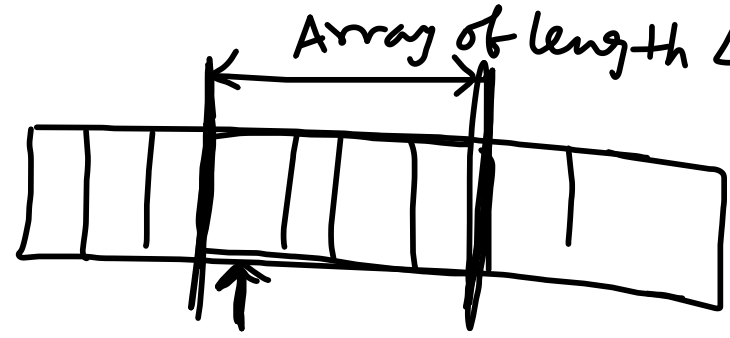
Static Interface (Sequence)

$\langle x_0, x_1, \ldots, x_{n-1} \rangle$

- build(X) → creates the sequence — O(n)
- iter_seq(X) → O/p $x_0, x_1, \ldots, x_{n-1}$ in ordered sequence — O(n)
- get_at(i) → O/p x_i — O(1)
- set_at(x, i) → Set x_i to the value x. — O(1)

Static → You cannot modify size of the sequence.

Static Array



(Word RAM)

Word → w-bit.

Array → consecutive chunk of words

Array[i] ≡ Memory[Addr(Array) + i]