Asymptotic Notations

Introduction

- In mathematics, computer science, and related fields, **big O notation** describes the limiting behavior of a function when the argument tends towards a particular value or infinity, usually in terms of simpler functions. Big O notation allows its users to simplify functions in order to concentrate on their growth rates: different functions with the same growth rate may be represented using the same O notation.
- The **time complexity** of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the size of the input to the problem. When it is expressed using big O notation, the time complexity is said to be described *asymptotically*, i.e., as the input size goes to infinity.

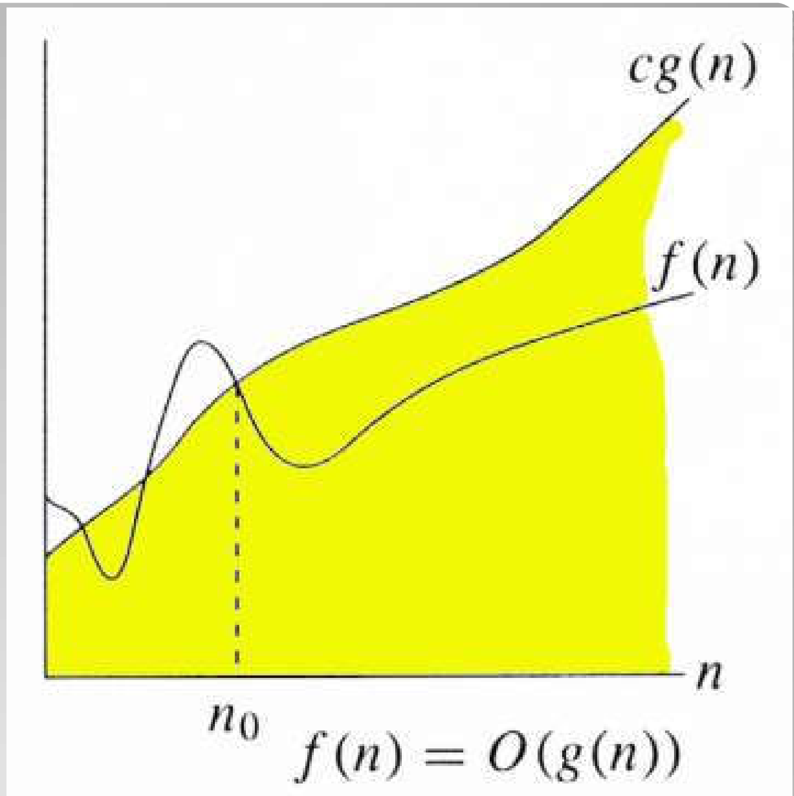
Asymptotic Complexity

- Running time of an algorithm as a function of input size n **for large *n***.
- Expressed using only the **highest-order term** in the expression for the exact running time.
  - Instead of exact running time, say $\Theta(n^2)$.
- Describes behavior of function in the limit.
- Written using ***Asymptotic Notation***.
- The notations describe different rate-of-growth relations between the defining function and the defined set of functions.

O-notation

For function $g(n)$, we define $O(g(n))$, big-O of n, as the set:

$O(g(n)) = \{f(n)$:
$\exists$ positive constants c and n_0, such that $\forall n \geq n_0$,
we have $0 \leq f(n) \leq cg(n)$ }



Intuitively: Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.

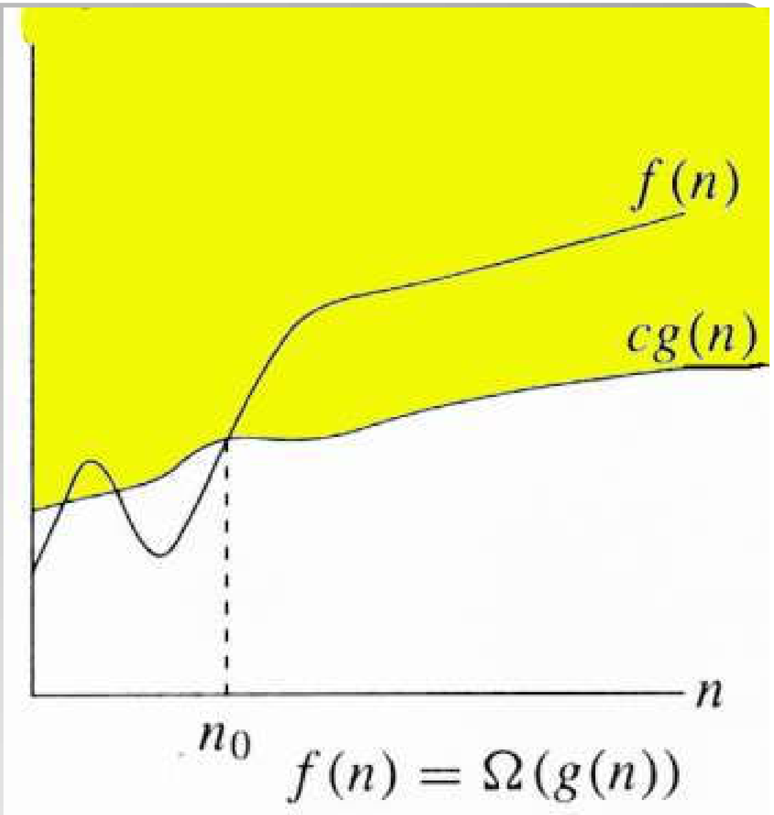
***g*(*n*) is an *asymptotic upper bound* for *f*(*n*).**

$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$.
$\Theta(g(n)) \subset O(g(n))$.

Ω -notation

For function $g(n)$, we define $\Omega(g(n))$, big-Omega of n, as the set:

$\Omega(g(n)) = \{f(n)$:
$\exists$ positive constants c and n_0, such that $\forall n \geq n_0$,
we have $0 \leq cg(n) \leq f(n)\}$



Intuitively: Set of all functions whose *rate of growth* is the same as or higher than that of $g(n)$.

$g(n)$ is an *asymptotic lower bound* for $f(n)$.

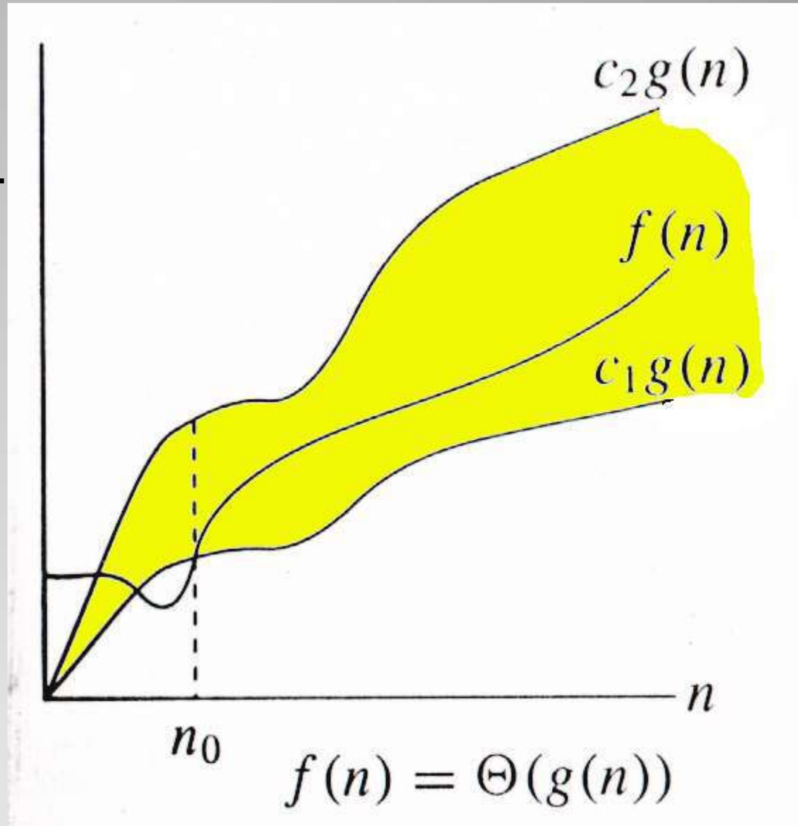
$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n))$.
$\Theta(g(n)) \subset \Omega(g(n))$.

Θ-notation

For function $g(n)$, we define $\Theta(g(n))$, big-Theta of n, as the set:

$\Theta(g(n)) = \{f(n)$:
$\exists$ positive constants c_1, c_2, and n_0, such that $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\}$

Intuitively: Set of all functions that have the same *rate of growth* as $g(n)$.

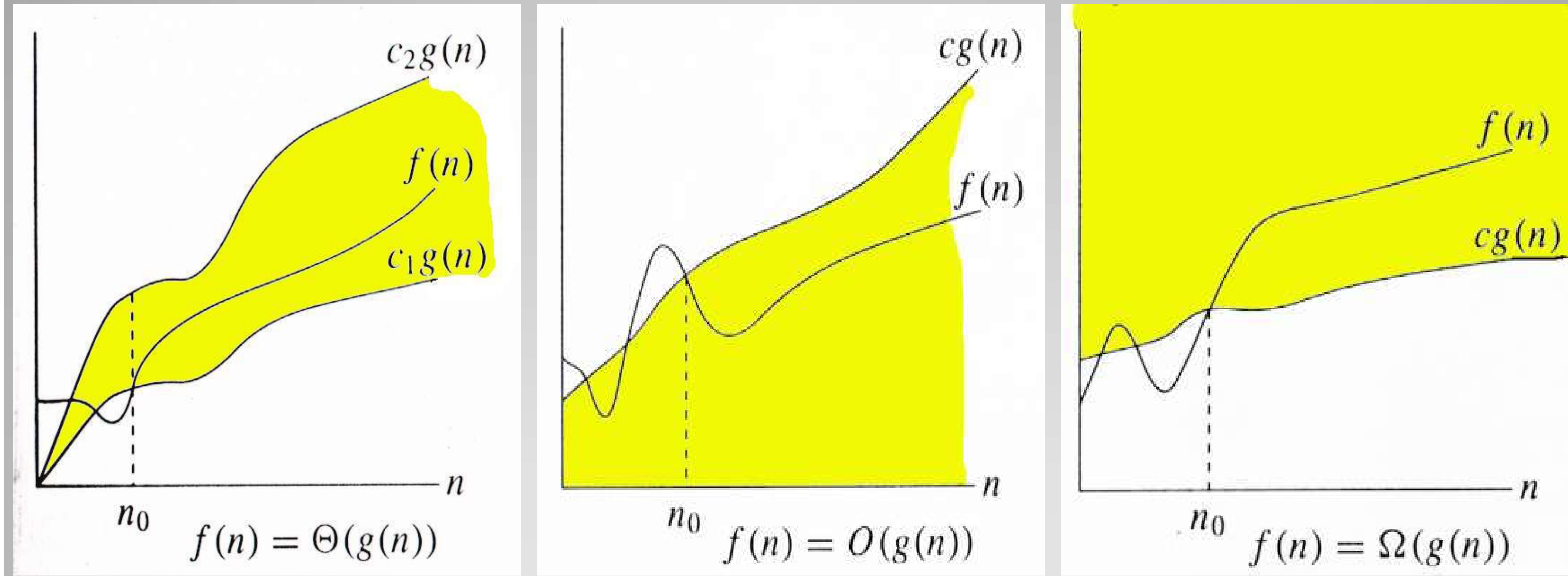


***g*(*n*) is an *asymptotically tight bound* for *f*(*n*).**

Definitions

- Upper Bound Notation:
  - f(n) is O(g(n)) if there exist positive constants c and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$
  - Formally, $O(g(n)) = \{ f(n): \exists$ positive constants c and n_0 such that $f(n) \leq c \cdot g(n)\ \forall\ n \geq n_0$
  - Big O fact: A polynomial of degree k is $O(n^k)$
- Asymptotic lower bound:
  - f(n) is $\Omega(g(n))$ if $\exists$ positive constants c and n_0 such that $0 \leq c \cdot g(n) \leq f(n)\ \ \forall\ n \geq n_0$
- Asymptotic tight bound:
  - f(n) is $\Theta(g(n))$ if $\exists$ positive constants c_1, c_2, and n_0 such that $c_1\, g(n) \leq f(n) \leq c_2\, g(n)\ \forall\ n \geq n_0$
  - $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ AND $f(n) = \Omega(g(n))$

Relations Between Θ, O, Ω



o-notation

For a given function $g(n)$, the set little-*o*:

$o(g(n)) = \{f(n)$: $\forall$ $c > 0$, $\exists$ $n_0 > 0$ such that $\forall$ $n \geq n_0$, we have $0 \leq f(n) < cg(n)\}$.

$f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity:

$$\lim_{n \to \infty} [f(n) / g(n)] = 0$$

$g(n)$ is an ***upper bound*** for $f(n)$ that is not asymptotically tight.

ω -notation

For a given function $g(n)$, the set little-omega:

$\omega(g(n))$ = $\{f(n)$: $\forall$ $c > 0$, $\exists$ $n_0 > 0$ such that $\forall$ $n \geq n_0$, we have $0 \leq cg(n) < f(n)\}$.

$f(n)$ becomes arbitrarily large relative to $g(n)$ as n approaches infinity:

$$\lim_{n \to \infty} [f(n) / g(n)] = \infty.$$

$g(n)$ is a ***lower bound*** for $f(n)$ that is not asymptotically tight.

Comparison of Functions

$$f \leftrightarrow g \;\approx\; a \leftrightarrow b$$

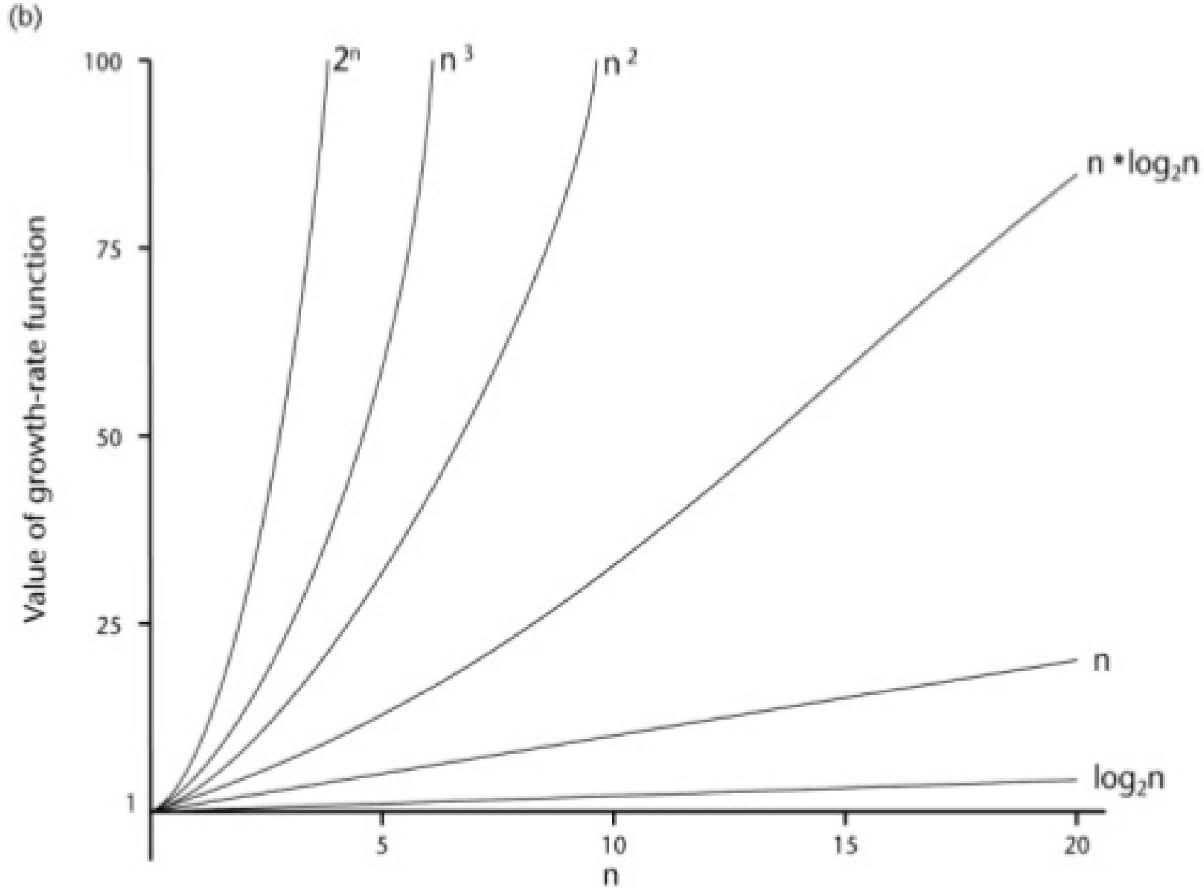
$$f(n) = O(g(n)) \;\approx\; a \le b$$
$$f(n) = \Omega(g(n)) \;\approx\; a \ge b$$
$$f(n) = \Theta(g(n)) \;\approx\; a = b$$
$$f(n) = o(g(n)) \;\approx\; a < b$$
$$f(n) = \omega(g(n)) \;\approx\; a > b$$

Review: Other Asymptotic Notations

- Intuitively, we can simplify the above by:

  - o() is like <
  - O() is like ≤
  - ω() is like >
  - Ω() is like ≥
  - Θ() is like =

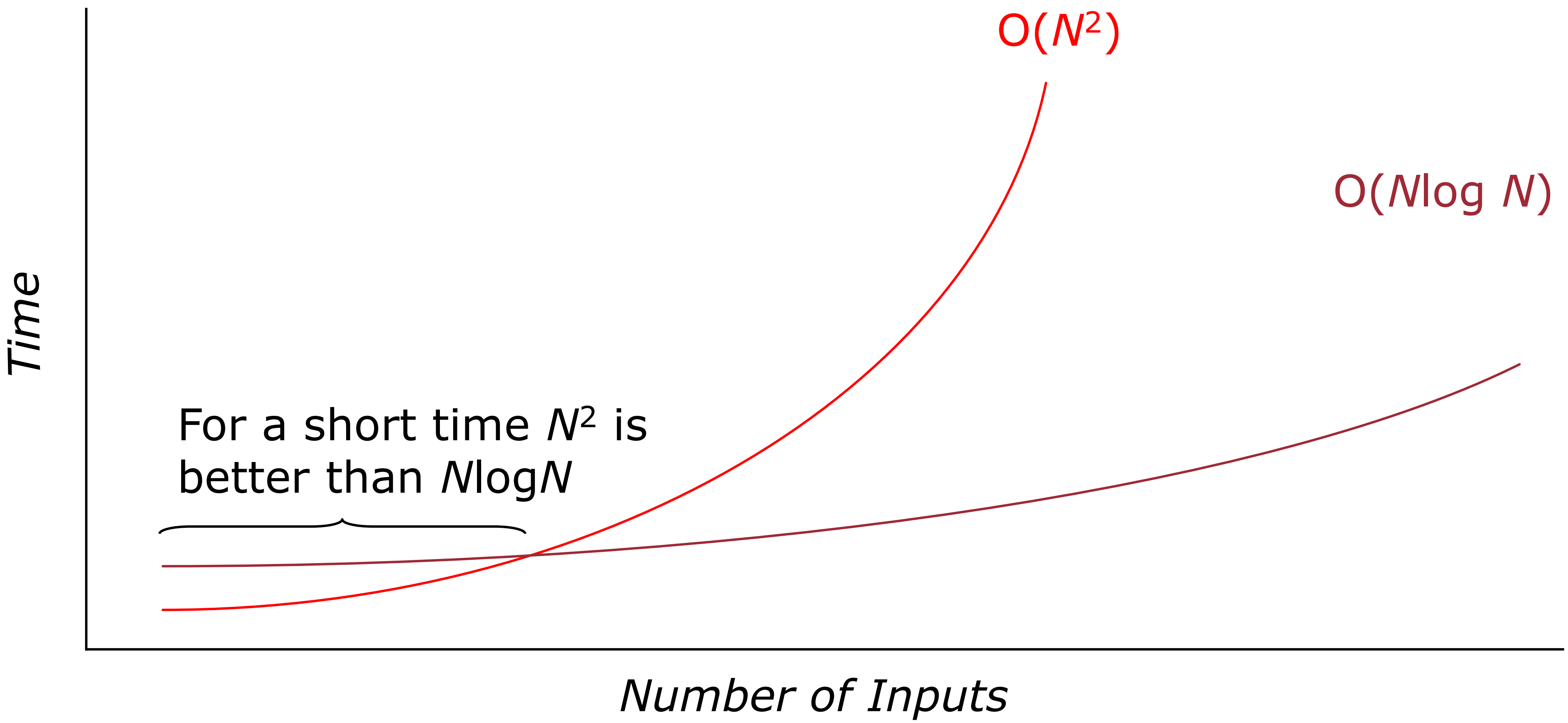
A Comparison of Growth-Rate Functions



Common growth rates

Time complexity		Example
O(1)	*constant*	Adding to the front of a linked list
O(log *N*)	*log*	Finding an entry in a sorted array
O(*N*)	*linear*	Finding an entry in an unsorted array
O(*N* log *N*)	*n-log-n*	Sorting n items by ‘divide-and-conquer’
O(N^2)	*quadratic*	Shortest path between two nodes in a graph
O(N^3)	*cubic*	Simultaneous linear equations
O(2^N)	*exponential*	The Towers of Hanoi problem

Growth rates



Running Times

- "Running time is $O(f(n))$" $\Rightarrow$ Worst case is $O(f(n))$
- $O(f(n))$ bound on the worst-case running time $\Rightarrow$ $O(f(n))$ bound on the running time of every input.
- $\Theta(f(n))$ bound on the worst-case running time $\Rightarrow$ $\Theta(f(n))$ bound on the running time of every input.
- "Running time is $\Omega(f(n))$" $\Rightarrow$ Best case is $\Omega(f(n))$
- Can still say "Worst-case running time is $\Omega(f(n))$"
  - Means worst-case running time is given by some unspecified function $g(n) \in \Omega(f(n))$.

Time Complexity Vs Space Complexity

- Achieving both is difficult and we have to make use of the best case feasible
- There is always a trade off between the space and time complexity
- If memory available is large then we need not compensate on time complexity
- If speed of execution is not main concern and the memory available is less then we can’t compensate on space complexity.

The End