Growth of Functions

CSCE310: Data Structures and Algorithms

`www.cse.unl.edu/~choueiry/S01-310/`

Berthe Y. Choueiry (Shu-we-ri)

Ferguson Hall, Room 104

`choueiry@cse.unl.edu`, Tel: (402)472-5444

Growth of Functions

By counting the cost of steps in an algorithm, we compute running time $T(n)$

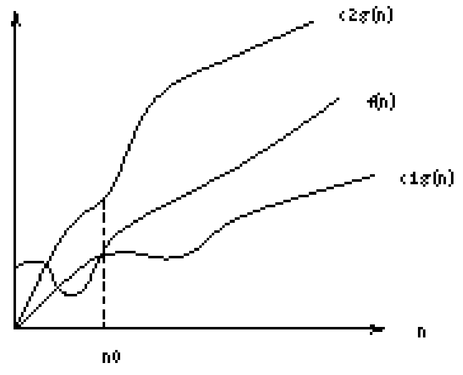
$T(n)$ is a numerical function whose domain is $\mathbb{N}$

Determining the **exact** running time $T(n)$ is too complicated

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + \\ c_6 \sum_{j=2}^{n}(t_j - 1) + c_7 \sum_{j=2}^{n}(t_j - 1) + c_8(n-1)$$

When n is large enough (large inputs), only the order of growth is relevant $\longrightarrow$ Asymptotic efficiency of algorithms

Definition: $\Theta(g(n))$ is a set of functions



$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2, n_0 such that

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \; \forall n \geq n_0\}$$

Note: $g(n)$ and any $f(n)$ in the set are nonnegative $\forall n \geq n_0$

To prove $f(n)$ is in the set, find: $c_1 \geq 0$, $c_2 \geq 0$, $n_0 \geq 0$

Formally: $f(n) \in \Theta(g(n))$, but we write $f(n) = \Theta(g(n))$

$g(n)$ is an asymptotically tight bound to $f(n)$

Example:

Prove that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Find c_1, c_2, n_0 such that $c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2 \ \forall n \geq n_0$

Divide by n^2: $c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$

Consider: $\frac{1}{2} - \frac{3}{n} \leq c_2$

Choose $c_2 = \ldots$

Consider: $c_1 \leq \frac{1}{2} - \frac{3}{n}$,

choose $n_0 = \ldots$ such that $c_1 \geq 0$

choose c_1

With $c_1 = 1/14, c_2 = 1/2, n_0 = 7$,

we can verify that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Example: (cont')

Prove that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Many choices exist, we need to find *one*

For $f(n) = \frac{1}{2}n^2 - 3n$, we choose $c_1 = 1/14, c_2 = 1/2, n_0 = 7$
For $f'(n)$, we need to find other constants

Further prove $6n^3 \neq \Theta(n^2)$

Suppose $\exists\, c_2, n_0$ such that $6n^3 \leq c_2 n^2$, $\forall n \geq n_0$,
then $n \leq c_2/6$, which does not hold for $\forall n \leq c_2/6$

Choice of c_1, c_2

Set:

c_1 slightly smaller than the coefficient of the highest order term of $f(n)$

c_2 slightly larger than the coefficient of the highest order term of $f(n)$

When highest order term has a coefficient, ignore it (c_1 and c_2 can adapt)

Suppose $f(n) = an^2 + bn + c$, with $a > 0, b, c$ constants

$$\left.\begin{array}{l}\text{Highest order term: } n^2 \\ a \text{ can be ignored}\end{array}\right\} \Longrightarrow f(n) = \Theta(n^2)$$

Otherwise, choose $c_1 = a/4, c_2 = 7a/4, n_0 = 2 \cdot max((|b|/a), \sqrt{(|c|/a)})$

Note

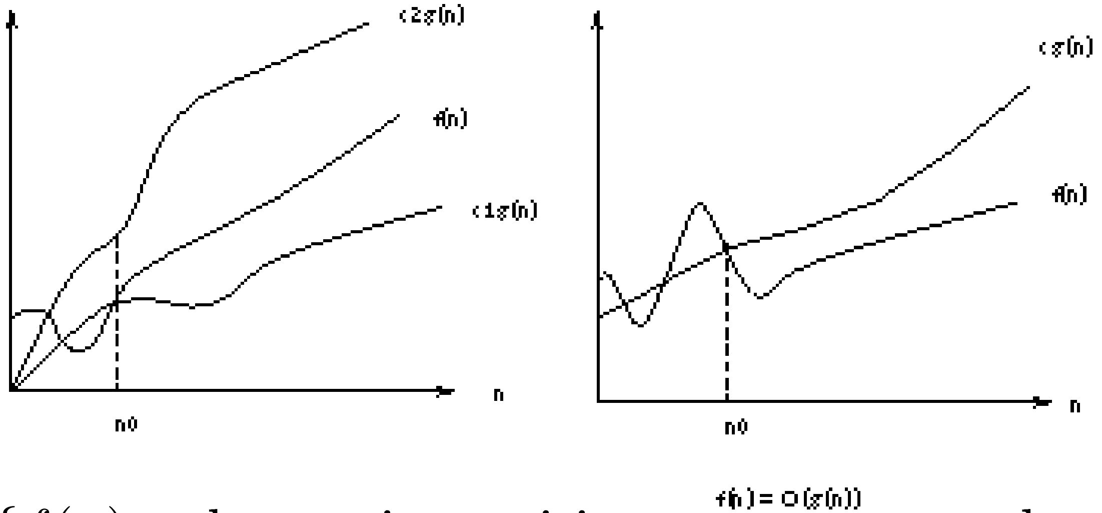
In general,
for any polynomial $p(n) = \sum_{i=0}^{d} a_i n^i$, with a_i constants and $a_d > 0$,

we have: $p(n) = \Theta(n^d)$

Special case, a constant!

A constant can be expressed as a degree $\ldots$ polynomial,
$\Theta(n^{\cdots}) = \Theta(\ldots)$

Definition: $O(g(n))$ asymptotic upper bound



$O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that
$$0 \leq f(n) \leq cg(n) \quad \forall n \geq n_0\}$$

$\forall\, n \geq n_0$, $f(n)$ is <u>on</u> or <u>below</u> $g(n)$

Formally: $f(n) \in O(g(n))$, but we write $f(n) = O(g(n))$

Note: $\Theta(g(n)) \subseteq O(g(n))$ and $f(n) = \Theta(g(n)) \Longrightarrow f(n) = O(g(n))$

Example: $f(n) = an^2 + bn + c$ is in$\Theta(n^2)$ $(a > 0)$, thus in $O(n^2)$
Also, $an + b$ is in $O(n^2)$, verified with $c = a + |b|$ and $n_0 = 1$

Warning

$an + b$ is in $O(n^2)$ looks weird

In the literature, O-notation often used to denote asymptotic tight bounds

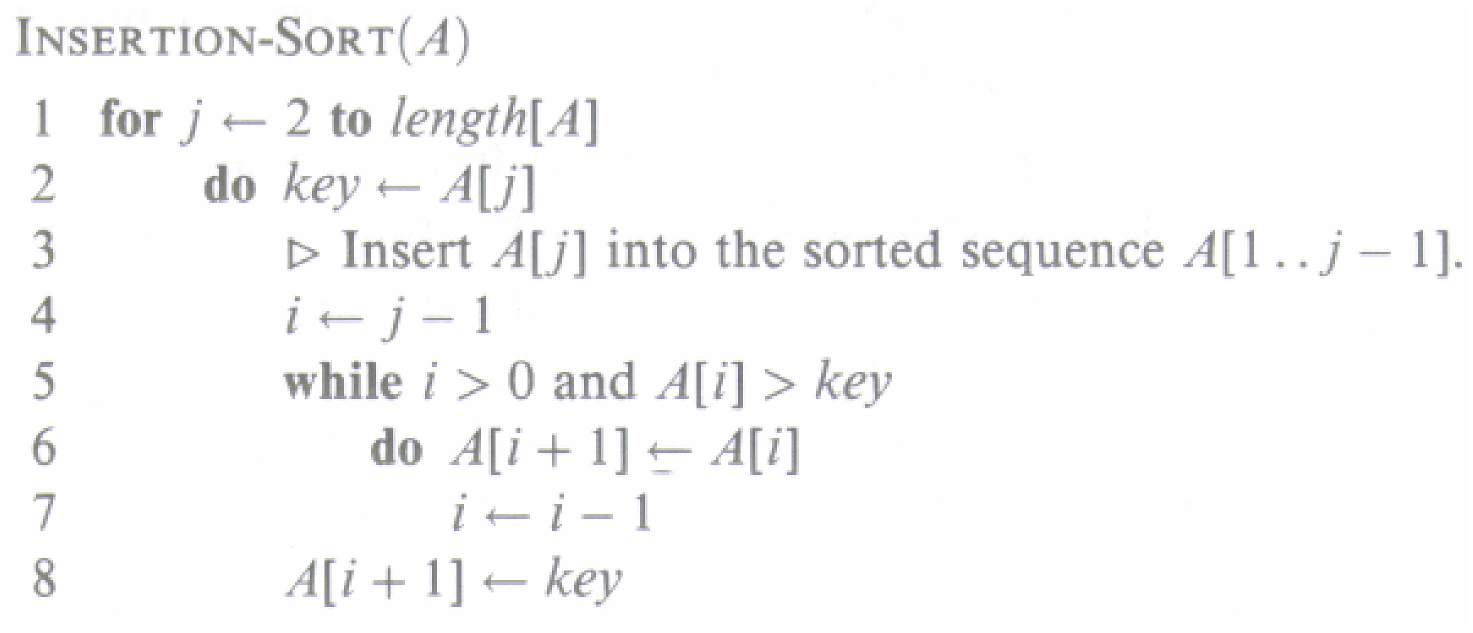
This informal use is quite frequent

Formally:

- Asymptotic tight bound: Θ-notation
- Asymptotic upper bound: O-notation

Running time of an algorithm

Upper bound on the running time of an algorithm for all inputs is usually easy to derive.



```
INSERTION-SORT($A$)
for $j \leftarrow 2$ to $length[A]$
    do $key \leftarrow A[j]$
       ▷ Insert $A[j]$ into the sorted sequence $A[1..j-1]$.
       $i \leftarrow j - 1$
       while $i > 0$ and $A[i] > key$
           do $A[i+1] \leftarrow A[i]$
              $i \leftarrow i - 1$
       $A[i+1] \leftarrow key$
```

Double nested loop; i, j at most n; thus $O(n^2)$

Use of O-notation to bound running time

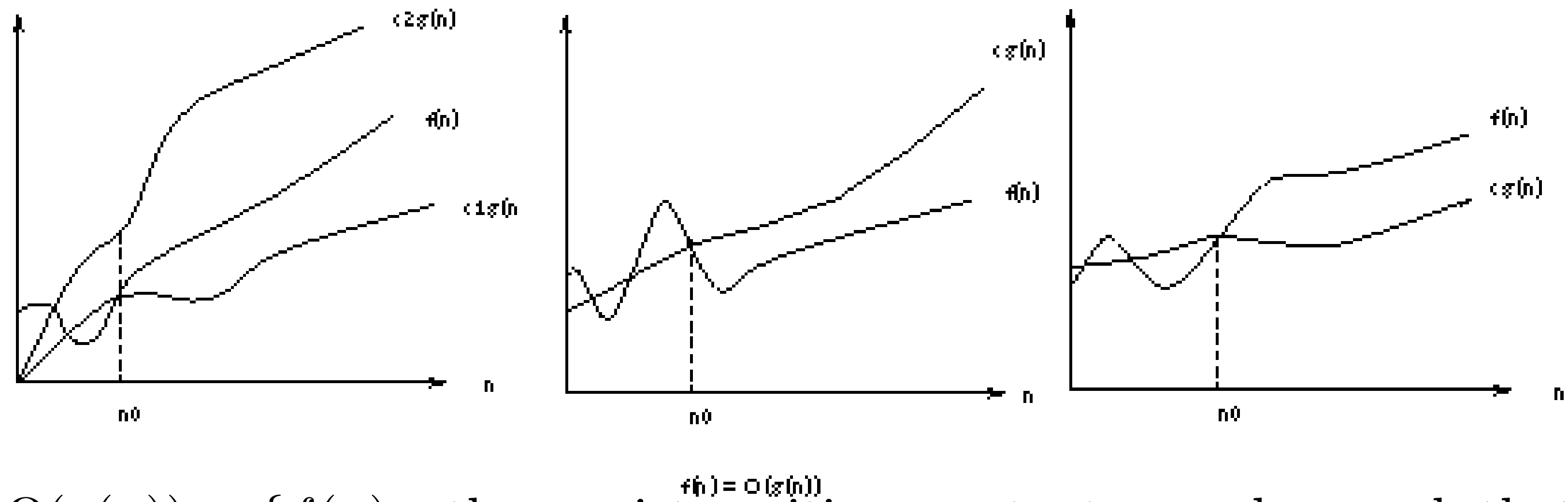
Consider:

- $T(n)$, the running time of an algorithm
- the worst-case of $T(n)$
- Θ-notation, the order of growth of the $\underline{\text{worst-case}}$ of $T(n)$
- $\rightarrow O$-notation, asymptotic upper bound $\underline{\text{on every input}}$

For insertion-sort, the worst-case running time is in $\Theta(n^2)$, thus the upper bound of the running time on every input is $O(n^2)$

However, $\underline{\text{worst-case}}$ running time of insertion-sort is in $\Theta(n^2) \not\Rightarrow$ the running time of insertion-sort is in $\Theta(n^2)$ $\underline{\text{(on every input)}}$
Why?.. remember the bound on the best-case running time

Definition: $\Omega(g(n))$ asymptotic lower bound



$\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that

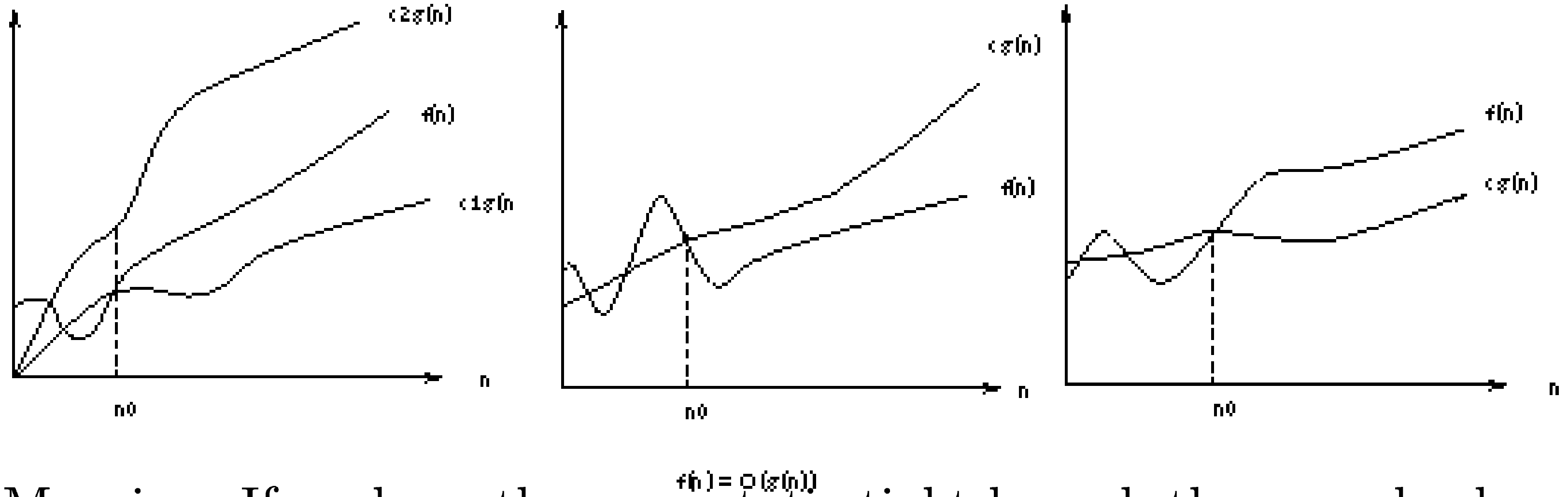
$$0 \leq cg(n) \leq f(n) \quad \forall n \geq n_0\}$$

$\forall\ n \geq n_0$, $f(n)$ is on or above $g(n)$

Important theorem

For any two functions $f(n)$ and $g(n)$,

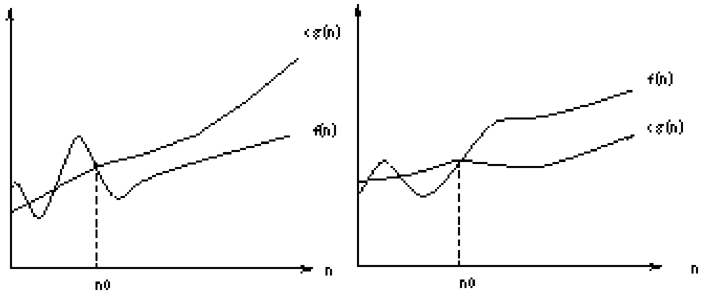
$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) and f(n) = \Omega(g(n))$$



Meaning: If we have the asymptotic tight bound, then we also have the asymptotic upper and lower bounds!

In practice, we don't have the asymptotic tight bound, but we try to derive them from the asymptotic upper and lower bounds

Worst/best-case running time vs. running time (on every input)



f(n) = O(g(n))

We saw that the order of growth of the worst-case running time of an algorithm is the upper bound of its running time on every input

Similarly, the order of growth of the best-case running time of an algorithm is the lower bound of its running time on every input

For insertion-sort, the order of growth of the best-case running time is $\Theta(n)$, thus the running time on every input is $\Omega(n)$

So, the running time of insertion-sort is between $\Omega(n)$ and $O(n^2)$, and these bounds are asymptotically as tight as possible

Arithmetic of Big-O, Ω, and Θ notations

- Transitivity:
  - $f(n) \in O(g(n))$ and $g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$
  - $f(n) \in \Theta(g(n))$ and $g(n) \in \Theta(h(n)) \Rightarrow f(n) \in \Theta(h(n))$
  - $f(n) \in \Omega(g(n))$ and $g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$
- Reflexivity: Ω, Θ, O
- Symmetry: Θ
- Transpose symmetry: Ω, O
- Scaling: if $f(n) \in O(g(n))$ then for any $k > 0, f(n) \in O(kg(n))$
- Sums: if $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$ then $(f_1 + f_2)(n) \in O(max(g_1(n), g_2(n)))$