Basics of Algorithm Analysis

History

- Al-khwarizmi (Persian) introduced the concept of algorithm.
- Also founded the discipline of algebra.
- Around 800 CE

What is an algorithm?

Definition:

Example: Suppose you have 8 balls ...

- We analyze an algorithm on the basis of its following resource requirements.

1. Time required.
2. Space required.

Theoretical Analysis

- Describe the algorithm in pseudo code.
- Count the number of pseudo code steps.
- Characterize the running time as a function of the input size, *n*.

Example

- The algorithm below finds the maximum element in an array of size n.

Algorithm	# operations
Algorithm $arrayMax(A, n)$	
$currentMax \leftarrow A[0]$	1
for $i \leftarrow 1$ **to** $n - 1$ **do**	$(n - 1)$
if $A[i] > currentMax$ **then**	$(n - 1)$
$currentMax \leftarrow A[i]$	$(n - 1)$
return $currentMax$	1
Total	$3n - 1$

Best case vs worst case

- What input array will lead to best case performance.
- What input array will lead to worst case performance.

Algorithm $arrayMax(A, n)$	# operations
$currentMax \leftarrow A[0]$	1
for $i \leftarrow 1$ **to** $n - 1$ **do**	$(n - 1)$
if $A[i] > currentMax$ **then**	$(n - 1)$
$currentMax \leftarrow A[i]$	$(n - 1)$
return $currentMax$	1
Total	$3n - 1$

Big-Oh Notation

- Given functions $f(n)$ and $g(n)$, we say that $f(n)$ is $O(g(n))$ if there are positive constants
  $c > 0$ and $n_0 >= 0$ such that

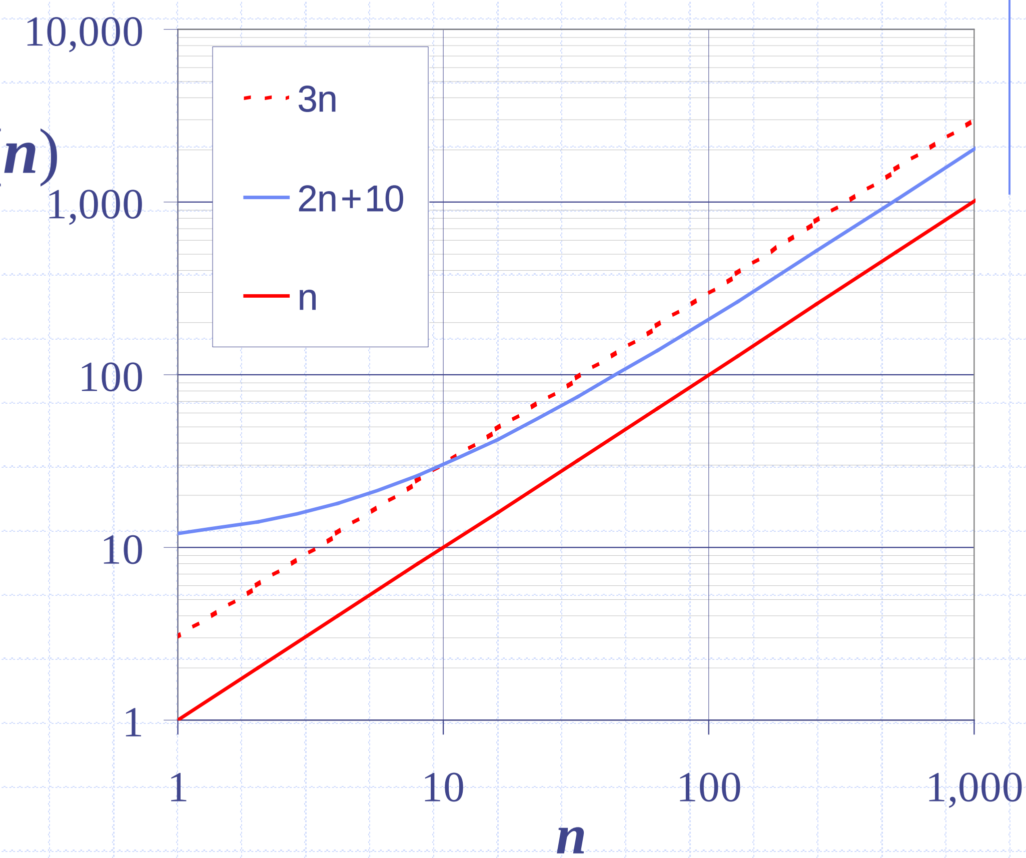
  $f(n) \leq cg(n)$ for $n \geq n_0$

- Example: $2n + 10$ is $O(n)$

  How ?

Big-Oh Notation

- Example: $2n + 10$ is $O(n)$
  - $2n + 10 \leq cn$
  - $(c - 2)\, n \geq 10$
  - $n \geq 10/(c - 2)$
  - Pick $c = 3$ and $n_0 = 10$



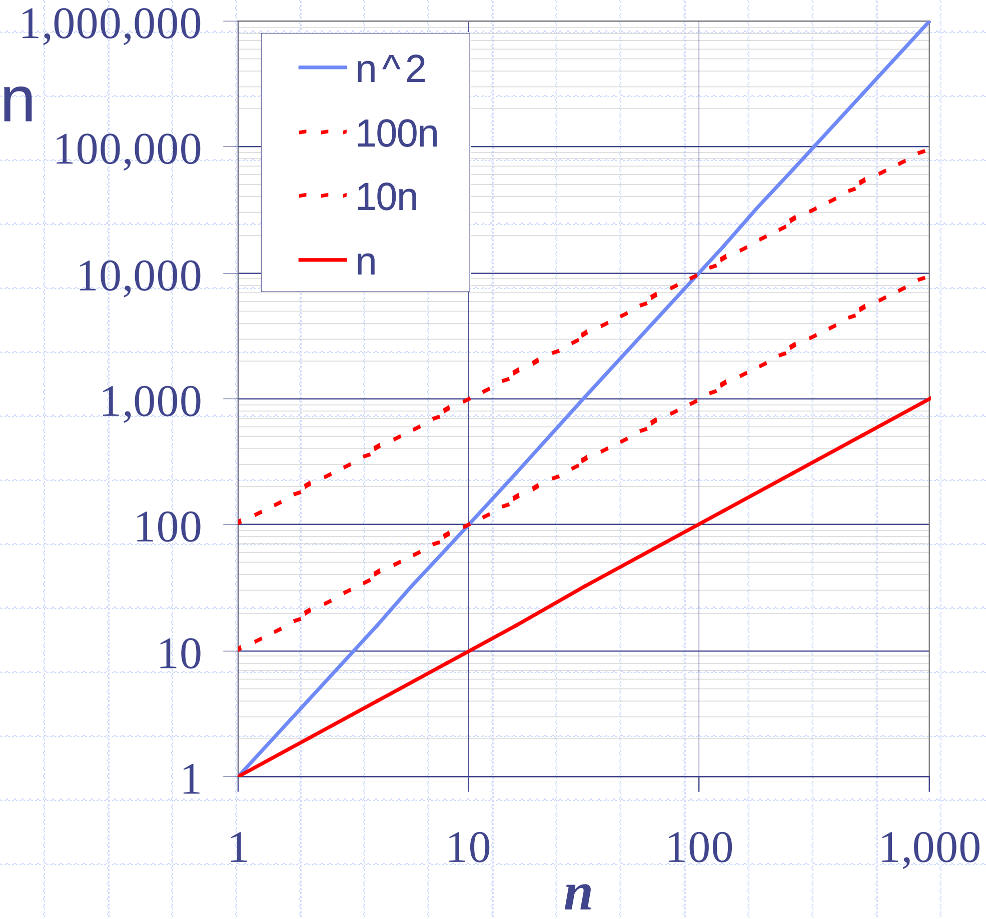
Big-Oh Example

- Example: the function n^2 is not $O(n)$

Why ?

Big-Oh Example

- Example: the function n^2 is not $O(n)$
  - $n^2 \leq cn$
  - $n \leq c$
  - The above inequality cannot be satisfied since c must be a constant



More Big-Oh Examples

- 7n-2

  7n-2 is O(n)

  need $c > 0$ and $n_0 \geq 1$ such that $7n-2 \leq c \cdot n$ for $n \geq n_0$

  this is true for $c = 7$ and $n_0 = 1$

- $3n^3 + 20n^2 + 5$

  $3n^3 + 20n^2 + 5$ is $O(n^3)$

  need $c > 0$ and $n_0 \geq 1$ such that $3n^3 + 20n^2 + 5 \leq c \cdot n^3$ for $n \geq n_0$

  this is true for $c = 4$ and $n_0 = 21$

- 3 log n + log log n

  3 log n + log log n is O(log n)

  need $c > 0$ and $n_0 \geq 1$ such that $3 \log n + \log \log n \leq c \cdot \log n$ for $n \geq n_0$

  this is true for $c = 4$ and $n_0 = 2$

Big-Oh and Growth Rate

- The big-Oh notation gives an upper bound on the growth rate of a function
- The statement "$f(n)$ is $O(g(n))$" means that the growth rate of $f(n)$ is no more than the growth rate of $g(n)$
- We can use the big-Oh notation to rank functions according to their growth rate,

e.g. $\log n$, $\log^2 n$, $\sqrt{n}$, n, n^3, 5^n

Big-Oh Rules

- If $f(n)$ is a polynomial of degree d, then $f(n)$ is $O(n^d)$, i.e.,
  1. Drop lower-order terms
  2. Drop constant factors
- Use the smallest possible class of functions
  - Say "$2n$ is $O(n)$" instead of "$2n$ is $O(n^2)$"
- Use the simplest expression of the class
  - Say "$3n + 5$ is $O(n)$" instead of "$3n + 5$ is $O(3n)$"

Example revisited

- We say that algorithm ***arrayMax*** "runs in $O(n)$ time"

Algorithm *arrayMax*(*A*, *n*)	# operations
$\quad currentMax \leftarrow A[0]$	1
$\quad$ **for** $i \leftarrow 1$ **to** $n - 1$ **do**	$(n - 1)$
$\quad\quad$ **if** $A[i] > currentMax$ **then**	$(n - 1)$
$\quad\quad\quad currentMax \leftarrow A[i]$	$(n - 1)$
$\quad$ **return** $currentMax$	1
Total	$3n - 1$

What constitutes a fast algorithm ?

$O(n^x)$ is considered fast. $(x > 0)$

$O(x^n)$ is considered slow. $(x > 1)$

What constitutes a fast algorithm ?

$O(n^x)$ is considered fast. $(x > 1)$

$O(x^n)$ is considered slow.

How about $f(n) = n^{500}$?

What constitutes a fast algorithm ?

$O(n^x)$ is considered fast. $(x > 1)$

$O(x^n)$ is considered slow.

How about $f(n) = n^{500}$?

Yes, it is also fast !!

Relatives of Big-Oh

- **big-Omega**
  - f(n) is $\Omega(g(n))$ if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

Relatives of Big-Oh

- **big-Theta**
  - $f(n)$ is $\Theta(g(n))$ if there are constants $c' > 0$ and $c'' > 0$ and an integer constant $n_0 \geq 1$ such that $c' \cdot g(n) \leq f(n) \leq c'' \cdot g(n)$ for $n \geq n_0$

Relatives of Big-Oh

- **big-Theta**
  - $f(n)$ is $\Theta(g(n))$ if there are constants $c' > 0$ and $c'' > 0$ and an integer constant $n_0 \geq 1$ such that $c' \bullet g(n) \leq f(n) \leq c'' \bullet g(n)$ for $n \geq n_0$

We say, f(n) and g(n) have same growth rate

Remember

f(x) is Θ(g(x)) if and only if

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = c$$

Relatives of Big-Oh

- **little-oh**
  - $f(n)$ is $o(g(n))$ if, for any constant $c > 0$, there is an integer constant $n_0 \geq 0$ such that $f(n) \leq c \bullet g(n)$ for $n \geq n_0$

Relatives of Big-Oh

little-oh

- f(n) is o(g(n)) if, for any constant $c > 0$, there is an integer constant $n_0 \geq 0$ such that $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

We say, g(n) is faster than f(n)

Remember

f(x) if o(g(x)) if and only if

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 0$$

Relatives of Big-Oh

- **little-omega**
  - $f(n)$ is $\omega(g(n))$ if, for any constant $c > 0$, there is an integer constant $n_0 \geq 0$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

Relatives of Big-Oh

- **little-omega**
  - f(n) is ω(g(n)) if, for any constant $c > 0$, there is an integer constant $n_0 \geq 0$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

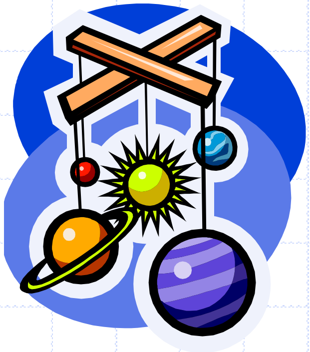
We say, g(n) is slower than f(n)

Remember

f(x) if ω(g(x)) if and only if

$$\lim_{x\to\infty}\frac{f(x)}{g(x)}=\infty$$

Example Uses of the Relatives of Big-Oh



- **$5n^2$ is $\Omega(n^2)$**

  f(n) is Ω(g(n)) if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

  let $c = 5$ and $n_0 = 1$
- **$5n^2$ is $\Omega(n)$**

  f(n) is Ω(g(n)) if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

  let $c = 1$ and $n_0 = 1$
- **$5n^2$ is $\omega(n)$**

  f(n) is ω(g(n)) if, for any constant $c > 0$, there is an integer constant $n_0 \geq 0$ such that $f(n) \geq c \cdot g(n)$ for $n \geq n_0$

  need $5n_0^2 \geq c \cdot n_0 \rightarrow$ given c, the n_0 that satifies this is $n_0 \geq c/5 \geq 0$