CS 240

Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

October 7, 2011

Counting Steps

In Code

Primitive operations on most modern processors include:

- Arithmetic (e.g., +, −, ∗, /)
- Conditionals (e.g., **if**, ==)
- Fetching/storing a single location in memory (e.g., setting a variable)

Example (The Searching Problem)

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times	Worst
`for(int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$	
`  if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$	
`    return true;`	c_3	1 or 0	
`return false;`	c_4	1 or 0	

Counting Steps

In Code

Primitive operations on most modern processors include:

- Arithmetic (e.g., +, −, ∗, /)
- Conditionals (e.g., **`if`**, ==)
- Fetching/storing a single location in memory (e.g., setting a variable)

Example (The Searching Problem)

- Let $t_i = \#$ times **`for`** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times	Worst
`for(int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$	n
`  if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$	
`    return true;`	c_3	1 or 0	
`return false;`	c_4	1 or 0	

Counting Steps

In Code

Primitive operations on most modern processors include:

- Arithmetic (e.g., +, −, ∗, /)
- Conditionals (e.g., **if**, ==)
- Fetching/storing a single location in memory (e.g., setting a variable)

Example (The Searching Problem)

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times	Worst
`for(int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$	n
`  if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$	n
`    return true;`	c_3	1 or 0	
`return false;`	c_4	1 or 0	

Counting Steps

In Code

Primitive operations on most modern processors include:

- Arithmetic (e.g., +, −, ∗, /)
- Conditionals (e.g., **`if`**, `==`)
- Fetching/storing a single location in memory (e.g., setting a variable)

Example (The Searching Problem)

- Let $t_i = \#$ times **`for`** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times	Worst
`for(int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$	n
`  if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$	n
`    return true;`	c_3	1 or 0	1
`return false;`	c_4	1 or 0	

Counting Steps

In Code

Primitive operations on most modern processors include:

- Arithmetic (e.g., +, −, ∗, /)
- Conditionals (e.g., **if**, ==)
- Fetching/storing a single location in memory (e.g., setting a variable)

Example (The Searching Problem)

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

Code	Cost	Times	Worst
`for(int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$	n
`if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$	n
`return true;`	c_3	1 or 0	1
`return false;`	c_4	1 or 0	1

Worst-Case Analysis

Problem: counting the *precise* number of steps is laborious...

Example

	Cost	Times	Worst
`for(i=0; i<n; i++)`	c_1	n	n
`  for(j=i; j<n; j++)`	c_2	$\sum_{i=0}^{n-1} n - i$	???
`    // ...`			

Worst-Case Analysis

Problem: counting the *precise* number of steps is laborious...

Example

	Cost	Times	Worst
`for(i=0; i<n; i++)`	c_1	n	n
`  for(j=i; j<n; j++)`	c_2	$\sum_{i=0}^{n-1} n - i$	$\leq n^2$
`    // ...`			

How Precise Do We Need To Be?

- Already throw away constant number of operations (they vary)
- What happens when n grows very large?
  - $n^2 - 10$?
  - $n^{100} - n$?
  - $n^3/1000 - 100n^2 - 100n + 3$?

Definition (Big-O Notation)

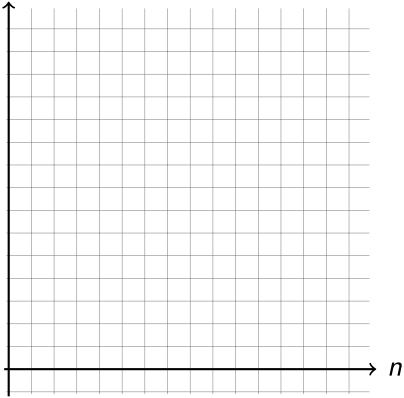
To consider the order of growth of a function, we classify it with O:

$$f \in O(g) \iff \exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5\log_2 n$ $\quad 10$ $\quad 2n + 5$

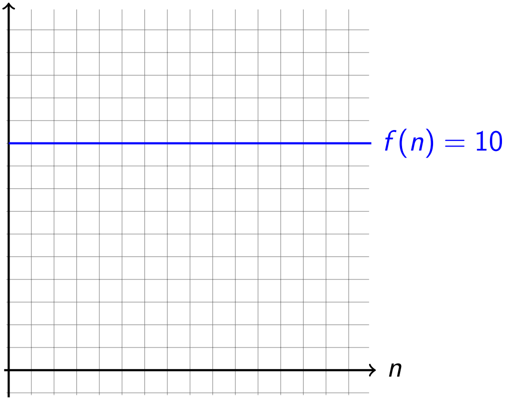


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5 \log_2 n \quad$ 10 $\quad 2n + 5$

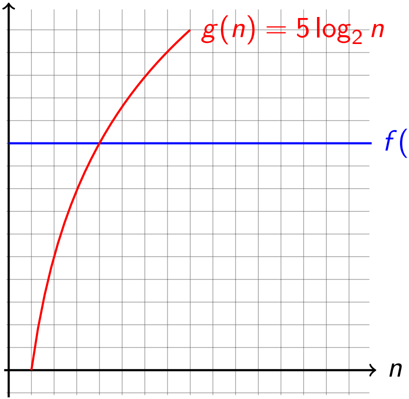


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad$ $5 \log_2 n$ $\quad$ 10 $\quad$ $2n + 5$

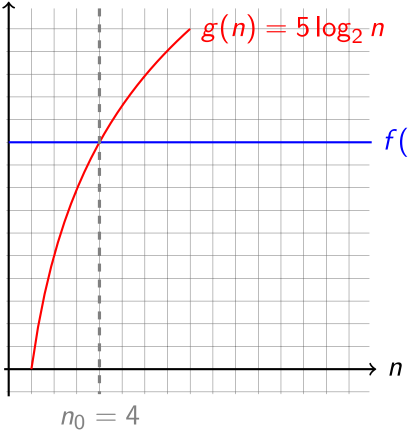


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad$ $5\log_2 n$ $\quad$ 10 $\quad$ $2n + 5$

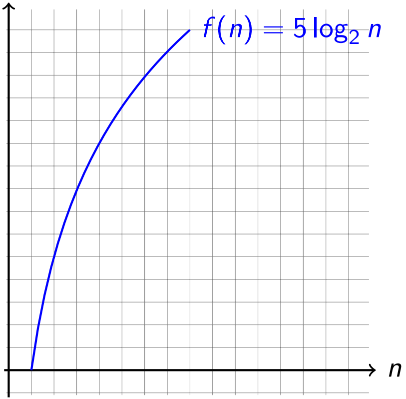


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5\log_2 n$ $\quad 10$ $\quad 2n + 5$



$\exists c > 0$

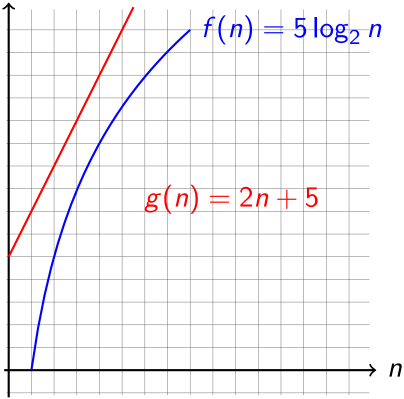
$\exists n_0 \geq 0$

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

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5\log_2 n \quad 10 \quad 2n + 5$

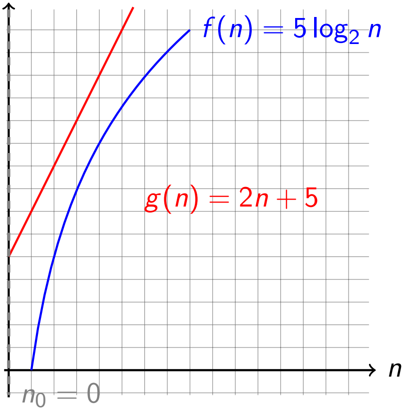


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad$ $5 \log_2 n$ $\quad$ 10 $\quad$ $2n + 5$



$\exists c > 0$

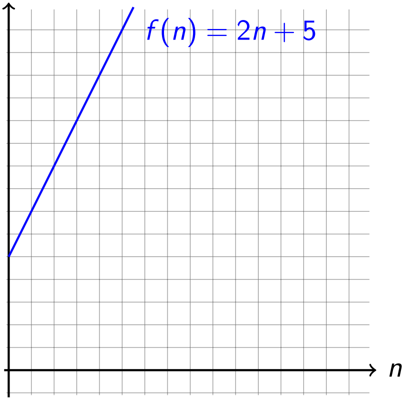
$\exists n_0 \geq 0$

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

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5 \log_2 n$ $\quad 10$ $\quad 2n + 5$

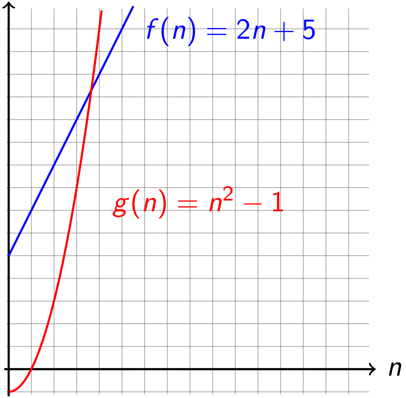


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5\log_2 n \quad 10 \quad 2n + 5$

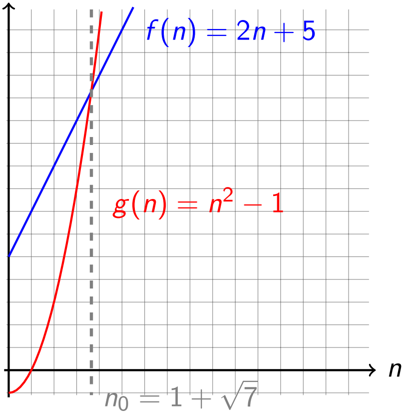


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $\quad 5\log_2 n$ $\quad 10$ $\quad 2n + 5$



$\exists c > 0$

$\exists n_0 \geq 0$

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

Example

Rearrange the following functions so that each is O of the next

$n^2 - 1$ $5\log_2 n$ 10 $2n + 5$

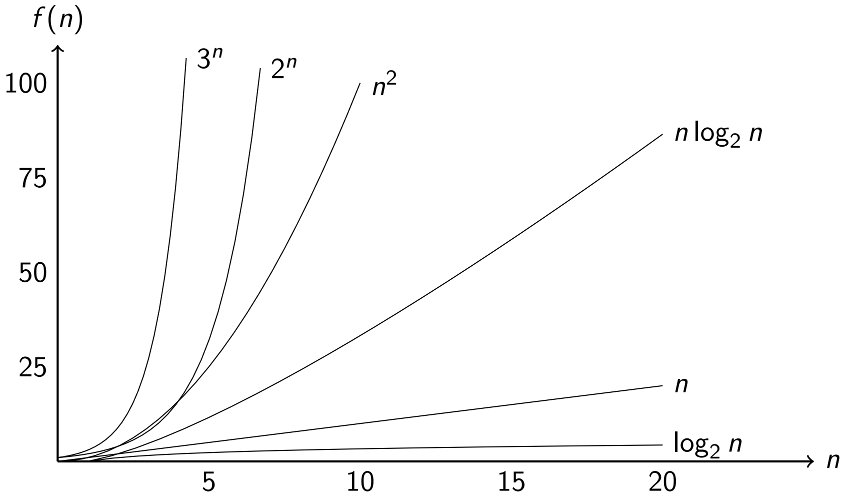
So,

$$
\begin{aligned}
10 &\in O(5\log_2 n)\\
5\log_2 n &\in O(2n+5)\\
2n+5 &\in O(n^2-1)
\end{aligned}
$$

and the proper arrangement is

10 $5\log_2 n$ $2n + 5$ $n^2 - 1$

Common Functions



Theorems About O

- O is reflexive. I.e., $\forall f, f \in O(f)$.
- O is transitive. I.e., $f \in O(g) \land g \in O(h) \implies f \in O(h)$.
- $f \in O(g) \implies f(n) + g(n) \in O(g)$.
- $f \in O(f') \land g \in O(g') \implies f(n) \cdot g(n) \in O(f'(n) \cdot g'(n))$
- $kf(n) + c \in O(f)$.