Order Notation and Time Complexity

The computing scientist's main challenge is not to get confused by the complexities of his own making. (E. W. Dijkstra)

Controlling complexity is the essence of computer programming. (Brian Kernigan)

If it doesn't work, it doesn't matter how fast it doesn't work. (Mich Ravera)

Definitions

- *Problem Size.* The number of elements in an algorithm's input. Usually denoted as n.
- *Time Complexity.* The number of steps needed by an algorithm to solve a problem of size n.
- *Space Complexity.* The memory needed by an algorithm to solve a problem of size n.
- *Worst-Case Analysis.* An upper bound on the time or space complexity of an algorithm.
- *Big-Oh Notation.* Used to summarize the worst-case complexity of an algorithm to within a constant factor.

Time Complexity

Complexity of Finding the Maximum

Count the number of times each operation is performed. Assume $n \geq 1$.

	Execution Count
procedure $maximum(a_1, \ldots, a_n)$	
$max := a_1$	1
$i := 2$	1
while $i \leq n$	n
if $max < a_i$ **then**	$n-1$
$max := a_i$	0 to $n-1$
$i := i + 1$	$n-1$
return max	1

Total: $3n + 1$ to $4n$

Linear Search

Assume $n \geq 1$.

procedure $linear\_search$ $(k, a_1, \ldots, a_n)$	
$loc := 0$	1
$i := 1$	1
while $i \leq n \wedge loc = 0$	2 to $n+1$
if $k = a_i$ **then**	1 to n
$loc := i$	0 or 1
else $i := i + 1$	0 to n
return loc	1

Total: 7 (while twice, others once except else) to $3n + 4$ (loc stays 0)

Bubble Sort

procedure $bubble\_sort$ $(a_1, \ldots, a_n)$	
$i := 1$	1
while $i < n$	n
$j := 1$	$n-1$
while $j < n$	$(n-1)n$
if $a_j > a_{j+1}$ **then**	$(n-1)^2$
$temp := a_j$	0 to $(n-1)^2$
$a_j := a_{j+1}$	0 to $(n-1)^2$
$a_{j+1} := temp$	0 to $(n-1)^2$
$j := j + 1$	$(n-1)^2$
$i := i + 1$	$n-1$

Total: $3n^2 - 2n + 1$ to $6n^2 - 8n + 4$

Big-Oh Notation

6

Definitions

- Big-Oh notation provides an upper bound on a function to within a constant factor.
- Let f and g be functions from nonnegative numbers to nonnegative numbers.
- $f(n)$ is $O(g(n))$ if there are positive constants C and k such that: $f(n) \leq C * g(n)$ whenever $n > k$
- $f(n)$ is $O(g(n))$ is equivalent to: $\exists C \exists k \forall n \, (n > k \rightarrow f(n) \leq C * g(n))$
- To prove big-Oh, find *witnesses*, specific values for C and k, and prove $n > k$ implies $f(n) \leq C * g(n)$.

One Approach for Finding Witnesses

- Generate a table for $f(n)$ and $g(n)$. using $n = 1$, $n = 10$ and $n = 100$. [Use values smaller than 10 and 100 if you wish.]
- Guess $C = \lceil f(1)/g(1) \rceil$ (or $C = \lceil f(10)/g(10) \rceil$).
- Check that $f(10) \leq C * g(10)$ and $f(100) \leq C * g(100)$. [If this is not true, $f(n)$ might not be $O(g(n))$.]
- Choose $k = 1$ (or $k = 10$).
- Prove that $\forall n (n > k \rightarrow f(n) \leq C * g(n))$. [It's ok if you end up with a larger, but still constant, value for C.]

Tricks for Proving Big-Oh

- Assume $n > 1$ if you chose $k = 1$ (or $n > 10$ if you chose $k = 10$).
- To prove $f(n) \leq C * g(n)$, you need to find expressions larger than $f(n)$ and smaller than $C * g(n)$.
- If the lowest-order term is negative, just eliminate it to obtain a larger expression.
- Repeatedly use $n > k$ and $2n > 2k$ and $3n > 3k$ and so on to "convert" the lowest-order term into a higher-order term.
- Check that your expressions are less than $C * g(n)$ by using $n = 100$.

0877425531509

Proving Big-Oh: Example 1

Show that $3n + 7$ is $O(n)$. In this case, $f(n) = 3n + 7$ and $g(n) = n$.

n	$f(n)$	$g(n)$	$\lceil f(n)/g(n) \rceil$
1	10	1	10
10	37	10	4
100	307	100	4

This table suggests trying $k = 1$ and $C = 10$ or $k = 10$ and $C = 4$.

Proving either one is good enough to prove big-Oh.

Example 1, Slide 2

Try $k = 1$ and $C = 10$.
Want to prove $n > 1$ implies $3n + 7 \leq 10n$.

Assume $n > 1$. Want to show $3n + 7 \leq 10n$.

7 is the lowest-order term, so work on that first.
$n > 1$ implies $7n > 7$, which implies
$3n + 7 < 3n + 7n = 10n$.

This finishes the proof. A short version is:
$n > 1$ implies $3n + 7 < 3n + 7n = 10n$.

Example 1, Slide 3

Try $k = 10$ and $C = 4$.
Want to prove $n > 10$ implies $3n + 7 \leq 4n$.

Assume $n > 10$. Want to show $3n + 7 \leq 4n$.

Work on the lowest-order term first.
$n > 10$ implies $n > 7$, which implies
$3n + 7 < 3n + n = 4n$.

This finishes the proof. A short version is:
$n > 10$ implies $3n + 7 < 3n + n = 4n$.

Proving Big-Oh: Example 2

Show that $n^2 + 2n + 1$ is $O(n^2)$. In this case, $f(n) = n^2 + 2n + 1$ and $g(n) = n^2$.

n	$f(n)$	$g(n)$	$\lceil f(n)/g(n) \rceil$
1	4	1	4
10	121	100	2
100	10201	10000	2

This table suggests trying $k = 1$ and $C = 4$ or $k = 10$ and $C = 2$.

Again, proving either one is good enough to prove big-Oh.

Example 2, Slide 2

Try $k = 1$ and $C = 4$.
Want to prove $n > 1$ implies $n^2 + 2n + 1 \leq 4n^2$.

Assume $n > 1$. Want to show $n^2 + 2n + 1 \leq 4n^2$.

Work on the lowest-order term first.
$n > 1$ implies $n^2 + 2n + 1 < n^2 + 2n + n = n^2 + 3n$

Now $3n$ is the lowest-order term.
$n > 1$ implies $3n > 3$ and $3n^2 > 3n$, which implies
$n^2 + 3n < n^2 + (3n)n = n^2 + 3n^2 = 4n^2$.

This finishes the proof. A short version is:
$n > 1$ implies $n^2 + 2n + 1 < n^2 + 3n < 4n^2$.

Example 2, Slide 3

Try $k = 10$ and $C = 2$.
Want to prove $n > 10$ implies $n^2 + 2n + 1 \leq 2n^2$.

Assume $n > 10$. Want to show $n^2 + 2n + 1 \leq 2n^2$.

Work on the lowest-order term first.
$n > 10$ implies $n > 1$, which implies
$n^2 + 2n + 1 < n^2 + 2n + n = n^2 + 3n$.

Now $3n$ is the lowest-order term.
$n > 10$ implies $n > 3$ and $n^2 > 3n$, which implies $n^2 + 3n < n^2 + n^2 = 2n^2$.

This finishes the proof. A short version is:
$n > 10$ implies $n^2 + 2n + 1 < n^2 + 3n < n^2 + n^2 = 2n^2$.

Proving Big-Oh: Example 3

Show that $(n + 1)^3$ is $O(n^3)$. In this case, $f(n) = (n + 1)^3$ and $g(n) = n^3$.

n	$f(n)$	$g(n)$	$\lceil f(n)/g(n) \rceil$
1	8	1	8
10	1331	1000	2
100	1030301	1000000	2

This table suggests trying $k = 1$ and $C = 8$ or $k = 10$ and $C = 2$.

Again, proving either one is good enough to prove big-Oh.

Example 3, Slide 2

Try $k = 1$ and $C = 8$.
Want to prove $n > 1$ implies $(n + 1)^3 \leq 8n^3$.

Assume $n > 1$. Want to show $(n + 1)^3 \leq 8n^3$.

$n > 1$ implies $(n + 1)^3 < (n + n)^3 = (2n)^3$.
It's a mistake to infer $(2n)^3$ and $2n^3$ are equal.

$(2n)^3 = (2n) * (2n) * (2n) = 8n^3$

This finishes the proof. A short version is:
$n > 1$ implies $(n + 1)^3 < (n + n)^3 = (2n)^3 = 8n^3$

Example 3, Slide 3

Try $k = 10$ and $C = 2$.
Want to prove $n > 10$ implies $(n+1)^3 \leq 2n^3$.
We will need $(n+1)^3 = n^3 + 3n^2 + 3n + 1$.

Assume $n > 10$. Want to show $(n+1)^3 \leq 2n^3$.

$n > 10$ implies $n^3 + 3n^2 + 3n + 1 < n^3 + 3n^2 + 4n$.

$n > 10$ implies
$n^3 + 3n^2 + 4n < n^3 + 3n^2 + n * n = n^3 + 4n^2$.

$n > 10$ implies $n^3 + 4n^2 < n^3 + n * n^2 = 2n^3$.

This finishes the proof. A short version is:
$n > 10$ implies $(n+1)^3 = n^3 + 3n^2 + 3n + 1 < n^3 + 3n^2 + 4n < n^3 + 4n^2 < 2n^3$

Proving Big-Oh: Example 4

Show that $8n^3 - 12n^2 + 6n - 1$ is $O(n^3)$. In this case, $f(n) = 8n^3 - 12n^2 + 6n - 1$ and $g(n) = n^3$.

n	$f(n)$	$g(n)$	$\lceil f(n)/g(n) \rceil$
1	1	1	1
10	6859	1000	7
100	7,880,599	1,000,000	8
1000	7,988,005,999	1,000,000,000	8

$C = 1$ and $C = 7$ are too small for $n = 100$. $C = 8$ works for $n = 1000$ and the other values of n, so try $k = 100$ and $C = 8$.
[$k = 1$ and $k = 10$ would also work.]

Example 4, Slide 2

Try $k = 100$ and $C = 8$. Want to prove
$n > 100$ implies $8n^3 - 12n^2 + 6n - 1 \leq 8n^3$

Assume $n > 100$. Want to show $f(n) \leq 8n^3$.

The lowest-order term is negative, so eliminate it.
$8n^3 - 12n^2 + 6n - 1 < 8n^3 - 12n^2 + 6n$.

$n > 100$ implies $n > 6$, which implies
$8n^3 - 12n^2 + 6n < 8n^3 - 12n^2 + (n) * n = 8n^3 - 11n^2$.

Now lowest-order term is negative, so eliminate.
$n > 100$ implies $8n^3 - 11n^2 \leq 8n^3$.

This finishes the proof. A short version is:
$n > 100$ implies $8n^3 - 12n^2 + 6n - 1 < 8n^3 - 12n^2 + 6n < 8n^3 - 11n^2 < 8n^3$.

Big-Omega Notation

Definitions

- Big-Omega notation provides a lower bound on a function to within a constant factor.
- Let f and g be functions from nonnegative numbers to nonnegative numbers.
- $f(n)$ is $\Omega(g(n))$ if there are positive constants C and k such that:
  $f(n) \geq C * g(n)$ whenever $n > k$
- $f(n)$ is $\Omega(g(n))$ is equivalent to
  $\exists C \, \exists k \, \forall n \, (n > k \rightarrow f(n) \geq C * g(n))$
- To prove big-Omega, find *witnesses*, specific values for C and k, and prove $n > k$ implies $f(n) \geq C * g(n)$.

Tricks for Proving Big-Omega

- Assume $n > 1$ if you chose $k = 1$ (or $n > 10$ if you chose $k = 10$).
- To prove $f(n) \geq C * g(n)$, you need to find expressions smaller than $f(n)$ and larger than $C * g(n)$.
- If the lowest-order term is positive, just eliminate it to obtain a larger expression.
- Repeatedly use $-k > -n$ and $-0.1k > -0.1n$ and so on to "convert" the lowest-order term into a higher-order term.
- Check that your expressions are greater than $C * g(n)$ by using $n = 100$.

Finding Big-Omega Witnesses

- Generate a table for $f(n)$ and $g(n)$. using $n = 1$, $n = 10$ and $n = 100$.
  [Use values smaller than 10 and 100 if you wish.]
- Guess $1/C = \lceil g(1)/f(1) \rceil$
  (or more likely $1/C = \lceil g(10)/f(10) \rceil$).
- Check that $f(10) \geq C * g(10)$ and $f(100) \geq C * g(100)$.
  [If this is not true, $f(n)$ might not be $\Omega(g(n))$.]
- Choose $k = 1$ (or $k = 10$).
- Prove that $\forall n(n > k \rightarrow f(n) \geq C * g(n))$.
  [It's ok if you end up with a smaller, but still positive, value for C.]

Proving Big-Omega: Example 1

Show that $3n + 7$ is $\Omega(n)$. In this case, $f(n) = 3n + 7$ and $g(n) = n$.

n	$f(n)$	$g(n)$	$\lceil g(n)/h(n) \rceil$	$C?$
1	10	1	1	1
10	37	10	1	1
100	307	100	1	1

This table suggests trying $k = 1$ and $C = 1$.
Want to prove $n > 1$ implies $3n + 7 \geq n$.

Short proof: $n > 1$ implies $3n + 7 > 3n > n$.

Proving Big-Omega: Example 2

Show that $n^2 - 2n + 1$ is $\Omega(n^2)$. In this case, $f(n) = n^2 - 2n + 1$ and $g(n) = n^2$.

n	$f(n)$	$g(n)$	$\lceil g(n)/f(n) \rceil$	$C?$
1	1	1	1	1
10	81	100	2	1/2
100	9801	10000	2	1/2

This table suggests trying $k = 10$ and $C = 1/2$. Trying $C = 1$ is too large.

Example 2, Slide 2

Try $k = 10$ and $C = 1/2$.
Want to prove $n > 10$ implies $n^2 - 2n + 1 \geq n^2/2$.

Assume $n > 10$. Want to show $f(n) \geq n^2/2$.

The lowest-order term is positive, so eliminate.
$n^2 - 2n + 1 > n^2 - 2n$

$n > 10$ implies $-10 > -n$, implies $-2 > -0.2n$.
$-2 > -0.2n$ implies $n^2 - 2n > n^2 - 0.2n^2 = 0.8n^2$.

$n > 10$ implies $0.8n^2 > n^2/2$.

This finishes the proof. A short version is:
$n > 10$ implies $n^2 - 2n + 1 > n^2 - 2n > 0.8n^2 > n^2/2$

Proving Big-Omega: Example 3

Show that $(n-1)^3$ is $\Omega(n^3)$. In this case, $f(n) = (n-1)^3$ and $g(n) = n^3$.

n	$f(n)$	$g(n)$	$\lceil g(n)/f(n) \rceil$	$C?$
1	0	1	undefined	undefined
10	729	1000	2	1/2
100	970299	1000000	2	1/2

This table suggests trying $k = 10$ and $C = 1/2$.
Trying $k = 1$ does not look like a good idea.

Example 3, Slide 2

Try $k = 10$ and $C = 1/2$.
Want to prove $n > 10$ implies $(n-1)^3 \geq n^3/2$.
We will need $(n-1)^3 = n^3 - 3n^2 + 3n - 1$.

Assume $n > 10$. Want to show $(n-1)^3 \geq n^3/2$.

$n > 10$ implies $n^3 - 3n^2 + 3n - 1 > n^3 - 3n^2 + 3n - 0.1 * n > n^3 - 3n^2 + 2n$.

$n > 10$ implies $n^3 - 3n^2 + 2n > n^3 - 3n^2$.

$n > 10$ implies $n^3 - 3n^2 > n^3 - 0.3n^3 = 0.7n^3$.

$n > 10$ implies $0.7n^3 > n^3/2$.

Example 3, Slide 3

This finishes the proof. A short version is:
$n > 10$ implies
$(n-1)^3$
$= n^3 - 3n^2 + 3n - 1$
$> n^3 - 3n^2 + 2n$
$> n^3 - 3n^2$
$> 0.7n^3$
$> n^3/2$

Proving Big-Omega: Example 4

Show that $n^3/8 - n^2/12 - n/6 - 1$ is $O(n^3)$. In this case, $f(n) = n^3/8 - n^2/12 - n/6 - 1$ and $g(n) = n^3$.

n	$f(n)$	$g(n)$	$\lceil g(n)/f(n) \rceil$	$C?$
1	-0.8	1	-1	-1
10	117.3	1,000	9	1/9
100	124,182.3	1,000,000	9	1/9

$C = -1$ is useless, so try $k = 10$ and $C = 1/9$.

Example 4, Slide 2

Try $k = 10$ and $C = 1/9$. Want to prove
$n > 10$ implies $n^3/8 - n^2/12 - n/6 - 1 \geq n^3/9$

Assume $n > 10$, which implies the following:

$n^3/8 - n^2/12 - n/6 - 1$
$= (3n^3 - 2n^2 - 4n - 24)/24$
$> (3n^3 - 2n^2 - 4n - 2.4n)/24$
$> (3n^3 - 2n^2 - 7n)/24$
$> (3n^3 - 2n^2 - 0.7n^2)/24$
$> (3n^3 - 3n^2)/24$
$> (3n^3 - 0.3n^3)/24$
$> (3n^3 - n^3)/24$
$= (2n^3)/24 = n^3/12$

Ended up with $k = 10$ and $C = 1/12$, proving
$n > 10$ implies $n^3/8 - n^2/12 - n/6 - 1 \geq n^3/12$

Proving Not Big-Oh

How to Prove Big-Oh is False

$f(n)$ is not $O(g(n))$ is equivalent to
$\forall C \, \forall k \, \exists n \, (n > k \wedge f(n) > C * g(n))$

Need to prove for all values of C and k.

One approach is to show $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$.

Another approach is to show $f(n)$ is $\Omega(n * g(n))$.
[More generally, to show $f(n)$ is $\Omega(h(n) * g(n))$, where $h(n)$ is not $O(1)$.]

Not Big-Oh Theorem

Thm: If $f(n)$ is $\Omega(n * g(n))$, then $f(n)$ is not $O(g(n))$.

Assume $f(n)$ is $\Omega(n * g(n))$ and $f(n)$ is $O(g(n))$. Want to show a contradiction. The approach is to find a value of n that violates the definitions.

Proof: $f(n)$ is $\Omega(n * g(n))$ implies there exists positive values C and k such that $n > k$ implies $f(n) \geq C * n * g(n)$.

$f(n)$ is $O(g(n))$ implies there exists positive values C' and k' such that $n > k'$ implies $f(n) \leq C * g(n)$.

Not Big-Oh Theorem

Let $n = k + k' + C'/C$.

$k > 0$ and $k' > 0$ and $C > 0$ and $C' > 0$ imply $n > k$ and $n > k'$ and $C * n > C'$.

$f(n)$ is $\Omega(n * g(n))$ and $n > k$ and $C * n > C'$ imply $f(n) \geq C * n * g(n) > C' * g(n)$.

$f(n)$ is $O(g(n))$ and $n > k'$ imply $f(n) \leq C' * g(n)$.

$f(n) > C' * g(n)$ contradicts $f(n) \leq C' * g(n)$.