CS4800: Algorithms & Data
Jonathan Ullman

Lecture 2:

- Asymptotic Order of Growth
- Divide and Conquer: Karatsuba's Algorithm

Jan 9, 2018

Ask the Audience!

$\mathbb{N} = \{1, 2, 3, \ldots\}$

- Review Question: Prove by induction that $\forall\, n \in \mathbb{N}$, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- Proof by induction on n):
  - Base Case: ??? $(n=1)$ $\sum_{i=1}^{1} i = 1 = \frac{1 \cdot 2}{2} = 1$ ✓
  - Inductive Step: ???

    We'll assume that it's true for $n<k$ and prove that it's true for $n=k$. (No matter what k we choose)

    $$(1+2+\ldots+k-1+k) \underset{\text{IH}}{=} \frac{(k-1)\cdot k}{2} + k = \frac{k\cdot(k+1)}{2}$$
  - ???

    Since the inductive step holds $\forall k$, the stmt is true by induction. $\square$

Asymptotic Order Of Growth

- Want to **compare** running times of algorithms

- Computing running time exactly is **difficult**:
  - Counting exact number of operations is tedious
  - Different algorithms use different "operations," exact running time depends on hardware/language

Want a way of reasoning about running time that:

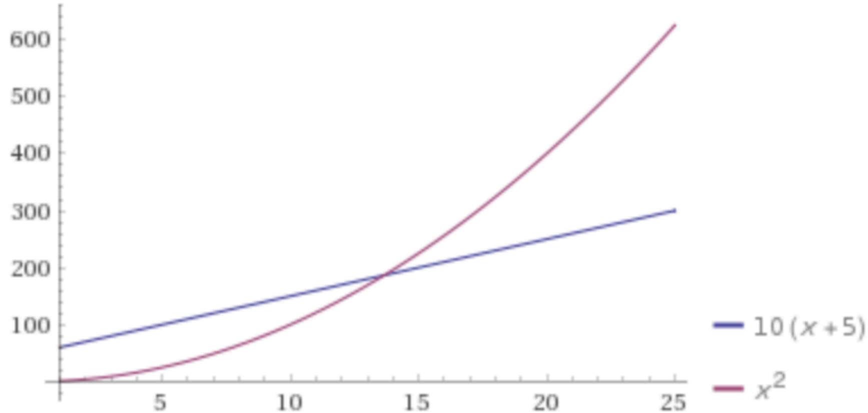
① Is simple.

② Is not specific to a particular machine.

Asymptotic Order Of Growth

- Want to **compare** running times of algorithms

- Often care more about **large inputs**:

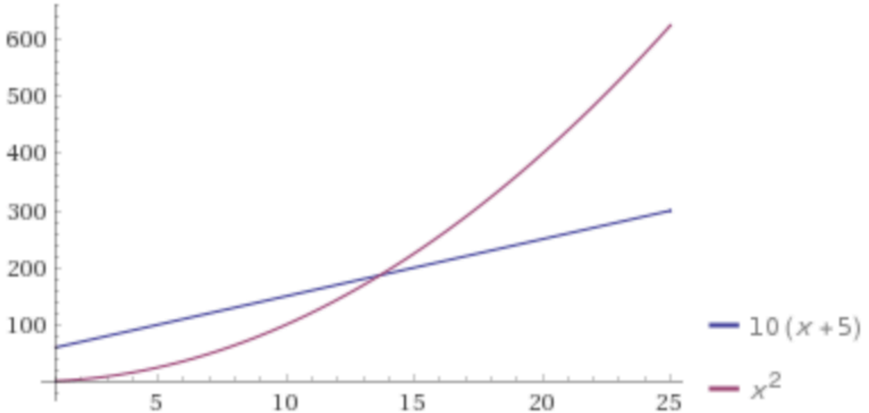


Want: ③ To reason about how running time scales.

Asymptotic Order Of Growth

- Want to **compare** running times of algorithms

- Asymptotic Analysis: How does the running time behave as the input size grows?

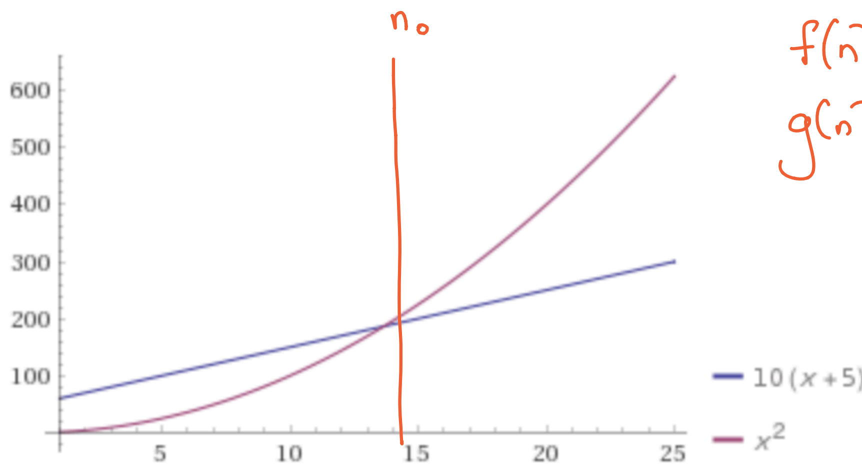


$f(n) \in O(g(n))$

Asymptotic Order Of Growth

makes no sense

- "Big-Oh": $f(n) = O(g(n))$ if there are constants $c > 0$ and n_0 s.t. $f(n) \le c \cdot g(n)$ for every $n \ge n_0$.
  - Analogous to saying $f(n) \le g(n)$
  - "For large n, $f(n)$ grows no faster than $g(n)$"



$f(n) = 10n + 50$

$g(n) = n^2$

Ask the Audience!

- "Big-Oh": $f(n) = O(g(n))$ if there are constants $c > 0$ and n_0 s.t. $f(n) \le c \cdot g(n)$ for every $n \ge n_0$.

- Which of these are true?
  - (a) $3n^2 + 100n = O(n^2)$
  - (b) $n^3 = O(n^2)$
  - (c) $2^n = O(n)$
  - (d) $n = O(2^n)$

$f(n) = 3n^2 + 100n$

$g(n) = n^2$

$\forall n \ge \underset{n_0}{100}$ $\qquad f(n) \le \underset{c}{4}\, g(n)$

$3n^2 + 100n \le 4n^2$

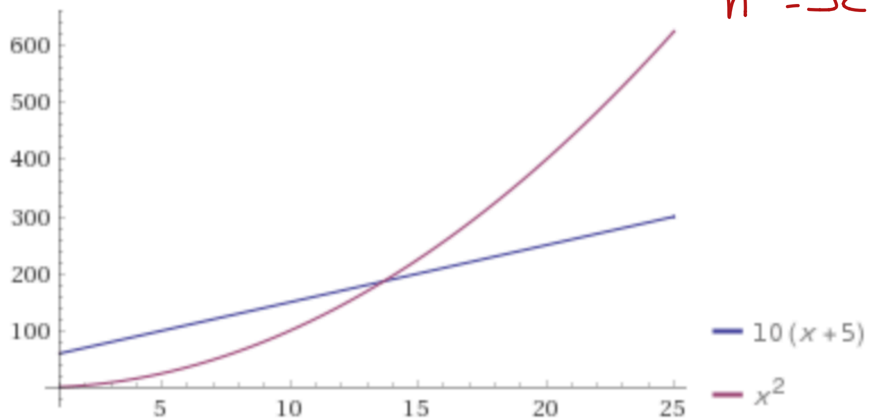
$100n \le n^2$

$100 \le n$

Asymptotic Order Of Growth

- "Big-Omega": $f(n) = \Omega(g(n))$ if there are const's $c > 0$ and n_0 s.t. $f(n) \boxed{\geq} c \cdot g(n)$ for all $n \geq n_0$.
  - Analogous to saying $f(n) \geq g(n)$
  - "For large n, $f(n)$ grows at least as fast as $g(n)$"

$n^2 = \Omega(10n+50)$



Asymptotic Order Of Growth

- "Big-Theta": $f(n) = \Theta(g(n))$ if and only if $f(n) = O\big(g(n)\big)$ and $f(n) = \Omega(g(n))$.
  - Analogous to saying $f(n) = g(n)$
  - "For large n, $f(n)$ grows at the same rate as $g(n)$"

- Roughly like saying $\exists c > 0$ s.t.

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

- Example: $f(n) = 10n + 50$

$g(n) = n$

$$\frac{10n+50}{n} = 10 + \frac{50}{n}$$

$$10n + 50 = \Theta(n)$$

Take Home Message:

If an algorithm uses $f(n)$ "operations" and $f(n) = \Theta(g(n))$ then we say the algorithm "runs in $\Theta(g(n))$ time."

Ask the Audience!

- "Big-Oh": $f(n) = O(g(n))$ if there are constants $c > 0$ and n_0 s.t. $f(n) \leq c \cdot g(n)$ for every $n \geq n_0$

- Consider the statement: "either $f(n) = O(g(n))$ or $g(n) = O(f(n))$ or both."

- Is this statement:
  - (a) True for all f, g?
  - (b) True for some f, g and not others?
  - (c) Never true for any f, g?

Asymptotics Rules of Thumb

- Constant factors can be ignored ✓
  - $100n = \Theta(n)$
- If $a > b$ then n^a grows faster than n^b

  polynomial
  - $n^2 = O(n^3), \quad n^3 \neq O(n^2)$
- Any exponential grows faster than any polynomial
  - $n^4 = O(2^n), \quad 2^n \neq O(n^4)$ $\quad n^{1000} = O(1.000001^n)$
- Any polynomial grows faster than any logarithm
  - $\log_2^3 n = O(n^{1/3}), \quad n^{1/3} \neq O(\log_2^3 n)$
- Lower order terms don't matter
  - $n^2 + 42n = \Theta(n^2)$

$42n = O(n^2)$

$$\boxed{\begin{aligned} &a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0 \\ &= \Theta(n^k) \end{aligned}}$$

Ask the Audience!

$$\frac{n \log_2 n}{100n} = \frac{\log_2 n}{100}$$

$100n \leq n \log_2 n$ if $n \geq 2^{100}$

- "Big-Oh": $f(n) = O(g(n))$ if there are constants $c > 0$ and n_0 s.t. $f(n) \leq c \cdot g(n)$ for every $n \geq n_0$

- Rank the following functions in increasing order of growth (i.e. f_1, f_2, f_3, f_4 so that $f_i = O(f_{i+1})$)
  - $n \log_2 n$
  - $n^2 = \Theta(n^2)$
  - $100n = \Theta(n)$
  - $3^{\log_2 n} = O(n^{1.59})$ $\Omega(n^{1.58})$

$100n$, $n\log_2 n$, $3^{\log_2 n}$, n^2

$$3^{\log_2 n} = \left(2^{\log_2(3)}\right)^{\log_2 n}$$

$$= \left(2^{\log_2 n}\right)^{\log_2(3)} = n^{\log_2(3)} = n^{1.59}$$

Useful Facts:

- $a^{\log_b n} = n^{\log_b(a)}$
- $\log n$ grows slower than n^a for any a

More Asymptotics

$f(n) \le g(n)$

- "Big-Oh": $f(n) = O(g(n))$ if there are constants $c > 0$ and n_0 s.t. $f(n) \le c \cdot g(n)$ for every $n \ge n_0$

- "little-oh": $f(n) = o(g(n))$ if for every constant $c > 0$ there exists n_0 s.t. $f(n) \le c \cdot g(n)$ for every $n \ge n_0$.
    - Analogous to saying $f(n) < g(n)$
    - For large n, $f(n)$ grows slower than $g(n)$
    - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

More Asymptotics

- "Big-Omega": $f(n) = \Omega(g(n))$ if there are const's $c > 0$ and n_0 s.t. $f(n) \geq c \cdot g(n)$ for all $n \geq n_0$.

- "little-omega": $f(n) = \omega(g(n))$ if for every constant $c > 0$ there exists n_0 s.t. $f(n) \geq c \cdot g(n)$ for every $n \geq n_0$.
  - Analogous to saying $f(n) > g(n)$
  - For large n, $f(n)$ grows faster than $g(n)$

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

Examples

① $\forall 0<a<b, \quad n^a = o(n^b)$

② $\forall\ a>0 \quad \log n = o(n^a) \qquad \log n = o(n^{.00001})$

③ $\forall a\ \forall b>1 \quad n^a = o(b^n) \qquad n^{1000} = o(1.00001^n)$

④ $\boxed{\text{True}}$ / False: If $f(n) = o(g(n))$ then $g(n) \neq O(f(n))$.

$f(n) < g(n) \quad$ if $\quad f(n) \not\geq g(n)$

Why Asymptotics Matter

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	4 sec
$n = 30$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	18 min	10^{25} years
$n = 50$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	11 min	36 years	very long
$n = 100$	$<$ 1 sec	$<$ 1 sec	$<$ 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	$<$ 1 sec	$<$ 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	$<$ 1 sec	$<$ 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	$<$ 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

$100\, n \log_2 n$:

2000 sec

Next several lectures

Divide and Conquer Algorithms: Karatsuba's Algorithm

Addition

Input

x_{n-1}	x_{n-2}	$\cdots$			x_0

- Given n-digit numbers x, y output $z = x + y$

	1	4	6	8
+	1	1	2	2
=	2	5	9	0

- $n+1$ operations $= \Theta(n)$

adding two digits plus a possible carry.

Multiplication

- Given n-digit numbers x, y output $z = x \cdot y$

multiply by 10 are "free"

"Gradeschool Algorithm"

					1	2	3	4	
				x	1	1	2	2	
n operations					2	4	6	8	2×1234
n operations	+			2	4	6	8	0	$10 \times 2 \times 1234$
n operations	+		1	2	3	4	0	0	$100 \times 1 \times 1234$
n operations	+	1	2	3	4	0	0	0	$1000 \times 1 \times 1234$
		1	3	8	4	5	4	8	

n additions of $\leq 2n$-digit numbers

- Time is (n rows)(n operations) + ($n-1$ additions)($2n$ operations)

$= \Theta(n^2)$ time algorithm

Divide and Conquer

1. Break the problem into a small number of simpler "subproblems"
2. Recursively solve the subproblems
3. Combine the solutions to the subproblems

Ron Popeil

3M sold

Divide and Conquer Multiplication

	$\boxed{1 \quad 2}$	$\boxed{3 \quad 4}$	x	$x = 10^2 \cdot 12 + 34$
x	$\boxed{1 \quad 1}$	$\boxed{2 \quad 2}$	y	$y = 10^2 \cdot 11 + 22$

	a	b	x	$x = 10^{n/2}a + b$
x	c	d	y	$y = 10^{n/2}c + d$
	n/2 digits	n/2 digits		

n digits

$$x \cdot y = (10^{n/2} \cdot a + b)(10^{n/2} \cdot c + d)$$

$$= 10^n \cdot \underbrace{ac}_{1} + 10^{n/2}(\underbrace{ad}_{2} + \underbrace{bc}_{3}) + \underbrace{bd}_{4}$$

Divide and Conquer Multiplication

	1	2	3	4	x	$x = 10^2 \cdot 12 + 34$
x	1	1	2	2	y	$y = 10^2 \cdot 11 + 22$

	a	b	x	$x = 10^{n/2}a + b$
x	c	d	y	$y = 10^{n/2}c + d$

$$x \cdot y = (10^{n/2}a + b)(10^{n/2}c + d)$$

$$= 10^n ac + \ 10^{n/2}(ad + bc) + bd$$

- Need to do four $(n/2)$-digit mults, three n-digit adds
- Recurrence: $T(n) = 4T(n/2) + 3n$

~~Ask The Audience!~~

$T(1)=1$

$T(n) = 4T(\frac{n}{2}) + 3n$

- Conjecture: If $T(1) = 1$, $T(n) = 4T(n/2) + 3n$, then $\forall n$, $\geq$ $T(n) \geq 3n^2$.
- Proof (Induction on ℓ):
  - Base Case: ???
  - Inductive Step: ???
  - ???

$T(k) \geq 4 \cdot T(\frac{k}{2}) + 3k$

$= 4 \cdot (3 \cdot (\frac{k}{2})^2) + 3k = 3k^2 + 3k$

$\geq 3k^2$

- Therefore the divide and conquer algorithm requires $T(n) = \Omega(n^2)$, no faster than schoolbook

Karatsuba's Algorithm

	a	b
x	c	d

$$x = 10^{n/2}a + b$$
$$y = 10^{n/2}c + d$$

$$x \cdot y = 10^{n}ac + 10^{n/2}(ad + bc) + bd$$

1 2

- Key Identity
  - $(b - a)(c - d) = \boxed{ad + bc} - ac - bd$

    3

- Suffices to do only three $n/2$-digit mults!

Karatsuba's Algorithm

Karatsuba(x, y, n):

If $n = 1$ then return xy

Base Case

Else

x: $\boxed{a}\ \boxed{b}$

y: $\boxed{c}\ \boxed{d}$

$m \leftarrow \lceil n/2 \rceil$

write $x = 10^m a + b$, $y = 10^m c + d$

$e \leftarrow$ Karatsuba(a, c, m)

$f \leftarrow$ Karatsuba(b, d, m)

$g \leftarrow$ Karatsuba$(b - a, c - d, m)$

return $10^{2m} e + 10^m (e + f + g) + f$

$$10^n \cdot ac + 10^{n/2}\left(\underbrace{(b-a)(c-d) + ac + bd}_{= ad + bc}\right) + bd$$

Ask The Audience!

Karatsuba(x, y, n):

    If $n = 1$ then return xy

    Else

        $m \leftarrow \lceil n/2 \rceil$

        write $x = 10^m a + b$, $y = 10^m c + d$

        $e \leftarrow$ Karatsuba(a, c, m)

        $f \leftarrow$ Karatsuba(b, d, m)

        $g \leftarrow$ Karatsuba$(b - a, c - d, m)$

        return $10^{2m} e + 10^m (e + f + g) + f$

- Carry out Karatsuba's Algorithm for $52 \cdot 48$

Karatsuba's Algorithm

Karatsuba(x, y, n):
    If $n = 1$ then return xy
    Else
        $m \leftarrow \lceil n/2 \rceil$
        write $x = 10^m a + b,\ y = 10^m c + d$
        $e \leftarrow \text{Karatsuba}(a, c, m)$
        $f \leftarrow \text{Karatsuba}(b, d, m)$
        $g \leftarrow \text{Karatsuba}(b - a, c - d, m)$
        return $10^{2m} e + 10^m (e + f + g) + f$

- Recursive Calls: $3T(n/2)$
- Additional Work (additions, shifts): Cn for some C
- Recurrence: $T(n) = 3T(n/2) + Cn$

Karatsuba's Algorithm

$\Theta(n^{\log_2 3})$

$O(n^{1.59})$

- Recurrence: $T(n) = 3T(n/2) + Cn,\ T(1) \leq C$
- Guess the right solution: $T(n) \leq 3Cn^{\log_2 3} - 2Cn$

Karatsuba's Algorithm

- Recurrence: $T(n) = 3T(n/2) + Cn, T(1) \leq C$
- Guess the right solution: $T(n) \leq 3Cn^{\log_2 3} - 2Cn$
- Proof by Induction:
  - Base Case (n=1): $T(1) \leq C$ ✓
  - Inductive Step: Assume that it's true for all n < k, we'll prove it for n = k.

Use the IH

use the recurrence

$$
\begin{aligned}
T(k) &= \boxed{3T(k/2) + Ck} \\
&\leq 3\left(3C(k/2)^{\log_2 3} - 2C(k/2)\right) + Ck \\
&\leq 9C\frac{k^{\log_2 3}}{2^{\log_2 3}} - 3Ck + Ck \\
&= 3Ck^{\log_2 3} - 2Ck
\end{aligned}
$$

Karatsuba's Algorithm

Karatsuba(x, y, n):
  If $n = 1$ then return xy
  Else
    $m \leftarrow \lceil n/2 \rceil$
    write $x = 10^m a + b, y = 10^m c + d$
    $e \leftarrow \text{Karatsuba}(a, c, m)$
    $f \leftarrow \text{Karatsuba}(b, d, m)$
    $g \leftarrow \text{Karatsuba}(b - a, c - d, m)$
    return $10^{2m} e + 10^m (e + f + g) + f$

- Recurrence: $T(n) = 3T(n/2) + Cn$
- Running time: $T(n) = O\left(n^{\log_2 3}\right) = O(n^{1.59})$

Karatsuba

Karatsuba(x, y, n):
If n=1 then return xy
Else:
$m \leftarrow \lceil n/2 \rceil$
write $x = 10^m a + b,\ y = 10^m c + d$
$e \leftarrow Karatsuba(a, c, m)$
$f \leftarrow Karatsuba(b, d, m)$
$g \leftarrow Karatsuba(a + b, c + d, m)$
return $10^{2m} e + 10^m (g - e - f) + f$

Conjecture: For every n, and all n-digit x,y, Karatsuba(x,y,n) = xy.

- Proof by Induction
  - Base Case: (n=1) Karatsuba(x,y,1) = xy
  - Inductive Step: Assume it's true for n < k, we'll it for n = k.