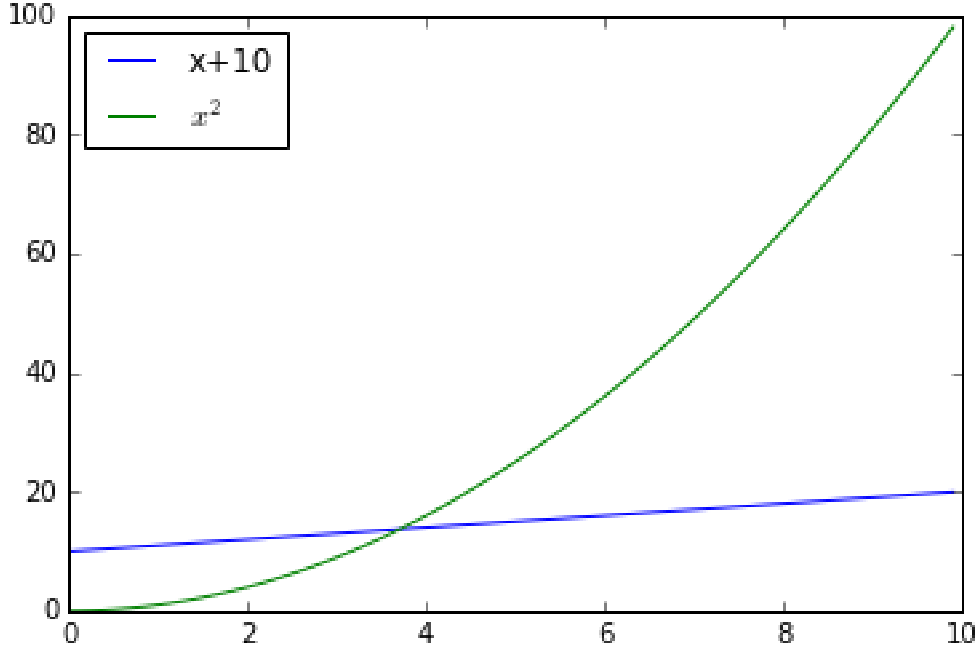
CS 4800: Algorithms & Data

Lecture 2

January 13, 2017

Asymptotic notations

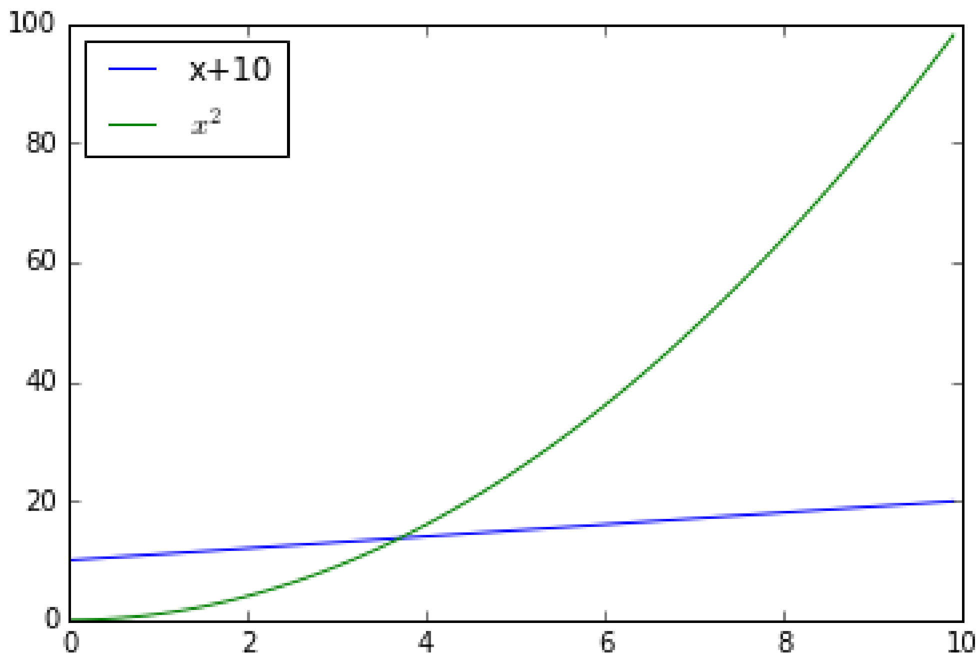
- Big-Oh: $f(n) = O(g(n))$ if there are constants c > 0 and n_0 such that $f(n) \leq c \cdot g(n)$ for all $n > n_0$
- Analog of $f(n) \leq g(n)$



$n + 10 = O(n^2)$

Asymptotic notations

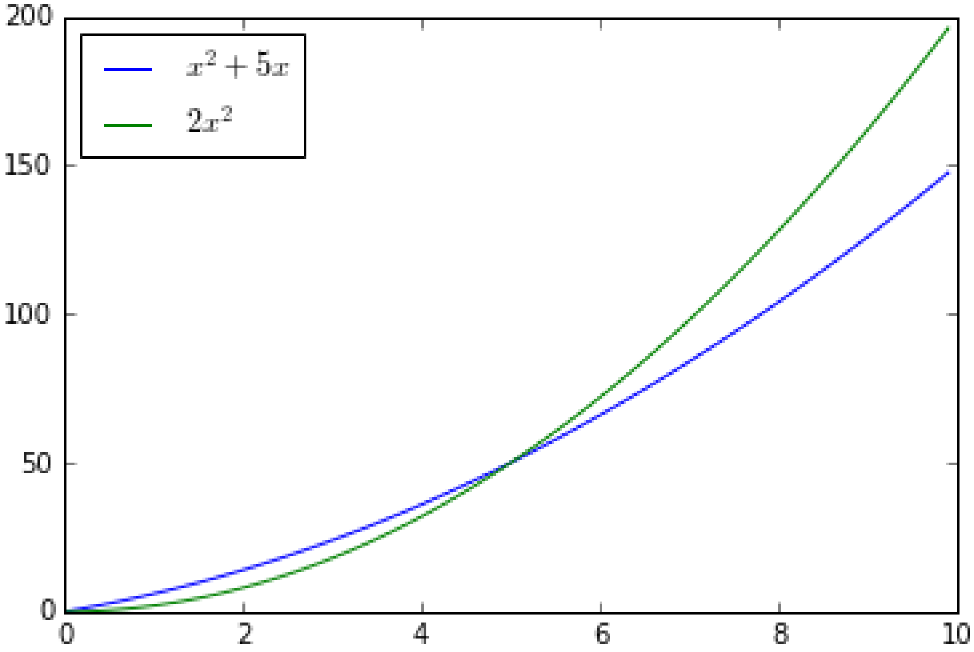
- Big-Ω (Omega): $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$
- Analog of $f(n) \geq g(n)$



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

Asymptotic notations

- Big-Θ (Theta): $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $g(n) = O(f(n))$
- Analog of $f(n) = g(n)$



$n^2 + 5n < 3 \cdot 2n^2$

$n^2 + 5n > 0.5 \cdot 2n^2$

$n^2 + 5n = \Theta(2n^2)$

Rules of thumb

- Multiplicative constants can be ignored: $14n^2 = \Theta(n^2)$
- n^a dominates n^b if a>b: $n^4 = \Omega(n^3)$
- If log(f(n)) > log(g(n)) for all large n then f = Ω(g)
- Any exponential dominates any polynomial: $2^n = \Omega(n^3)$
- Any polynomial dominates any logarithm: $n^{0.1} = \Omega((\log_2 n)^3)$

Example

- Rule: If log(f) > log(g) for all large n then f = Ω(g)
- A lot of times, can compare by taking logs:
- Compare n^4 and n^3
- $\log(n^4)$
  - $= 4\log_2 n$
- $\log(n^3)$
  - $= 3\log_2 n$
- Thus, $n^4 = \Omega(n^3)$

Example

- Rule: If log(f) > log(g) for all large n then f = Ω(g)
- Compare $4^{\log_2 n}$ and $8^{\log_3 n}$
- $\log\left(4^{\log_2 n}\right)$
  - $= \log(4) \log_2 n$
  - $= \frac{\log(4) \log(n)}{\log(2)}$
- $\log\left(8^{\log_3 n}\right)$
  - $= \log(8) \log_3 n$
  - $= \frac{\log(8) \log(n)}{\log(3)}$

Homework discussion

- Rank the following functions in order of decreasing order of growth
- n^3
- 1.01^n
- $\sqrt{n}$
- $7^{\log_4 n}$

- Multiplicative constants can be ignored: $14n^2 = \Theta(n^2)$
- n^a dominates n^b if a>b: $n^4 = \Omega(n^3)$
- Any exponential dominates any polynomial: $2^n = \Omega(n^3)$
- Any polynomial dominates any logarithm: $n^{0.1} = \Omega((\log_2 n)^3)$
- If log(f) > log(g) then f = Ω(g)

Arithmetic algorithms

Addition

	1	4	2	8
+		5	7	9
	2	0	0	7

O(1) operations per digit so O(n) running time

Multiplication

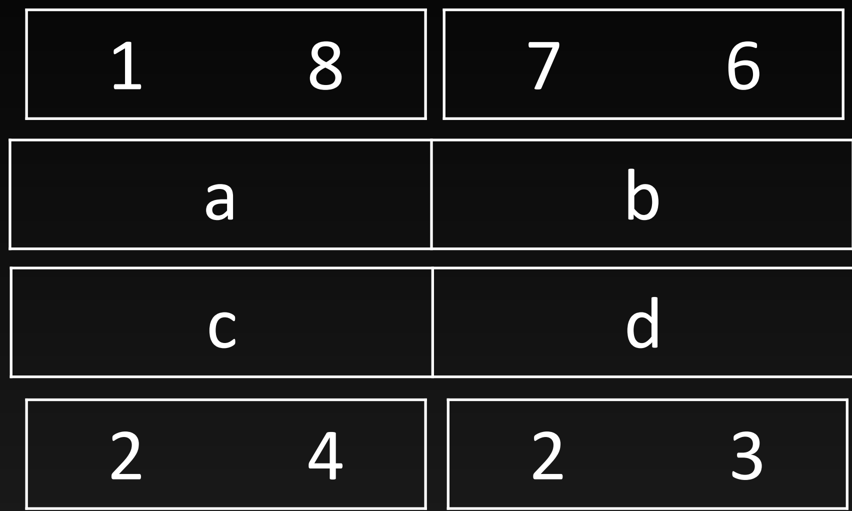
			1	8	7	6	n digits
		x	2	4	2	3	
			5	6	2	8	n operations
		3	7	5	2		
							n rows
	7	5	0	4			
3	7	5	2				
4	5	4	5	5	4	8	

$O(n^2)$ time

Divide & Conquer

- Break a problem into smaller subproblems of the same type
- Recursively solve subproblems
- Combine their answers

New multiplication algorithm

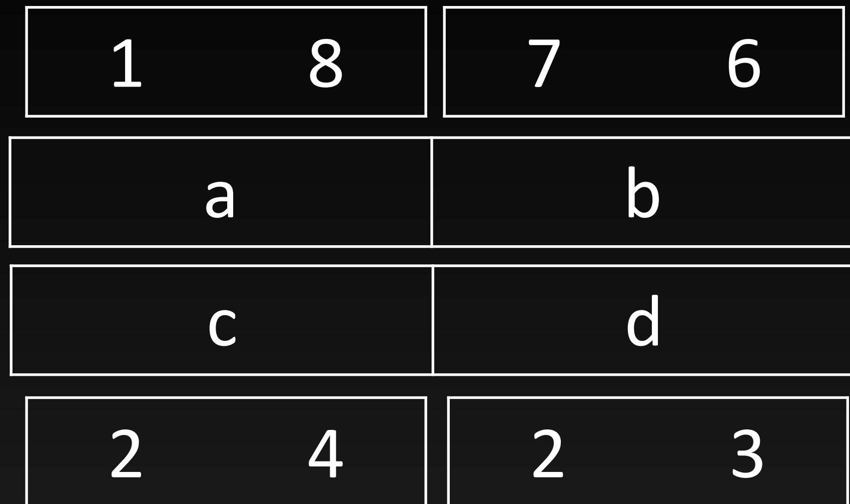


$$X = 10^2 \cdot 18 + 76$$
$$Y = 10^2 \cdot 24 + 23$$

- Initial problem: compute $X \cdot Y$
- Four subproblems: $18 \cdot 24$, $76 \cdot 24$, $18 \cdot 23$, $76 \cdot 23$
- Recursively solve subproblems
- Combine their answers (add 0s, additions)

$$XY = (10^2 \cdot 18 + 76)(10^2 \cdot 24 + 23)$$
$$= 10^4 \cdot 18 \cdot 24 + 10^2(76 \cdot 24 + 18 \cdot 23) + 76 \cdot 23$$

New multiplication algorithm



$$X = 10^{n/2} \cdot a + b$$

$$Y = 10^{n/2} \cdot c + d$$

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

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

- 4 multiplications of n/2-digit numbers
- 3 additions of n digit numbers

Running time

- $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + Cn$
- Prove by induction that $T(n) \leq (C+1)n^2 - Cn$ (assume T(1)=1, just one multiplication)
- For n=1, T(n)=1
- Assume claim is true for n<k for integer k>0. Will prove the claim for n=k.
- $T(k) = 4 \cdot T\left(\frac{k}{2}\right) + Ck$

$$\leq 4\left((C+1)\left(\frac{k}{2}\right)^2 - \frac{Ck}{2}\right) + Ck$$

- Do algebra, get $T(k) \leq (C+1)k^2 - Ck$

Carl Friedrich Gauss (1777-1855)

- Complex numbers $a + b \cdot i$
- $i^2 = -1$
- Product of complex numbers:
  $$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$
- Can be computed with 3 multiplications
- $(a - b)(c - d) = ac + bd - ad - bc$
- $ad + bc = ac + bd - (a - b)(c - d)$

Karatsuba's algorithm

Karatsuba(X, Y, n)

- If n = 1 then return $X \cdot Y$
- Else:
  - $m = \lceil n/2 \rceil$
  - Rewrite $X = 10^{\lceil n/2 \rceil} a + b, Y = 10^{\lceil n/2 \rceil} c + d$
  - $e = Karatsuba(a, c, m)$
  - $f = Karatsuba(b, d, m)$
  - $g = Karatsuba(a - b, c - d, m)$
  - Return $10^{2m} e + 10^{m} (e + f - g) + f$

Homework discussion

- Carry out Karatsuba's algorithm for 12 * 98

Karatsuba(X, Y, n)

- If n = 1 then return $X \cdot Y$
- Else:
  - $m = \lceil n/2 \rceil$
  - Rewrite $X = 10^{\lceil n/2 \rceil}a + b, Y = 10^{\lceil n/2 \rceil}c + d$
  - $e = Karatsuba(a, c, m)$
  - $f = Karatsuba(b, d, m)$
  - $g = Karatsuba(a - b, c - d, m)$
  - Return $10^{2m}e + 10^{m}(e + f - g) + f$

Running time analysis

- Recursive calls: $3 \cdot T(n/2)$
- Additional work (additions and multiplications with powers of 10) : Cn for some constant C

Karatsuba(X, Y, n)

- If n = 1 then return $X \cdot Y$
- Else:
  - $m = \lceil n/2 \rceil$
  - Rewrite $X = 10^{\lceil n/2 \rceil} a + b, Y = 10^{\lceil n/2 \rceil} c + d$
  - $e = Karatsuba(a, c, m)$
  - $f = Karatsuba(b, d, m)$
  - $g = Karatsuba(a - b, c - d, m)$
  - Return $10^{2m} e + 10^{m}(e + f - g) + f$

Running time analysis

- Recursive calls: $3 \cdot T(n/2)$
- Additional work (additions and multiplications with powers of 10): $\mathrm{Cn} = \ \mathrm{O}(n)$ for constant C>1
- Recurrence:

$$T(n) = 3 \cdot T\left(\frac{n}{2}\right) + Cn$$

- Solution:

$$T(n) = O(n^{\log_2 3})$$

Induction

- Guess an induction hypothesis

$$T(n) \le 3Cn^{\log_2 3} - 2Cn$$

- Base case: n=1, $T(1) \le C$
- Inductive step, assume true for all n<k, will prove for n=k
  - $T(k) = 3 \cdot T\left(\frac{k}{2}\right) + Ck$
  - $\qquad \le 3 \cdot \left(3C\left(\frac{k}{2}\right)^{\log_2 3} - \frac{2Ck}{2}\right) + Ck$
  - $\qquad = 9C \cdot \frac{k^{\log_2 3}}{2^{\log_2 3}} - 3Ck + Ck$
  - $\qquad = 9C \cdot k^{\log_2 3}/3 - 2Ck$
  - $\qquad = 3Ck^{\log_2 3} - 2Ck$