Prof. Wheeler Ruml
TA Steve Wissow

`http://www.cs.unh.edu/~ruml/cs758`

4 handouts:
course info, schedule, slides, asst 1

online handouts: programming, functions

1 physical sign-up laptop (for grades, piazza)

Algorithms

Algorithms Today

- web: search, caching, crypto
- networking: routing, synchronization, failover
- machine learning: data mining, recommendation, prediction
- bioinformatics: alignment, matching, clustering
- hardware: design, simulation, verification
- business: allocation, planning, scheduling
- finance: trading, credit scoring, fraud detection
- government: bail, sentencing, social scoring
- AI: robotics, games, language, vision

Definition

Algorithm

- input into output
- precisely defined
- mechanical steps
- terminates

What might we want to know about it?

Why?

- Computer scientist $\neq$ programmer
  - understand program behavior
  - have confidence in results, performance
  - know when optimality is abandoned
  - solve ‘impossible’ problems
  - sets you apart (eg, interviewing)
- CPUs aren’t getting faster
- Devices are getting smaller
- Software is the differentiator

  ‘Software is eating the world’ — Marc Andreessen, 2011
- Everything is computation

The Word: Abū ‘Abdallāh Muḥammad ibn Mūsā al-Khwārizmī

780-850 AD
Born in Uzbekistan, worked in Baghdad.
Solution of linear and quadratic equations.
Founder of algebra.
Popularized arabic numerals, decimal positional numbers
→ algorism (manipulating digits)
→ algorithm.

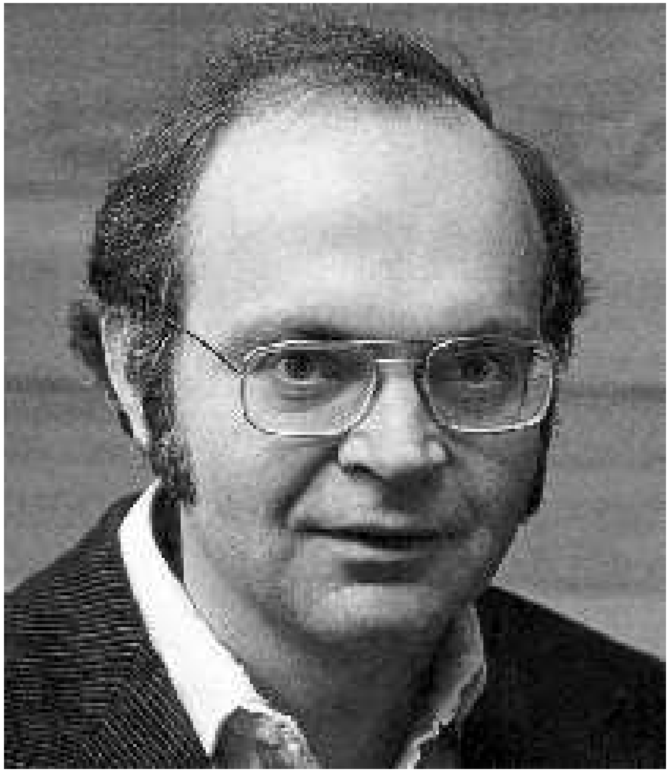
The Compendious Book on Calculation by Completion and Balancing, 830.

The Founder: Donald E. Knuth

invented algorithm analysis, O
The Art of Computer Programming, vol. 1, 1968

developed TEX,
literate programming

famous results, students
published in MAD magazine



This Class

Relations

- requires 531/659 (proofs), 515 (data structures), 420 (C)
  - some intentional overlap!
  - beware imposter syndrome
  - problems intentionally unpredigested
- central (required for BS CS and BS DS)
- same content both semesters
- continuous improvement!

Topics

'Greatest Hits'

1. data structures: trees, tries, hashing
2. algorithms: divide-and-conquer, dynamic programming, greedy, graphs
3. correctness: invariants
4. complexity: time and space
5. NP-completeness: reductions

Not including

1. (much) computability
2. (many) randomized algorithms
3. parallel algorithms
4. distributed algorithms
5. numerical algorithms, eg: crypto, linear algebra
6. geometric algorithms
7. on-line or 'run forever' algorithms
8. fancy analysis

Course Mechanics

- names $\rightarrow$ faces
- sign up sheet
- General information
  - contact, books, C, due dates, collaboration, `piazza.com`
- Schedule
  - wildcard slot
- Expectations
  - 50/4=12.5; 50/3=16.7
  - 2018: median 12, mean 12.8, stddev 5.5
  - 2023: median 16, mean 16.1, stddev 5.2
- Feedback is always needed and appreciated.
  - eg, EOLQs. Try coming to my office hours!

Sorting

Sorting

- Bubble Sort
- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort

How to sort one million records?

Sorting

- Bubble Sort
- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort

How to sort one million records?

How to sort one billion 16-bit integers?

- Bubble Sort
- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort

How to sort one million records?

How to sort one billion 16-bit integers?

How to sort one trillion 4-bit integers?

Counting Sort

For n numbers in the range 0 to k:

1. for i from 0 to k
2.   count$[i] \leftarrow 0$
3. for each input number x
4.   increment count$[x]$
5. for i from 0 to k
6.   do count$[i]$ times
7.   emit i

Counting Sort

For n numbers in the range 0 to k:

1. for i from 0 to k
2. count$[i] \leftarrow 0$
3. for each input number x
4. increment count$[x]$
5. for i from 0 to k
6. do count$[i]$ times
7. emit i

Correctness?

Complexity?

Correctness

property 1: output is in sorted order

proof sketch: output loop increments i, never decrements

Correctness

property 1: output is in sorted order
proof sketch: output loop increments i, never decrements

property 2: output contains same numbers as input
invariant:

Correctness

property 1: output is in sorted order
proof sketch: output loop increments i, never decrements

property 2: output contains same numbers as input
invariant: for each value,
remaining input + sum of counts = total
proof sketch:

Correctness

property 1: output is in sorted order
proof sketch: output loop increments i, never decrements

property 2: output contains same numbers as input
invariant: for each value,
remaining input $+$ sum of counts $=$ total
proof sketch:
initialized/established: before line 3
maintained: through lines 3–4
at termination: no remaining input by line 5
each number printed count times
therefore, output has same numbers as input

Complexity

For n numbers in the range 0 to k:

1. for i from 0 to k
2.   count$[i] \leftarrow 0$
3. for each input number x
4.   increment count$[x]$
5. for i from 0 to k
6.   do count$[i]$ times
7.   emit i

Correctness? Yes.

Complexity?

RAM model: no cache

order of growth

worst-case

[try with previous slide]

Counting Sort

For n numbers in the range 0 to k:

for x from 0 to k	$O(k)$
count$[x] \leftarrow 0$	
for each input number x	$O(n)$
increment count$[x]$	
for x from 0 to k	$O(k)$ times around loop
do count$[x]$ times	iterates $O(n)$ times total
emit x	$O(1)$ each time

$$O(k + n + k + n) = O(2n + 2k) = O(n + k) \neq O(n \lg n)$$

Order Notation

ignore constant factors

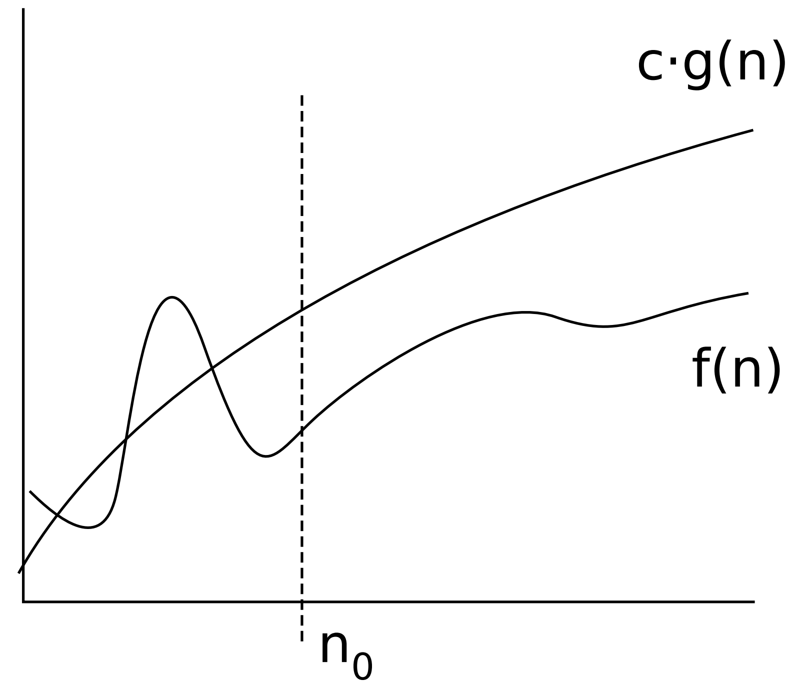
ignore 'start-up costs'

upper bound

Order Notation

ignore constant factors
ignore 'start-up costs'
upper bound

f(n) = O(g(n))



eg, running time is $O(n \log n)$

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c, n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_o\}$$

We can upper-bound (the tail of) f by scaling up g.

Note non-transitive use of $=$. Pronounced 'is'.

Eg:

1. $0.002x^2 - 35,456x + 2^{80}$
2. $O(n^2)$ vs $O(n^3)$
3. $O(2^n)$ vs $O(3^n)$
4. $O(2^n)$ vs $O(2^{n+2})$ vs $O(2^{2n})$ vs $O(n^n)$

"What is n?"

Examples

is $n^3 = O(n^2)$

$0.2x^2 - 456x + 2^{20}$

$10n^2 + 5n$

$O(n^2)$ vs $O(n^3)$

And Friends

Upper bound ('order of'):

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c, n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_o\}$$

Lower bound:

$$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c, n_0 \text{ such that } cg(n) \leq f(n) \text{ for all } n \geq n_o\}$$

Tight bound:

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2, n_0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_o\}$$

Asymptotics

Upper bound ('dominated by'):

$$o(g(n)) = \{f(n) : \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\}$$

Lower bound ('dominates'):

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

EOLQs

- What's still confusing?
- What question didn't you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.
Thanks!