CMSC 351
Introduction to Algorithms

Spring 2020

Administration

- General Administration:
  Andrew Witten <awitten1@terpmail.umd.edu>

- Exam Scheduling:
  Jamie Matthews <jamiem@cs.umd.edu>

Administration (continued)

- Webpage
  - Get homework assignments
  - Syllabus
  - Other documents
- Piazza
  - Ask questions
    - Do **not** post solutions.
    - Do **not** ask if your answer or approach is correct.
  - Discuss issues
  - Public versus Private
- ELMS
  - Get homework solutions
  - See final grades
- Gradescope
  - Hand in homework
  - See graded homeworks and exams

Administration (continued)

- Textbook (bookstore/on reserve at McKeldin Library)
  - Cormen, Leiserson, Rivest, and Stein, *Introduction to Algorithms* (3rd ed., 2009). MIT Press. (Any edition is fine.)
- Class attendance
  - You are responsible for what is said in class.
  - Laptops and other devices: Do not share during class.
  - Lectures will be posted (mostly).
- Office hours
- Exams
  - Two evening midterms: **6:00-8:00pm**.
    - Wednesday, March 11
    - Tuesday, April 21
  - Final exam: **4:00-6:00pm**.
    - Saturday, May 16th

Administration (continued)

- Homework
  - Three types
    - Regular: typically due each Friday. (1% each)
    - NP-completeness: typically due every other Wednesday. (1/2% each)
    - Programming project (maybe). (4%)
  - Must be in PDF.
  - Must be easy to read (your responsibility).
  - Late date: 25% off your actual grade.
    - One get-out-of-jail-free card for each type.
  - Your neighbor should understand your answers.
  - Study groups. (Teach it.)
  - Must write up homework solutions yourself.
    - State who is in your study group at top of homework.
    - State what outside resources you used to solve each problem.
  - Do problems from book (and other books).
- Grading

Topics (tentative)

- Introduction, Ch. 1,2
- Quadratic sorting algorithms
- Mergesort, Ch. 2
- Summations, Appendix A
- Recurrences, Ch. 4
- Heapsort, Ch. 6
- Quicksort, Ch. 7
- Sorting in Linear Time, Ch. 8
- Medians and Order Statistics, Ch. 9
- Graphs and Trees, Appendix B
- Minimum Spanning Trees, Ch. 23
- Shortest Paths: Dijkstra's algorithm, Ch. 24.3
- Introduction to NP-completeness, Ch. 34

Why learn this material?

Why learn this material?

- Algorithms are everywhere in Computer Science (and elsewhere).

Why learn this material?

- Algorithms are everywhere in Computer Science (and elsewhere).
- Useful for later courses.

Why learn this material?

- Algorithms are everywhere in Computer Science (and elsewhere).
- Useful for later courses.
- Useful for computer programming.

  “Micro-algorithms”

Why learn this material?

- Algorithms are everywhere in Computer Science (and elsewhere).
- Useful for later courses.
- Useful for computer programming.

  "Micro-algorithms"
- Useful to get a job.

Why learn this material?

- Algorithms are everywhere in Computer Science (and elsewhere).
- Useful for later courses.
- Useful for computer programming.

  “Micro-algorithms”
- Useful to get a job.
- Useful on the job.

What is an algorithm?

Definition

An *algorithm* is a finite list of step-by-step instructions for solving a problem.

Efficiency

- Time
- Space

Example

Tournament assignment. (Think about at home.)

Good Algorithms Are Critical

Example

Two sorting algorithms:

- Slow algorithm (bubble sort): $4n^2$ instructions
- Fast algorithm (merge sort): $80n \lg n$ instructions

Two computers:

- Fast computer: 10 billion instructions/second
- Slow computer: 10 million instructions/second

Time to sort 10 million numbers:

Good Algorithms Are Critical

Example

Two sorting algorithms:

- Slow algorithm (bubble sort): $4n^2$ instructions
- Fast algorithm (merge sort): $80n \lg n$ instructions

Two computers:

- Fast computer: 10 billion instructions/second
- Slow computer: 10 million instructions/second

Time to sort 10 million numbers:

- Slow algorithm, fast computer: $\approx$ 11 hours

Good Algorithms Are Critical

Example

Two sorting algorithms:

- Slow algorithm (bubble sort): $4n^2$ instructions
- Fast algorithm (merge sort): $80n \lg n$ instructions

Two computers:

- Fast computer: 10 billion instructions/second
- Slow computer: 10 million instructions/second

Time to sort 10 million numbers:

- Slow algorithm, fast computer: ≈ 11 hours
- Fast algorithm, slow computer: $\approx \frac{1}{2}$ hour

Calculate Time

Example

- Slow algorithm, fast computer:

$$\frac{4 \cdot \left(10^7\right)^2 \text{ instructions}}{10^{10} \text{ instructions/second}} = 40000 \text{ secs} \approx 11 \text{ hrs}$$

- Fast algorithm, slow computer:

$$\frac{80 \cdot 10^7 \lg\left(10^7\right) \text{ instructions}}{10^7 \text{ instructions/second}} \approx 1860 \text{ secs} \approx 31 \text{ mins}$$

Calculate Time

Example

Other way around:

Calculate Time

Example

Other way around:

- Slow algorithm, slow computer: $\approx 1\frac{1}{4}$ years
- Fast algorithm, fast computer: ≈ 2 seconds

Calculate Time

Example

Other way around:

- Slow algorithm, slow computer: $\approx 1\frac{1}{4}$ years
- Fast algorithm, fast computer: ≈ 2 seconds

Both using fast computer:

Calculate Time

Example

Other way around:

- Slow algorithm, slow computer: $\approx 1\frac{1}{4}$ years
- Fast algorithm, fast computer: ≈ 2 seconds

Both using fast computer:

- Slow algorithm, fast computer: ≈ 11 hours
- Fast algorithm, fast computer: ≈ 2 seconds