ECS 122A
Algorithm Design and Analysis

Instructor: Qirun Zhang

Course slides (partially) adopted from the notes by David Luebke.

Agenda

- Logistics
- Proof by induction
- Gentle introduction to analysis of algorithm
  - Insertion sort

The Course

- Purpose: a rigorous introduction to the design and analysis of algorithms
  - Not a lab or programming course
  - Not a math course, either
- Textbook: *Introduction to Algorithms*, Cormen, Leiserson, Rivest, Stein
  - The "Big White Book"
  - Second edition: now "Smaller Green Book"
  - An excellent reference you should own
- Course website
  - http://helloqirun.github.io/course/122A/index.html

The Course

- Instructor: Qirun Zhang
  - *qrzhang@ucdavis.edu*
  - Office: RM. 1031 ASB (Academic Surge Building)
  - Office hours: 2pm-4pm Wednesday
- TA: Thong Le
  - *thmle@ucdavis.edu*
  - Office: RM. 55 Kemper
  - Office hours: 2pm-4pm Monday
- No discussion TODAY!

The Course

- Format
  - Three lectures/week
  - One homework assignment/week
    - About 5 problems
    - No homework for the exam weeks
    - 4 homework assignments in total
    - Dates:
      - Release: Friday of week k
      - Due: Thursday of week (k+1)
  - Two exams
    - Midterm: 8/25
    - Final: 9/15

Submitting your homework

- We use the website gradescope
  - https://gradescope.com/
  - We will *not* mark your homework in the paper form or submitted via email.
  - Sign up using your *UCD email* and *Student ID* (very important!)
    - You are responsible for providing the correct UCD email and SID at sign-up, as incorrect information may affect your homework score.

The Course

- Grading policy:
  - Homework: 40%
  - Midterm: 30%
  - Final: 30%
- Academic integrity:
  - http://sja.ucdavis.edu/academic-integrity.html
- One suggestion for this course

Tentative Schedule

Date	**Topics**	**Reading**
8/7	Introduction	
8/9	Asymptotic notation	CLRS 3.1-3.2
8/11	Divide-and-conquer: recurrences (mergesort)	CLRS 4.1-4.3
8/14	Divide-and-conquer: master theorem	CLRS 4.5
8/16	Sorting: heapsort and priority queues	CLRS 6
8/18	Sorting: quicksort and its analysis	CLRS 7
8/21	Graph algorithms: basics	CLRS 22.1-22.3
8/23	Graph algorithms: minimum spanning trees	CLRS 23
8/25	Mid-term	
8/28	Shortest paths: Bellman-Ford	CLRS 24.1
8/30	Shortest paths: Dijkstra	CLRS 24.2-24.3
9/1	Disjoint sets	CLRS 17
9/4	**Holiday**	
9/6	Dynamic programming	CLRS 15.1 15.3
9/8	Dynamic programming: longest comment subsequence	CLRS 15.4
9/11	NP-completeness	CLRS 34.1 34.2
9/13	NP-completeness: reductions	CLRS 34.3
9/15	Np-completeness and review for final	CLRS 34.4

Review: Induction

- Suppose
  - S(k) is true for fixed constant k
    - Often k = 0
  - S(n) → S(n+1) for all n >= k
- Then S(n) is true for all n >= k

Proof By Induction

- Claim:S(n) is true for all n >= k
- Base step:
  - Show formula is true when n = k
- Inductive hypothesis:
  - Assume formula is true for an arbitrary n
- Inductive step:
  - Show that formula is then true for n+1

Induction Example: Gaussian Closed Form

Induction Example:
Geometric Closed Form

Analysis of Algorithms

- Analysis is performed with respect to a computational model
- We will usually use a generic uniprocessor random-access machine (RAM)
  - All memory equally expensive to access
  - No concurrent operations
  - All reasonable instructions take unit time
    - Except, of course, function calls
  - Constant word size
    - Unless we are explicitly manipulating bits

An Example: Insertion Sort

- The sorting problem
  - Input: a sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$
  - Output: a permutation (reordering) $\langle a'_1, a'_2, \dots, a'_n \rangle$ of the input sequence such that $a'_1 \le a'_2 \le \cdots \le a'_n \rangle$
- Insertion sort
  - Playing card

Insertion Sort

Input Size

- Time and space complexity
  - This is generally a function of the input size
    - E.g., sorting, multiplication
  - How we characterize input size depends:
    - Sorting: number of input items
    - Multiplication: total number of bits
    - Graph algorithms: number of nodes & edges
    - Etc

Running Time

- Number of primitive steps that are executed
  - Except for time of executing a function call most statements roughly require the same amount of time
    - y = m * x + b
    - c = 5 / 9 * (t - 32)
    - z = f(x) + g(y)
- We can be more exact if need be

Analysis

- Worst case
  - Provides an upper bound on running time
  - An absolute guarantee
- Average case
  - Provides the expected running time
  - Very useful, but treat with care: what is “average”?
    - Random (equally likely) inputs
    - Real-life inputs

Analyzing Insertion Sort

- $T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 T + c_5(T - (n-1)) + c_6(T - (n-1)) + c_7(n-1)$

  $= c_8 T + c_9 n + c_{10}$
- What can T be?
  - Best case -- inner loop body never executed
    - $t_i = 1$ ➔ T(n) is a linear function
  - Worst case -- inner loop body executed for all previous elements
    - $t_i = i$ ➔ T(n) is a quadratic function

The End