CS 375 – Analysis of Algorithms

Professor Eric Aaron

Lecture – M W 1:00pm

Lecture Meeting Location: It’s complicated…

Instructor Info

• Professor Eric Aaron

  Website: https://cs.colby.edu/eaaron
  Office: Davis 113
  Office Hours: M 2:30-4pm, T 4-5pm,
  W 2:30-4pm and 6:30-7:30pm,
  and by email appointment
  Phone/Voicemail: (207) 859-5857
  E-mail: eaaron@colby.edu

  The above email address is the best way to contact me

  Course Website: https://cs.colby.edu/courses/S21/cs375

My website is the most important info on this slide—
the rest of it can be found there.
Colby College prohibits and will not tolerate sexual misconduct or gender-based discrimination of any kind. Colby is legally obligated to investigate sexual misconduct (including, but not limited to, sexual assault and sexual harassment) and other specific forms of behavior that violate federal and state laws (Title IX and Title VII, and the Maine Human Rights Act). Such behavior also requires the College to fulfill certain obligations under two other federal laws, the Violence Against Women Act (VAWA) and the Jeanne Clery Disclosure of Campus Security Policy and Campus Statistics Act (Clery Act).

To learn more about what constitutes sexual misconduct or to report an incident, see: www.colby.edu/studentlife/handbook-section/f-sexualmisconduct/

If you wish to speak confidentially about an incident of sexual misconduct, you may contact:

• Counseling Center: 207-859-4490
• Gender and Sexual Diversity Program: Director Emily Schusterbauer (eeschust@colby.edu/ 207-859-4093)
• Office of Religious & Spiritual Life: 207-859-4272
  – Dean of Religious & Spiritual Life, Kurt Nelson (kdnelson@colby.edu)
  – Jewish Chaplain, Erica Asch (elsasch@colby.edu)
  – Catholic Campus Minister, Charles Demm (cademm@colby.edu)

Students should be aware that faculty members are considered "responsible employees"; as such, if you disclose an incident of sexual misconduct to a faculty member, they have an obligation to report it to Colby's Title IX Coordinator. "Disclosure" may include communication in-person, via email/phone/text, or through class assignments.

I take Title IX seriously. Please feel free to talk with me for more information!
A tiny bit about the course

- Course textbook:
  - *Introduction to Algorithms, 3rd edition*
    by Cormen, Leiserson, Rivest, and Stein
  - Online version is available FREE through the Colby library!

- Important elements for any course on algorithms:
  - Classic algorithms (which you might use or adapt for your work)
  - Algorithm design techniques and paradigms
    - Creating and working with algorithm specifications
  - Analyzing and explaining an algorithm’s correctness
  - Analyzing and explaining an algorithm’s complexity

Underlying all of these is the idea that there’s an essential relationship between an algorithm and a program!

What’s the difference between an algorithm and a program?

A tiny bit more about the course: Components of graded work in CS375

- Problem sets
  - Due at least one week after being assigned

- Concept-Heavy Intensive Problem Sets
  - More focused on concepts/analysis than on standard problem sets
  - There will be opportunity for revisions after getting feedback from me—at least one meeting with me will be required
  - More details about the process to come, later in the semester

- Smaller assignments
  - Given occasionally; typically due beginning of next class meeting
  - May be discussed in next class meeting

- Class participation
  - Anything that constructively contributes to class discussion: speaking in class, making points in office hours, making points by email, etc.
Problems and Solutions

- On a fundamental level, CS375 (and all of Computer Science!) is all about problems—and problem solving

  - What makes a solution to a problem good (or bad)?
    - What makes one solution to a problem better than another?
Problems and Solutions

• On a fundamental level, CS375 (and all of Computer Science!) is all about problems—and problem solving.

• Throughout the course (and throughout all of Computer Science!), definitions are very important, so let’s start here!
  – What is a problem, in a useful, computational sense?
  – What does it mean to solve a problem?

(Note: This isn’t the question of “How does one solve a problem?” This is about what it means to have a solution, no matter how that solution is found.)

  – What makes a solution to a problem good (or bad)?
    • What makes one solution to a problem better than another?


• What is a problem, in a useful, computational sense?
  – Informal definition: In a relevant sense, a problem is an input/output relationship

From our textbook: “The statement of a problem specifies, in general terms, the desired input / output relationship.”

• What is a problem, in a useful, computational sense?
  – Informal definition: In a relevant sense, a problem is an input/output relationship

From our textbook: “The statement of a problem specifies, in general terms, the desired input / output relationship”

– For example, you may have heard of the sorting problem

| Input: A sequence L of n numbers (a₀, ..., aₙ₋₁) |
| Output: A sequence L' of n numbers (b₀, ..., bₙ₋₁) that re-orders the input sequence (perhaps leaving them unchanged) such that b₀ ≤ b₁ ≤ ... ≤ bₙ₋₁ |

It’s often a good idea to have a specification like the one in yellow above as part of the comments for functions you write in your programs—it makes understanding the code and ensuring its correctness much, much easier!


• What is a problem, in a useful, computational sense?
  – Informal definition: In a relevant sense, a problem is an input/output relationship

From our textbook: “The statement of a problem specifies, in general terms, the desired input / output relationship”

• What does it mean to solve a problem?

Now that we have a definition of problem, we can come up with a cleaner answer to this!

• What makes a solution to a problem good (or bad)?
  – What makes one solution to a problem better than another?

• What is a *problem*, in a useful, computational sense?
  – Informal definition: In a relevant sense, a problem is an input/output relationship
• What’s an algorithm? Informal definition, from CLRS:

  An algorithm is a well-defined computational procedure that takes input and produces output.

• What does it mean to solve a problem?
  – Informal definition: In this computational sense, a solution to a problem is an algorithm…
  – We say an algorithm *correctly solves a problem* when it transforms every input to its related, correct output

What makes a solution to a problem good (or bad)?
• For example, consider the sorting problem (below). It has many solutions! What makes one solution better than another?

**Input:** A sequence L of n numbers \((a_0, \ldots, a_{n-1})\)

**Output:** A sequence L' of n numbers \((b_0, \ldots, b_{n-1})\) that re-orders the input sequence (perhaps leaving them unchanged) such that \(b_0 \leq b_1 \leq \cdots \leq b_{n-1}\)

• What makes a solution to a problem good (or bad)?
• (For instance, consider the sorting problem.) What makes one solution to a problem better than another?
  – Correctness (i.e., does it work?)
  – Time complexity (i.e., how fast is it?)
  – Space complexity (i.e., how much memory does it use?)
  – (Other?)

Input: A sequence L of n numbers \((a_0, \ldots, a_{n-1})\)
Output: A sequence L’ of n numbers \((b_0, \ldots, b_{n-1})\) that re-orders the input sequence (perhaps leaving them unchanged) such that \(b_0 \leq b_1 \leq \ldots \leq b_{n-1}\)


• Just as a problem can have many algorithms that solve it (e.g., sorting, searching problems)…
  (There are multiple sort and search algos. Which ones do you know?)
• … An algorithm can have many possible implementations in code
  – For example, every programming language would lead to a different implementation
• For CS375, we’ll consider problems and algos more than code
  – You already know enough to implement algos in at least one language!

Determining the best solutions for a problem is often better at the algorithm level than the code level—all implementations of the same algorithm will have the same time / space complexity!
A tiny bit about the course, the Remix: Introduction to some Main Ideas

- Recall important elements for any course on algorithms:
  - Classic algorithms (which you might use or adapt for your work)
  - Algorithm design techniques and paradigms
    - Creating and working with algorithm specifications
  - Analyzing and explaining an algorithm’s correctness
  - Analyzing and explaining an algorithm’s complexity

- We’ll spend a lot of the semester on these important elements

<table>
<thead>
<tr>
<th>Design Paradigm</th>
<th>Analysis</th>
<th>Complexity (Efficiency)</th>
<th>Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative</td>
<td>Counting</td>
<td>(Exact count of operations / space used)</td>
<td>Loop invariants</td>
</tr>
<tr>
<td>Recursive</td>
<td>Solving recurrences</td>
<td></td>
<td>Induction</td>
</tr>
</tbody>
</table>

To me, these are the most important elements

Business

- **IMPORTANT**: Please email me from the account at which you’d want me to contact you
  - Include a sentence on what you’d like to get out of the course
  - … plus anything else you might like to tell me!
  - Also, please tell me whether or not you’re on campus this semester, and if not, what your time zone is
  - Also, in your email, let me know if you were able to access the course website and lecture notes without any difficulties
    - Recall: website is at [https://cs.colby.edu/courses/S21/cs375](https://cs.colby.edu/courses/S21/cs375)

(i’ll post lecture notes by the end of the day—i.e., midnight—today)
Business, pt. 2

- The first Problem Set will be posted on course website by the end of the day today
  - Due at the beginning of class on Wednesday, Feb. 17
  - Please be sure to do the reading on it! If you have questions about the math pre-requisites in the Appendices, please let me know—otherwise, I will assume you are comfortable with them
    - (We will also review them in contexts where they come up in class—e.g., we’ll briefly review graphs when we discuss graph algorithms)