CPSC 327
Data Structures and Algorithms

Data Structures and Algorithms

- a **data structure** deals with the organization of data for efficient access and modification
- an **algorithm** is a procedure to accomplish a specific task

Ideally an algorithm is
correct,
efficient, and
easy to implement.

(But you can't always get all three.)

Challenge

An array contains each of the numbers 1 – 1,000,000 plus one duplicate value. Which value is duplicated?

- fairly easy to come up with algorithms
- correctness is fairly straightforward
- main question is efficiency

Challenge

An actor has received *n* offers of movies to star in. The first and last days of filming for each movie is known. Which offers should the actor accept? She wants to be in as many movies as possible but cannot work on two movies whose filming times overlap.

- not too hard to come up with possible algorithms
- correctness in terms of non-overlapping movies is easy to show
- optimality (“as many movies as possible”) is not necessarily straightforward and harder to show – reasonable-looking algorithms can easily be incorrect

Challenge

A robot needs to solder *n* points on a circuit board. The time it takes for the arm to move from one point to another is proportional to the distance between the points. Find the order in which the points should be soldered so as to get the job done as fast as possible. (The arm must return to its initial position when the job is done.)

- trivial to find an easily-proven-to-be-optimal algorithm but impossible* to find an efficient optimal algorithm
- typically must settle for heuristics or approximation algorithms

* under current models of computation, as far as we know

Key Questions

How do we come up with a data structure or an algorithm?

Is the algorithm correct?

How much time/space does it take? Can we do better?

Are there some problems that fundamentally require more time/space to solve than others? Are any unsolvable?

Course Content

- essential tools
  - analysis of algorithms
  - establishing correctness
- data structures toolbox (with a bit of algorithms)
  - collections
  - searching and lookup
  - sorting
  - fancier data structures
- algorithms toolbox (with a bit of data structures)
  - graphs and graph algorithms
- algorithmic techniques toolbox
  - it's just a data structure!
  - modeling
    - solving problems with graphs
  - divide-and-conquer
  - incremental buildup
    - iterative algorithms
  - series of choices
    - greedy algorithms
    - recursive backtracking
    - dynamic programming
- real world stuff
  - a bigger toolbox
  - identifying and dealing with hard problems
  - implementation

Goals

- developing the skill of analyzing a problem and creating an efficient and provably correct solution to that problem
  - includes both algorithm development and choice/design of data structures
- fostering an appreciation for the practical value of studying algorithms and data structures
- developing other skills useful in computer science
  - abstract thinking
  - comfort with the idea of tradeoffs
  - a habit of critical reflection and revision

An Observation

"This material is difficult. There is no getting around that."

[Jeff Edmonds, *How to Think About Algorithms*]

- there are lots of open problems!
- a (seemingly) very small change can turn an easy problem into a hard problem
- there are lots of techniques and clever tricks – experience helps a lot
  - (ask if there's something unfamiliar that we don't spend time on)

But we also accept this as a premise – a major focus is on tactics for trying to make progress.

- the goal is to understand various techniques, how to apply them, when they apply, and what you are aiming for even if you still get stuck on some individual problems

Another Observation

"Ask questions. Why is it done this way and not that way? Invent other algorithms for solving a problem. Then look for input instances for which your algorithm gives the wrong answer. Mathematics is not all linear thinking."

[Jeff Edmonds, *How to Think About Algorithms*]

Keeping this in mind will make you stronger at designing complete, correct, and efficient data structures and algorithms.

Prerequisites

- **C- or better in CPSC 225**

This means you are comfortable with Java programming, including –

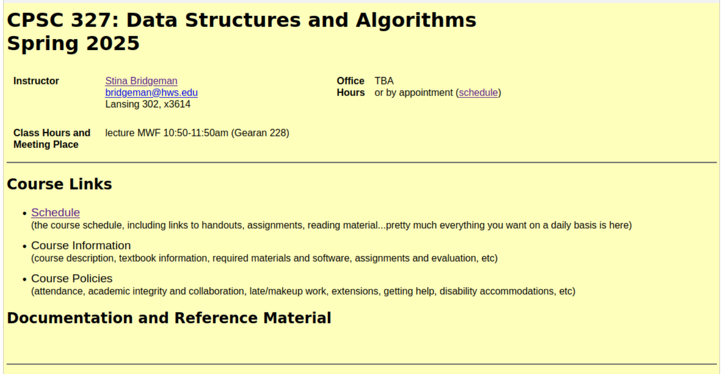
- fundamental programming constructs and concepts (variables, assignment statements, conditionals, loops, static methods)
- input and output, including reading from files
- fundamental OO programming constructs and concepts (classes, objects, inheritance, abstract classes, interfaces, polymorphism)
- constructing a program from specifications, without a lot of specific direction about how to achieve those specifications

This also means you are familiar with –

- basic abstract data types (lists, stacks, queues)
- basic data structures (arrays, linked lists, binary trees)
- recursion

Course Website

http://math.hws.edu/bridgeman/courses/327/s25/



CPSC 327: Data Structures and Algorithms
Spring 2025

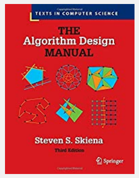
Instructor	Stina Bridgeman bridgeman@hws.edu Lansing 302, x3614	**Office Hours**	TBA or by appointment (schedule)
Class Hours and Meeting Place	lecture MWF 10:50-11:50am (Gearan 228)		

Course Links

- Schedule
  (the course schedule, including links to handouts, assignments, reading material...pretty much everything you want on a daily basis is here)
- Course Information
  (course description, textbook information, required materials and software, assignments and evaluation, etc)
- Course Policies
  (attendance, academic integrity and collaboration, late/makeup work, extensions, getting help, disability accommodations, etc)

Documentation and Reference Material

Course Materials

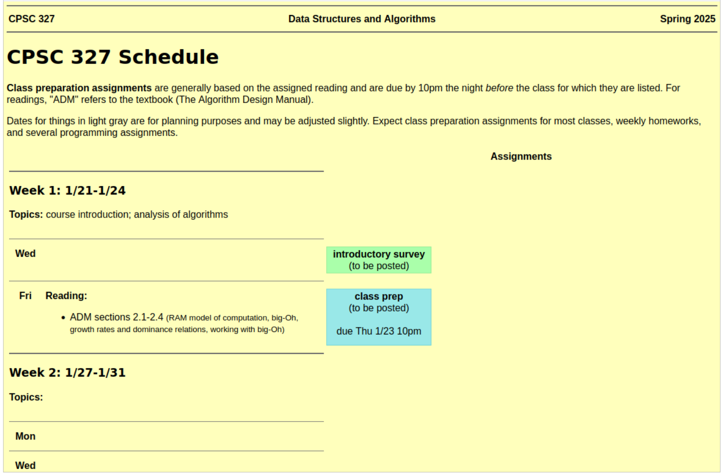


- textbook is *The Algorithm Design Manual* by Steven Skiena, 3rd ed
  - focus is on practical algorithm design
  - also has an extensive reference of data structures and algorithms that are of practical use

- all of the necessary software is available on the lab machines in Rosenberg 009 and Lansing 310 and via the Linux VDI
  - it is also possible to set up your own computer (optional)

Schedule Page

check here for readings, assignments, handouts, examples from class, etc



CPSC 327 **Data Structures and Algorithms** **Spring 2025**

CPSC 327 Schedule

Class preparation assignments are generally based on the assigned reading and are due by 10pm the night *before* the class for which they are listed. For readings, "ADM" refers to the textbook (The Algorithm Design Manual).

Dates for things in light gray are for planning purposes and may be adjusted slightly. Expect class preparation assignments for most classes, weekly homeworks, and several programming assignments.

	Assignments
Week 1: 1/21-1/24 **Topics:** course introduction; analysis of algorithms	
Wed	**introductory survey** (to be posted)
Fri **Reading:** • ADM sections 2.1-2.4 (RAM model of computation, big-Oh, growth rates and dominance relations, working with big-Oh)	**class prep** (to be posted) due Thu 1/23 10pm
Week 2: 1/27-1/31 **Topics:**	
Mon	
Wed	

Assignments

- the purpose of assignments is to gain knowledge and practice skills

- readings and class prep assignments introduce new material
  - due the night before class
  - expect some reading and a short assignment for most classes

- homeworks and programming assignments are an opportunity to practice and learn the material
  - expect weekly homeworks
  - expect several programming assignments, primarily in the second part of the semester

Assessment

- homeworks and programming assignments are an opportunity for formative assessment
  - get feedback and (in most cases) an opportunity to revise and resubmit
- summative assessment is based on in-class exams and in-person interviews for design problems and programming assignments

- final grade
  - 30% engagement
    - based on effort and achievement on homeworks and programming assignments, effort on class prep assignments, attendance
  - 70% mastery
    - based on demonstration of competencies through in-class exams, design problem and programming assignment interviews
  - must achieve a passing mastery grade in order to pass the course regardless of the engagement grade

Expectations

- attend all scheduled class sessions
- spend approx. 8 hours per week outside of class on reading, completing assignments, and studying
  - you may need to spend more
  - if you routinely spend significantly less, you may not be sufficiently mastering the material
- attend 5 hours of additional activities over the course of the semester
  - problem and programming assignment interviews (required, 1-2 hours total)
  - help/skills sessions for programming assignments
  - office hours

Generative AI and Other Resources

- generative AI can be used in multiple ways, both good and bad
  - as a professional tool
    - e.g. for code generation, testing and debugging, an unreliable peer to bounce ideas off of
  - as a learning aid
    - to help you understand something new
  - as a learning cheat
    - to get the end result without learning the process of how to get there for yourself
- (this applies to all other resources as well – YouTube, other websites, friends and colleagues, etc)

Generative AI and Other Resources

- **only authorized resources – and no AI – may be used on exams and during in-person interviews**
  - your grade in the course should reflect *your* knowledge and skills
- some places where AI may be used appropriately as a professional tool and/or learning aid will be discussed
  - AI use is always optional (reasons you may not want to use it will also be discussed)
- you should not use AI (or other resources) as a learning cheat on homework or other assignments
  - where detected, you will receive a 0 (or similar low score) for engagement on that assignment
  - final grade is heavily based on in-person assessments so you shortchange yourself if you bypass learning