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?

Outline

- 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
  - modeling
    - solving problems with graphs
  - divide-and-conquer
  - incremental buildup
    - iterative algorithms
  - series of choices
    - greedy algorithms
    - recursive backtracking
    - dynamic programming
- doing better
  - improving running times
  - identifying and dealing with hard problems

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 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 Materials

http://math.hws.edu/bridgeman/courses/327/s24/

CPSC 327: Data Structures and Algorithms
Spring 2024

Instructor	Stina Bridgeman bridgeman@hws.edu Lansing 302, x3614	**Office Hours**	study session (327-specific): TBD drop-in office hours: TBD or by appointment (schedule)
Class Hours and Meeting Place	lecture MWF 10:50-11:50am - Stern 304		

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)
- Coding Standards
  (expected naming, formatting, commenting standards for your programs)

Documentation and Reference Material

- Using Linux at HWS
  (lots of useful information about the Linux systems at HWS)
- Java 17 API (or google `java 17 classname`)

Schedule Page

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

CPSC 327 Schedule

Reading is to be done for the class period where it is listed; "ADM" refers to the textbook (The Algorithm Design Manual). **Warmups** are due by 10pm the night before the class for which they are listed.

Dates for things in light gray are for planning purposes and may be adjusted slightly. Expect homework problems for most class periods.

		Assignments
Week 1: 1/22-1/26		
Topics: course introduction; analysis of algorithms		
Mon		
Wed	**Reading:** • ADM sections 2.1-2.5 (RAM model of computation, big-Oh, growth rates and dominance relations, working with big-Oh, reasoning about efficiency)	**warmup** (on Canvas, under "Quizzes")
Fri	**Reading:** • big-Oh for sums • ADM sections 2.7-2.8, logarithms and exponents reference (logarithms and exponents) • big-Oh for recursive algorithms • ADM section 2.9 (war story) • (optional) ADM sections 2.6, 5.3-5.4 (behind the sums and recurrence relations tables)	**warmup** (on Canvas, under "Quizzes")
Week 2: 1/29-2/2		
Topics:		

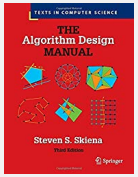
Homework

- complete the introductory survey (see the schedule page)
  - includes scheduling problem sessions
- consult the schedule page for the reading and warmup for Wed (warmup due Tue 10pm)
  - we begin by considering algorithmic efficiency
- review the policies and other information available on the course web page

http://math.hws.edu/bridgeman/courses/327/s24/

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
  - it is also possible to set up your own computer (optional)
- if you do not have a Linux account or do not remember the password, let me know

Expectations

- attend all scheduled class sessions
- attend 5 hours of problem sessions over the course of the semester
  - schedule will be established starting next week
  - for additional help or if you can't attend the problem sessions, the expectation can also be met by coming to office hours
  - 5 hours = e.g. 1 problem session every 3 weeks or a 20-minute office hours visit every week
- spend approx. 8 hours per week outside of class on reading, assignments, and studying
  - you may need to spend more
  - if you routinely spend significantly less, you may not be sufficiently mastering the material

Assignments and Evaluation

exam dates are on the schedule page

- reading and warmups – first exposure to the material
  - reading will be addressed in class on the day it is listed
  - warmups due at 10pm on the day before class – graded on effort rather than correctness
- homework and programming exercises [50%] – opportunity for practice
  - mostly written, some programming
  - graded on correctness, with the opportunity to revise and resubmit
- exams [40%] – demonstration of mastery
  - three midterms, covering essential tools and toolbox material (data structures, graphs, graph algorithms) from the first half of the course
  - final exam, covering data structure and algorithm design material from (mostly) the second half of the course
  - all take home

Assignments and Evaluation

key dates are on the schedule page

- engagement [10%] – your active participation in the learning process
  - warmups, attendance, participating in class, responding to “do you have any questions?” prompts, coming to office hours, …
  - minimum expectation for a passing grade
    - complete 50% of the warmups
    - satisfy the attendance policy
  - for full credit, demonstrate greater engagement through some of the following –
    - completing more warmups
    - missing fewer classes
    - participating in class
    - regularly responding to “do you have questions?” prompts
    - coming to office hours

Etiquette

- arriving late, leaving early, and coming and going during class is distracting
  - please do your best to arrive on time, and to take care of any necessary business before or after class so you can stay for the whole period
    - habitual late arrivals or early departures may be marked as an absence
  - let me know if you know in advance that you need to miss part or all of a class

Policies

- attendance
  - this is an in-person course, and being in class is an important part of the course
    - students who regularly miss class often do not do as well
  - if you must miss class, it is your responsibility to be aware of and make up missed content
    - check the schedule page for material from class and new assignments
    - come to the problem session or office hours if you have questions
  - in addition –
    - more than 6 absences will lower your engagement grade
    - 4, 5, or 6 absences will lower your engagement grade unless you are proactive about managing your absences and demonstrate engagement in other ways
      - plan ahead when an absence is known about in advance
      - take steps to promptly make up missed content
      - communicate when you’ll be absent and what you are doing to catch up
      - do warmups, participate in class, come to office hours, …

Policies

- there is a steady workload, and often multiple things in progress at one time – late handins can quickly snowball into falling behind
- foundational topics introduced earlier are used and built on later
- late policy
  - partial credit for warmups handed in after 10pm but by class time, no credit after that
  - homework and programming assignments will be graded twice – once after the initial due date and once after the resubmit deadline
    - can be handed in for full credit until the resubmit deadline, but anything handed in after the first grading cannot be resubmitted
    - no credit for handin after the resubmit has been graded
  - every effort should be made to hand in work on time, even if you are absent from class that day
    - emailing is acceptable if it is not possible to hand in a hardcopy on time, but should not become a regular practice or routinely substitute for poor planning
    - if a last-minute emergency prevents handing in your work on time, let me know as soon as possible
  - no work can be handed in after the end of the final exam timeslot
- extensions
  - no extensions for warmups
  - for homework and programming assignments, extensions which allow for an initial handin and/or a resubmit after the resubmit deadline should be rare and will only be considered for circumstances (such as illness) that are outside your control and which significantly impact your ability to focus on academics for several days or more

Policies

- exams
  - if you have an unavoidable conflict with the date(s), see me in advance to discuss rescheduling
  - if an illness, personal or family emergency, or other crisis outside of your control occurs during an exam, contact me as soon as possible
  - otherwise exams are not accepted late

Policies

- academic integrity and collaboration
  - copying part or all of someone else's program or solution is expressly prohibited, and it is not acceptable to be in possession of someone else's program or solution before you have handed in your own
  - exams demonstrate mastery
    - solely your work, and only those resources explicitly authorized in the instructions
  - homework and programming assignments for practice and learning
    - can discuss aspects of the problem with others, but the goal is for *you* to master the material – you should attempt the problem yourself first, and must write up your solution independently and in your own words (*you may not collaboratively write a solution, copy a solution from others, or use a solution written by someone else "as a guide" for your solution)*
  - warmups are about self-assessment
    - the point is to identify your level of understanding – it should be solely your work
    - discussion with others after you've submitted your answers is encouraged
    - graded on effort rather than correctness

Policies

- academic integrity and collaboration
  - unless otherwise prohibited, other materials (such as reference books or websites) may be used as technical references to learn about a particular topic, however **looking for and/or copying a solution is *not* acceptable**, even if you make modifications
    - learning from examples is valuable – with an example, you have to understand it in order to able to apply what it is illustrating to constructing your own solution
    - with solutions, you take someone else's work largely as-is and do not need to understand much about it
  - AI systems (e.g. ChatGPT, Codex), homework help or study aid sites (e.g. Chegg, Course Hero), and sites where you post a homework problem or question and solicit answers from others **may *not* be used in the completion of graded work**
    - use of generative AI for study purposes is discouraged because it is an unreliable source
  - this should be a learning environment, not a surveillance state

Being Successful

- stay on top of the material – things will pile up quickly if you fall behind
  - regularly utilize problem sessions and office hours
  - be organized in managing deadlines
- ask questions / come to problem sessions and office hours – don't wait if something is confusing or doesn't make sense
  - office hours are drop-in – no appointment needed
  - email and/or make an appointment if you can't come to the problem session or office hours
- be proactive if you must miss class, especially if you miss several classes in a row, or if there is an ongoing issue which adversely affects your work
- helpful practices
  - let me know if there are things about how the class is run that would be helpful for your learning
- disability accommodations
  - see the syllabus statement from Disability Services (on the Policies page)