Computer Science 210: Data Structures

Introduction
Welcome to Data Structures!

- Data structures are fundamental building blocks of algorithms and programs

- Csci 210 is a study of data structures
  - abstract data structures
  - design
  - analysis
  - implementation
  - use

- Prerequisites:
  - csci 101 (at Bowdoin or in high-school)
  - In other words
    - beginner knowledge of programming (in Java)
    - enjoy programming and problem solving
    - have a desire to learn more of it
Logistics

- **Instructor: Laura Toma**
  - office: Searles 219
  - Office hours:
    - T, Th 2-4pm
    - quick questions any time I am in the office
- **TA: Yuna Oh**
  - office hours: TBA
- **Textbook:**
  - Lewis & Chase
  - online: Sedgewick & Wayne, Programming in Java
- **Website:**
  - http://www.bowdoin.edu/~ltoma/teaching/cs210/fall09/
  - class not maintained on Blackboard
Course outline

• Week 1: Searching and sorting.
• Week 2: Analysis of algorithms.
• Week 3: Object-oriented (OO) concepts.
• Week 4: Stacks and queues.
• Week 5: Linked lists.
• Week 6, 7: Recursion.
• ------ Exam 1
• Week 8: Searching and backtracking. Breadth- and depth-first search
• Week 9: Trees and search trees.
• Week 10: Balanced binary search trees.
• Week 11: Priority queues.
• Week 12: Maps and hash tables.
• Week 13, 14: Networks.
• ------ Exam 2
Work and grading policy

• Class work:
  • weekly homeworks and lab assignments (approx. 50%)
  • 2 exams (approx. 50%)
  • readings, class participation, in-class assignments

• The class is programming-intensive

• Lab assignments are not meant to be finished during lab time. You have one week to complete them.
  • Handing in: hard copy + email
  • on hard copy sign that you have followed class honor code

• Lab work: team of <= 2 people

• Late policy: 25% per day
  • Why? it is absolutely essential that you do not fall behind
  • failure to turn in a lab ==> fail the class
Honor code

- Students are expected to follow the Bowdoin Computer Use Policy and the Academic Honor Code.
- You are encouraged to discuss ideas and techniques broadly with other class members, but not specifics of assigned problems except as part of group projects.
- Discussions should be limited to questions that can be asked and answered without using any written medium (e.g. pencil and paper or email).
- This means that at no time should a student read any code written by another student unless they are part of the same group.
- Sharing of code or intermediate designs is expressly prohibited.
- The same rules apply once you have finished the course: sharing your code with other students will be considered a violation of Bowdoin's honor code.
- Violation of this policy is grounds for me to initiate an action that would be filed with the Dean's office and would come before the J Board.
- If you have any questions about this policy, PLEASE do not hesitate to contact me. This will be a zero-tolerance policy.
- Don’t leave lab for the last night!
- Just submit what you have, even if not working.
More about the class

- The class is about designing, analyzing, implementing and using fundamental data structures.

- 101
  - you learnt how to use the basic constructs in Java. Put it differently, you learnt how to use a hammer and saw.
  - focus was on learning the tools available when writing a program
    - syntax, conditionals, loops, arrays, etc

- 210
  - Knowing how to use a hammer and saw does not mean you can build a house. In 210 you’ll learn how to build a house.
  - you’ll learn more tools, and most importantly you’ll learn to put them together to create a large program
  - and...you’ll learn to LEARN
More about the class

• It is occasionally programming intensive

• However ... is NOT about programming
  • but about **programming methodology**
    • design + analysis + programming + debugging

• **Programming language: Java. Why?**
  • makes graphics and web applications easy
  • available on all platforms
  • new language, in fashion

• **Most of the class will be independent of Java**
  • maybe next semester ...Python?
    • You’ll learn to distinguish between Java questions (check the Java doc pages to answer), and language-independent questions

• **Java graphics NOT the core of the class**
  • Java graphics will be used to improve the interface of your programs
  • Usually the graphical part will be given, you’ll need to write the “core”
Labs

- Attendance is mandatory

- The labs are not meant to be finished during lab time
  - Labs due one week after they are assigned; they are part of your homework

- Labs are not all equal
  - in general, progressively harder
  - at the beginning: lots of info guiding you towards the solution

- The labs are not always connected to the topic studied in class that week
  - the focus is on solving a problem

- The labs are often harder than they look. You’ll spend a lot of time understanding what the task is. It is a good idea to read the lab beforehand, so that you can ask many questions during lab time.

- Labs are challenging and fun. They are the most important learning tool
  - you will learn in class
  - you will REALLY learn while working out the labs
  - times the process will seem painful, and occasionally you will find a lab unfair.

- However, at the end of the class you’ll find that you’ve learned a lot.
Expectations

• TOGETHER
  • During class time we’ll talk about data structures concepts, we’ll analyze various options and we’ll work out the implementation details for some of these options
  • Often during class-time we’ll program together as a group
  • Occasionally there’ll be in-class assignments and team work

• YOU
  • you’ll learn to think like a computer scientist
  • you’ll learn to find out what it takes to get a task accomplished
  • you’ll start your lab in a timely manner
  • The bulk of your effort will be to get the lab assignments to work
  • You need to develop your code so that it can be debugged!!
    • flowchart, develop incrementally, debug, test
Scenario

- You develop all classes at once. Nothing works! HELP!!!
  - if code has too many errors, their combinations are infinite ==> impossible to debug
  - Moral: structure your code so that you implement one feature at a time, you debug it and test it, and then move on.

- You get stuck in Java graphics (GUI) before solving the actual problem.
  - Why don’t the buttons show?
  - Moral: Solve the core of the problem first, with a simple interface. If you have time at the end you can make your GUI more fancy.
More Expectations

- Problem: various backgrounds
  - 101 A vs. 101B
  - 101 vs. highschool
  - highschool 1 vs. highschool 2

- Willingness to work in a group environment
- Patience with material that is not new and when class is slow
  - participate
  - share with others
- Ask plenty of questions when something is unclear
- Goal: we want to work as a class
Class Outcomes

• You will learn the fundamental data structures:
  • lists, vectors stacks, queues, priority queues, trees, hash tables and maps

• Design: you will learn to model and come up with a solution to a problem
  • modularity, data abstraction, building blocks

• Analysis: you will learn to analyze the efficiency of your solution
  • you will learn to use efficiency considerations to decide the choice of data structures

• Program development: you will learn the importance of each step in getting a program to work: design, debug, test
  • Practice of programming:
    • Simplicity
    • clarity
    • generality

• You’ll learn to find out what it takes to get a task accomplished
This being said...

• Yes, 210 will be challenging

• But, most of the people who take 210
  • like it
  • say it is one of the most fun classes they took
  • continue with Computer Science
    • 210 is the pre-requisite for all other classes
    • If you like 210, you should think about majoring or minoring in computer science

• You are all here because you liked 101

• Welcome, and hope you’ll stay!
The major in Computer Science

1 entry-level class

101 Introduction to CS

5 core classes

210 Data Structures

cs 231 Algorithms

cs 270 Artificial Intelligence

cs 289 Theory of Computation

Math 200 Introduction to mathematical reasoning

3 electives

320 Robotics

340 Spatial Data Structures

350 GIS

355 Cognitive Architecture

360 Computer and Network Security

375 Optimization and Uncertainty

380 Computer Graphics

260 Software Design

291-294 Independent Study

401-404 Advanced Independent Study