Course Overview

Course Description

Reviews basic data structures. Covers the mechanics and relative efficiencies of advanced data structures. Students will implement several data structures such as AVL trees, heaps, hash tables, and adjacency lists. Discusses abstract data types such as maps, priority queues, and graphs. Introduction to algorithm analysis, sorting algorithms, and graph algorithms. 4 Credits.

Course Goals

- Provide an introduction to the analysis of algorithms
- Introduce some sorting algorithms and compare them to those already known
- Introduce more advanced data structures
- Provide insight as to how basic and advanced data structures may be implemented

Student Outcomes

Program outcome 1: Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. In this course, students perform basic analysis of code and algorithms for time complexity.

- Describe how an algorithm works to solve a problem, and describe its time complexity
- Apply well-known algorithms to problem instances (either by hand or in code), such as Huffman coding, minimum-spanning-tree, and shortest-path algorithms.
- Identify the time-complexities of key operations of common data structures
- Combine known data structures to propose a solution to solve a problem efficiently

Program outcome 2: Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. In this course, students implement data structures and algorithms covered in lecture; these programs are thoroughly evaluated for correctness, with some evaluated for performance; students implement unit testing for selected programs.

- Implement data structures such as heaps, tries, linked-lists, balanced binary search trees, priority queues, hash tables, and graphs
- Use an IDE like IntelliJ or VS Code with debugging tools to find and fix errors in programs
- Write unit tests using a standard testing framework like JUnit

Prerequisites

- CS416 or CS417 or CS419

Lecture

- Section 01: MWF 10:10am-11:00am KING S145

Labs

- Lab 01: W 12:40pm-2:00pm KING N218
- Lab 02: W 2:10pm-3:30pm KING N218
- Lab 03: Hours Arranged

Faculty & Teaching Assistants

Instructor - Section 01:

- Michael Kulik, Lecturer, Department of Computer Science
- Michael.Kulik@unh.edu
- Kingsbury W235
- Office hours: T 11:00am-12:30pm, W 2:10pm-3:00pm, R 2:00pm-3:30pm, and by appointment

Teaching Assistant

- Chris Nippert
- Chris.Nippert@unh.edu
- Office hours: T 2:00pm-3:00pm on Microsoft Teams and R 11:00am-12:00pm in Kingsbury W244

Evaluation

Assesment Type	Percentage
Programming Assignments	36%
Labs	24%
Exams	30%
Worksheets	10%

Programming Assignments

- There will be tentatively 9 full-credit **individual** programming assignments. See intro slides for details on academic integrity. Workload for many assignments will be significant.
- Assignments are generally due **at 11:59pm** on the given due date (usually Friday). You have at least a 15-minute grace period (until at least 12:14 am the next day) to submit your program. Submissions are done using Gradescope. There are no late day submissions.
- You may resubmit any programming assignments within the next full day after the initial grading results are released. For example, if grading results are returned on Monday, then you will have until Tuesday night to make and submit changes. We will also release the tests we used for grading at that point. Your grade for the assignment will be calculated as follows:

  $$\text{Final Grade} = \text{If } F < S \text{ then } 0.2 * F + 0.8 * S \text{ else } F$$

  Where F is your score on the first submission and S is your score on the second submission. **You must turn in a first submission for a second submission to count!**
- Students will not be given extensions for programs unless they have extenuating circumstances as determined by the instructor.
- Programs will be graded mainly on "external correctness" (behavior). They may at times also be graded on "internal correctness" (style and design). Disputes about homework grading must be made to your TA (cc instructor) within 1 week of receiving the second program grade.
- The lowest program grade will be dropped (only one assignment).

Labs

- You should not always expect to finish the lab during the lab period. In most cases, the effort required to complete the lab will be significant (this is part of why they are each worth over half as much as an assignment).
- You will receive 50% credit on the point deduction of your submission if you have attended the lab. Attendance will be taken during your scheduled lab section.

  $\text{Final Grade} = \text{If attended lab } 50\% + 0.5 * L \text{ else } L$

  Where L is your score for the lab.
- Labs are generally due at 11:59pm of the due date. You have at least a 15-minute grace period (until at least 12:14 am the next day) to submit your lab programs. Submissions are done using Gradescope.
- You are encouraged to work with a lab partner for all lab assignments. You may share code with your lab partner for labs (only!) but each of you must turn in separate submissions.
- There will be no make-ups for labs except in extreme circumstances. Late submissions are not accepted. Disputes about lab grading should be made to a course TA within 1 week of receiving the grade.
- The lowest lab grade will be dropped (only one lab).

Exams

- There will be two in-person exams, both near the end of the semester: Each exam will focus on specific aspects of the course, as will be specified before the exams.
- Make-up exams will be given only in case of a serious emergency. You must show evidence that you are unable to take the exam, such as a doctor's note. No make-ups will be granted for personal reasons such as travel or leisure.

Worksheets

- The completion of these exercises will be used as a basis for measuring your class participation and for you to use as a study aid for exams.
- The lowest 2 in-class exercise grades will be dropped.

Texts and Resources

The textbook for this course is: *Data Structures and Algorithms in Java, 6th Edition by Goodrich et al.*

We will be using Canvas (myCourses) for course announcements, listings of resources, and grading.

Temporary Academic Supports for Extended Absences with Letter

If you are dealing with an unexpected, extenuating circumstance that will keep you out of class or affect your performance for more than a day or two, reach out to the Dean of Students (dean.students@unh.edu) to request a letter be sent to all your faculty.

Accessibility Services

According to the Americans with Disabilities Act (as amended, 2008), each student with a disability has the right to request services from UNH to accommodate his/her/their disability. If you are a student with a documented disability or believe you may have a disability that requires accommodations, please contact Student Accessibility Services (SAS) at 227 Smith Hall, or sas.office@unh.edu.

Accommodation letters are created by SAS with the student. Please follow-up with your instructor as soon as possible to ensure timely implementation of the accommodation identified in the letter.

For more information refer to www.unh.edu/sas or contact SAS at 603.862.2607, 711 (Relay NH), or 227 Smith Hall, or sas.office@unh.edu

Academic Integrity

While you are allowed to work together on worksheets and labs, you are expected to do your own work for programming assignments. If you use other resources, such as code from other people, or code found online or through AI tools, such code must be clearly marked with clear specifics about the source of that code, and the instructor or responsible TA must be notified at time of submission (so that we may adjust the credit given for the work you've done on your own). Otherwise, any code found not to be yours will be treated as a violation of the University Academic Integrity Policy. If we determine that cheating has occurred, the minimum penalty will be a zero for the assignment or exam, and the maximum penalty will be an F grade for the course. (If you relent and tell us before we discover it, the penalty may be lesser.)

Emotional or Mental Health Distress

Your academic success and overall mental health are very important. If, during the semester, you find you are experiencing emotional or mental health issues, please contact the University's Psychological and Counseling Services (PACS, **3rd floor, Smith Hall; 603 862-2090/TTY: 7-1-1**), which provides counseling appointments and other mental health services. If urgent, students may call PACS M-F, 8 a.m.-5 p.m., and schedule an Urgent Same-Day Appointment.

Support for Academic Success

Center for Academic Resources (CFAR) is where students go to improve their study skills, time management, and understanding of UNH's academic culture. Our professional educational counselors and peer academic mentors work within students' course materials to demonstrate best practices for learning concepts and preparing for exams.

Knack is a Peer-to-Peer tutoring platform that will be available to all enrolled students for all undergraduate courses in Durham at no cost to students. Students looking for additional assistance outside of the classroom are advised to consider working with a peer tutor through Knack. UNH has partnered with Knack to provide students with access to verified tutors who have successfully completed this course. To view available tutors, visit http://unh.joinknack.com and sign in with your student account.

Confidentiality and Mandatory Reporting of Sexual Violence | Harassment

The University of New Hampshire and its faculty are committed to assuring a safe and productive educational environment for all students and for the university as a whole. To this end, the university requires faculty members to report to the university's Title IX Coordinator (Bo Zaryckyj, Bo.Zaryckyj@unh.edu, 603-862-2930/1527 TTY) any incidents of sexual violence and harassment shared by students. If you wish to speak to a confidential support service provider who does not have this reporting responsibility because their discussions with clients are subject to legal privilege, you can contact the SHARPP Center for Interpersonal Violence Awareness, Prevention, and Advocacy at (603) 862-7233/TTY (800) 735-2964. For more information about what happens when you report, how the university treats your information once a report is made to the Title IX Coordinator, your rights and reporting options at UNH (including anonymous reporting options) please visit student reporting options. The uSafeUS app is also available to keep reporting options and resources easily accessible on your phone.

Help us improve our campus and community climate. If you have observed or experienced an incident of bias, discrimination or harassment, please report the incident by contacting the Civil Rights & Equity Office at UNH.civilrights@unh.edu or TEL # (603) 862-2930 voice/ (603) 862-1527 TTY / 7-1-1 Relay NH, or visit the CREO website. Anonymous reports may be submitted.

Course Disruption

In the event of a major campus emergency, course requirements, deadlines and grading percentages are subject to change when necessitated by revised course delivery, semester calendar or other circumstances. Information about changes in this course can be obtained at the myCourses site or by contacting me via email. If the course is not able to meet face-to-face students should continue to check myCourses for announcements and updates to this syllabus as needed.

Specific Student Learning Outcomes

The following are the specific student learning outcomes we plan to cover in this course.

- Describe and implement stack, queue, and variations of list data structures (e.g. ring, doubly linked list, lists with sentinels).
- Implement a binary search tree data structure, including depth- and breadth-first traversals.
- Describe skip lists and understand the concept of randomized algorithms.
- Describe AVL trees, 2-3-4 trees, Red-Black trees and B-trees, along with common applications of them.
- Describe hashing and implement mechanisms to deal with collisions in a hash-table. E.g. linear probing, quadratic probing and double hashing.
- Describe tries and their applications in string search and matching.
- Describe and implement binary heaps. Implement the priority-queue ADT using lists and heaps. Understand and implement the Floyd's O(N) heap construction algorithm.
- Understand Map/Set ADT and compare the performance of various implementations using concrete data structures (e.g. linked lists, search trees, hash tables and tries).
- Understand when and how to use lambda expressions with collections and be able to implement methods for collections that accept lambda expressions.
- Select an appropriate data structure to implement an ADT under a given set of constraints. Determine the efficiency category of operations using big O notation.
- Understand the basic concept of amortized analysis.
- Analyze, compare, and implement various sorting algorithms, including bubble sort, insertion sort, selection sort, quicksort, merge sort, heap sort, bucket sort and radix sort. Understand the lower bound of comparison-based sorting algorithms.
- Describe and implement the graph ADT; implement depth-first and breadth-first graph traversals.
- Describe and implement the Disjoint-Set ADT and its application in graph algorithms.
- Describe and implement the minimum spanning tree algorithms: Prim's and Kruskal's.
- Describe Dijkstra's single source shortest path algorithm; Floyd-Warshall's all-pair shortest path algorithm.
- Select and use appropriate algorithm to solve graph problems.
- Describe algorithm classes:
  - Greedy algorithms: Dijkstra's, Prim's, Kruskal's algorithms; Huffman coding.
  - Divide and conquer: merge, quick sorts.
  - Dynamic programming: Floyd-Warshall's algorithm.
  - Randomized algorithms: skip list insert.
- Select and use appropriate abstract data types, data structures, and algorithms to solve moderately complex problems.
- Program testing and debugging.

Tentative Content Schedule

Week 1: Abstract data types (ADTs): Immutable lists; Algorithms & Big-O notation
Week 2: Unit testing concepts; Junit; Stacks; Queues
Week 3: Set & map ADTs; Unit testing methodologies, Set notation
Week 4: Recursion & recursive data structures; White-box testing methodologies; Binary search tree (BST) structure
Week 5: BST operations, Threaded BST's; Lambda expressions
Week 6: Algorithm design techniques; Heaps; Priority queue ADT; Tries; Huffman coding
Week 7: Using lambda expressions; Applying data structures to new problems; Graph ADT
Week 8: Disjoint Set ADT
Week 9: Sorting algorithms, especially merge-, heap-, quick-, bucket-, and radix-sorts
Week 10: Skip list structure, randomization; Minimum spanning-tree algorithms
Week 11: AVL balanced BST; Balanced BST's: 2-3-4 trees, B-Trees, and red-black trees; Shortest-path algorithms
Week 12: Hash table structure
Final weeks: Overview of structures like quad/R-trees, blocking queues; review & exam preparation