Computer Science 361
Information Structures
Fall 2018

Instructor: Dr. Chris Healy
My office is located in Room 200-I in Riley Hall. My office hours are MWF 9:30-1:20, TR 1:00-2:15, and also by appointment. Please see me if you ever have any questions during the course. Office phone number 294-2233 and e-mail address chris.healy@furman.edu

Class Meetings
MWF 1:30-2:20 in 204 Riley Hall. Labs are held Thursdays 2:30-4:30 in 203 Riley Hall.

Purpose
This course is an in-depth study of data structures and algorithms commonly used in computer science. We will look at new ways of solving problems, and closely examine the properties of data structures. With this knowledge, you will be able to solve a greater variety of problems, and write programs that are more efficient.

Web site
Course announcements, notes and handouts can be found here:
http://cs.furman.edu/~chealy/cs361

Textbook
Algorithm Design and Applications, by Michael Goodrich and Roberto Tamassia, published by Wiley, 2015. We will cover chapters 1-6, 8-16 plus some special topics.

Grade calculation

10% Labs	
20% Homework	
5% Quizzes	
20% Test #1	Friday, September 28, 2018
20% Test #2	Friday, November 9, 2018
25% Final	Tuesday December 18, 2018, 12:00-2:30

Please note the dates/times of these exams. Any appropriate documentation supporting special arrangements necessary for any test must be given to me during the first week of class.

Attendance
Furman's attendance policy states that you cannot pass a course if you miss more than one-quarter of the class meetings. If you miss a test, you will earn a score of zero, unless your absence is excused. If you know in advance that you cannot take a test, please let me know as soon as possible so that you can take it early. Otherwise, if you are absent from a test due to an excused

absence, then your final exam grade will substitute for that test's score. Travel plans are not an acceptable excuse for absences.

Preparation

Study includes reviewing notes, becoming acquainted with the material to be discussed in the next class, completing homework assignments and preparing for exams. Studying on a consistent schedule each day will work far better for you than cramming before a test. Don't forget the most important thing – I am here for you. Please come to my office anytime for help or advice in this course.

Homework and labs

In order to successfully complete a lab or a programming assignment, you need to do two things: give me a personal demonstration of how your program works, and send your source code to me via e-mail.

Deadlines and late policy

Type of activity	Deadline	Is late work accepted?
Labs	Six calendar days after the lab period	No
Homework (programming assignment)	This will be stated on the handout	Yes, but the submission will be penalized 10 percentage points. No work will be accepted more than 2 weeks late.

Guiding principles for students in Computer Science classes

1. We are here to learn and explore.
    a. Seek discussions with the instructor and classmates about the material to reinforce your understanding and practice communicating ideas.
    b. Have fun. Live in the moment (i.e. don't dwell too much on the difficulties of yesterday or tomorrow). Enjoy the journey and intellectual feast. Be enthusiastic about what you are doing.
2. You can be successful in this class. Every day is an opportunity for an epiphany. Don't let mistakes or setbacks hold you back. After some effort, things can suddenly click in your mind.
3. Learn by doing, not just passively reading, listening or watching.
   Each study period needs to have a clear goal.
   Pay attention to the big picture and the facts that you are collecting.
4. Be organized: Take notes on what you read. Review earlier material as needed. Create a cumulative study outline every few weeks. Maintain a portfolio of your work.
5. Be patient when solving a homework or lab problem.
    a. There is no need to rush.

Don’t worry if your first attempt at a solution is wrong.
Read all instructions and be methodical.
Take time to gather your thoughts.
Deliberately write out your thought process and plan of attack.

   b. A computer program or other homework assignment may take up to several hours to complete. In programs you need to comment your code as you go, because you will quickly forget what looks obvious right now! Realize that you don’t need to finish everything in one sitting.
   c. Break up large problems into small, more manageable pieces.
   d. Don’t get bogged down with too many mechanical details. Computing is all about removing tedium from routine tasks.

6. Be curious, and always ask questions.
   a. Find a topic or application that you are enthusiastic about.
   b. Consider alternative solutions to a problem.
   c. When finishing a problem, ask yourself if this problem or its solution lends itself to other problems.
7. Computer science is about logic, structured thinking, information, communication and problem solving. Thus, it has connections to many other fields in the sciences, humanities and social sciences. You will find the analytical techniques useful in your career.

<u>Outline of course objectives</u>

1. What is a data structure
   a. always consider list of desired operations; representation
   b. which operations you want to optimize may influence representation & implementation
2. Algorithm efficiency: analysis
   a. asymptotic bounds, big O and its relatives
   b. amortization technique
3. Linear data structures
   a. write down set of possible desired ops -> becomes an ADT such as "vector" or "list" or "sequence". Don't memorize differences between names.
   b. array vs. list implementation tradeoffs
4. Trees
   a. purpose, definitions and terminology
   b. desired operations
   c. binary trees, generalizations of them
   d. operations specific to binary trees
   e. properties: bounding number of leaves, etc.
   f. review of expression trees and postfix evaluation

   - g. review of simple traversals
   - h. Euler tour traversal
   - i. representation of trees

5. Priority Queue / heap
   - a. motivation
   - b. desired operations
   - c. representation and implementation
   - d. insertion
   - e. deletion
   - f. heapsort
   - g. building entire heap out of a list, including its big O analysis
6. Dictionary ADT
   - a. Definition, purpose
   - b. desired operations
   - c. overview of implementation and complexity of operations
   - d. Hash table implementation
   - e. How hashing works, good hash functions
   - f. Resolving collisions (quadratic probing, double hashing, etc.)
7. Sorted dictionary ADT
   - a. operations
   - b. bad and good representation/implementations (which leads into next subject)
8. Binary search tree
   - a. review how to search, insert, delete…
9. Improvements to BST
   - a. why?
   - b. AVL tree
     - i. definition, properties
     - ii. insertion, deletion (with examples)
     - iii. rebalancing!
   - c. B-tree (non binary tree)
     - i. motivate why we'd want non-binary tree
     - ii. insertion, deletion (as always, consider examples and analyze complexity)
   - d. Red-black tree
     - i. definition, properties
     - ii. analogy to B tree, can convert both ways
     - iii. insertion & deletion
   - e. Splay tree

            i. rationale
            ii. the splay operation
            iii. insert, delete, search operations
    f. skip list (non-tree solution)
            i. definition, example, observations
            ii. how to search, insert, delete
    g. analysis of bounding splay cost
    h. analysis of skip search (much less tedious to look at than splay tree)

10. Sorting - review as needed
    a. complexity of analysis of various techniques
    b. Solving recurrences for merge and stooge sort. Technique useful later for other divide-conquer problems.
11. Set ADT
12. Finding the k-th smallest value
    a. randomized quick select
    b. proof of its complexity
13. Greedy algorithms
    a. general features to look out for
    b. example: maximizing # jobs to accomplish
    c. example: minimizing # rooms needed
    d. argue optimality
    e. example: Fractional knapsack
    f. Get into groups to discuss examples from Kleinberg book
14. Divide & Conquer
    a. Master theorem
    b. example: finding inversions in 2 sets of rankings
    c. example: matrix mult, including its complexity
    d. example: closest pair of points
15. Dynamic programming
    a. motivation
    b. example: Fibonacci numbers
    c. example: Playoff winning probabilities
    d. example: Matrix chain mult
    e. example: Knapsack problem
    f. example: Longest common subsequence
    g. Get into groups to practice examples from Kleinberg book
16. Genetic algorithms
17. Graphs

   a. definitions, assumptions, purpose, rep'n
   b. design: attributes & desired ops
   c. review BFS and DFS if needed
   d. applications of DFS

18. Biconnected components
19. Directed graphs
    a. properties
    b. revisit DFS
    c. Transitive closure: Floyd-Warshall algorithm
20. DAG
    a. topological ordering: proof gives algorithm
21. Weighted graphs
    a. review rep'n
    b. 3 algorithms related to single-source shortest paths
       i. review Dijkstra's algorithm
       ii. negative weights: Bellman-Ford algorithm
       iii. DAG shortest paths algorithm relies on topological ordering
    c. all-pairs shortest paths
       i. Floyd-Warshall again
    d. Minimum spanning tree
       i. Boruvka
       ii. Kruskal
       iii. Prim
    e. Minimum Steiner tree
22. Flow networks
    a. definition, properties, observations
    b. cut properties
    c. Ford-Fulkerson algorithm
       i. residual capacity and residual network
       ii. augmenting path
       iii. now we're ready for the algorithm
    d. Bipartite matching
    e. Circulation with demand
       i. ex. Airline scheduling
    f. Examples from Kleinberg book

Tentative pacing schedule

Week of	Topics
8/27	(Class begins Wednesday) Chapter 1: Exact and asymptotic analysis
9/3	(No class on Labor Day) Chapters 1-2: Amortized analysis; lists, trees, binary trees
9/10	Chapters 2, through section 4.2: Binary tree properties, applications, traversals, implementation, sorted dictionary, binary search tree, AVL tree
9/17	Sections 20.2, 4.3 and 4.5: B tree, red-black tree and splay tree
9/24	Section 19.6 and Chapter 5: Skip list, priority queue, heap insertion and deletion, heapsort, bottom-up heap creation, analysis of heaps (Test Friday)
10/1	Sections 6.1-6.3, Chapters 8-9: Dictionary ADT, hashing, review of sorting
10/8	(No class on 10/8) Section 9.2 and Chapter 10: Randomized quick select; Greedy algorithms
10/15	Chapters 10-11: Greedy algorithms, divide and conquer
10/22	Chapters 11-12: Closest pair problem, master theorem; Dynamic programming: playoffs, matrix chain multiplication
10/29	Chapter 12: 0/1 Knapsack problem, longest common subsequence
11/5	Chapter 12: Practice dynamic programming (Test Friday)
11/12	Genetic algorithms; Chapter 13: Graphs: breadth-first and depth-first searches,
11/19	(No class on 11/21 or 11/23; no lab on 11/22) Chapter 13: directed graphs: transitive closure, DAG, topological ordering
11/26	Chapters 14-15: Weighted graphs, Dijkstra's algorithm, Bellman-Ford algorithm; DAG shortest paths, Floyd-Warshall all pair shortest paths; Spanning trees
12/3	Chapter 16: Flow networks, residual capacity, augmenting path
12/10	(Last class is 12/10) Review
12/17	(Final exam is December 18 at 12:00)