Course Overview

CSE 3318 – Algorithms and Data Structures
University of Texas at Arlington

Algorithms and Data Structures

- *What is an Algorithm?*
  - A sequence of steps to follow for solving a problem
  - Examples of algorithms that you already implemented (think of what problems you solved):
  - TED: https://www.ted.com/talks/david_j_malan_what_s_an_algorithm
  - TED (book sorting): https://www.ted.com/talks/chand_john_what_s_the_fastest_way_to_alphabetize_your_bookshelf
- *What is a Data Structure?*
  - A way to store and organize the data that:
    - facilitates a specific type of processing of that data (e.g. hash tables for searching, arrays for direct access) or
    - matches the natural structure of the real-world data being modelled (e.g. use a tree to represent a family tree or an organization's hierarchy).

Data Structures	Specific Algorithms	Techniques & Theory
Array Linked list Queue (FIFO) Stack Priority queue – Heap Hash table Tree Search trees (binary, 2-3-4, etc) Graphs	Searching - linear, binary (in array) - using a search tree - Using a hash table - In a graph Sorting (several algorithms) - comparison-based - Non-comparison based Optimization: - Greedy and - Dynamic programming - Graph-related: MST, SPST Graph-related	Techniques: - Recursion - Divide and conquer - Greedy (optimization) - Dynamic programming (optimization) Theory (requires math) - Complexity: O, Θ, Ω (for iterative and recursive algorithms) - Recurrences - Summations - Logarithms
For all the algorithms we discuss we will look at: - their ***time complexity*** and ***space complexity***. - where they are applicable (e.g. for what data, or for what type of problem). - their pros and cons (what they are best for, how different algorithms that solve the same problem compare). For sorting algorithms we will discuss a set of properties.		

Course Objectives

MAIN goals:

1. Cover specific algorithms and data structures. For each discuss
   - Method,
   - Code/pseudocode
   - Time and space complexity – theory (requires math)
   - Applicability (what data it applies to for sorting, what problem it solves for graphs)
2. Problem solving/ develop your own algs/ reasoning about correctness of an algorithm/ read code and turn it into a verbal algorithm/ practice on leetcode and other resources
3. Touch on algs and DS in libraries
   - Hash table implementation in Java, also Arraylist implementation
   - Qsort in C
   - Other, at students request
4. Continue to practice writing code
   - Hws
   - Quiz questions- little or no partial credit
   - Weight on bug-free code: severe penalties for incorrect code
5. If possible, communication skills
   - Participate on discussion boards and group work (currently as part of class discussions and study groups)
   - Justification for pbs in quizzes (brief and to the point, either use only 5 words, or underline 3-5 words)

Sub goals

- Use web resources
- Practice good habits:
  - develop bug-free code (think professional, not homework),
  - use tools (e.g. Valgrind to catch memory leaks and other memory-related bugs),
  - create your own resources (e.g. list of steps and commands needed to compile on omega)
- Give counter example to show it does not have a property e.g. not stable - aligned with main goal 2 above
- Prove correctness using induction (if time permits)
- Gain confidence and be able to study on your own – student self driven task

How to study for this class?

Study in depth. If you just "get the idea" you will most likely fail the class.

- Pencil and paper
- Allocate time in your schedule for weekly study for this class. Set-up/join a study group.

Review and strengthen your programming skills with the goal of good standards and deep understanding

- review your CSE 1320 resources
- Other web resources
- Leetcode, leetcode, leetcode (or other)
- Deep understanding: do not just 'fix' the code, or use the solution that works, but understand why your first version was not working. Do not just take sample code and tweak it to solve your problem, but understand how that library function works. E.g. strtok
- Good standards:
  - good code,
  - develop test cases (read large data from files, use input redirection to easily store and run test cases),
  - print the data in a user friendly way that allows you to trace your code (print formatted tables)
  - Write modular code, be able to pass data as arguments by value or by reference. No global variables allowed in this class.

Study as we go, not only before the quiz

- Understand what we did in class and be able to reproduce it yourself at home (**compute time and space complexity, write code, simulate how data is moved by a certain algorithm** – these are the main categories of problems). You will see that you do not remember them as well as you thought even if you do it the same day. If you do not get the same answer as in the posted solution or as done in class, **meet with me or a TA to clarify any misunderstanding**). Good to work with a group and take turns explaining to the others – strengthens understanding and communication skills
- solve practice problems when we are covering that topic
- For each upcoming class review the past lecture and/or relevant material and math background (e.g. logs, summations). We do NOT have enough time to practice as much in class to where you can just learn it from class.
- Know all the topics covered in class, especially the major algorithms.
- Make notes of the TIME when a specific important aspect is discussed in class so that you can fast-forward the lecture video for it.
- Understand the homework problems and be able to write code on paper
- Be able to solve similar problems to those covered in class (or a mix of components of those)
- Create your own problems
- Work problems from leetcode and other resources as they strengthen both programming and problem-solving skills
- When studying, simulate the quiz set-up: time limit, write code on paper, use the cheat sheet, write clear answers

- ask questions
- follow up with an email/chat after discussion in class or after. Especially if I give some permission.
- submit multiple times and check final submission