Data Structures (CS-1202)

Course Information (Monsoon 2022)

Instructor: Soumyottam Chatterjee

August 4, 2022

1 Overview

Data Structures, or some variant thereof, is taught in almost every computer science (CS) program worldwide. It is considered to be one of the "core" courses in the undergraduate CS curriculum. And rightfully so. If *computer science* could, arguably, be defined as the study of *data*, then how we represent, store, organize, access, and manipulate that data is surely central to *all of* computer science. As we will see throughout this course, the choice of "data structures" for a particular collection of data is inextricably linked with the choice of and the performance of algorithms we may want to perform on that data. In fact, it is not unusual for a modern computer to take several days to finish some very common computational tasks if the programmer fails to choose the correct data structure for that task. We will look at such examples firsthand and we will see how to bring that execution time down to less than a second.

Throughout the course, a strong emphasis will be placed on measuring the performance of the data structures and algorithms we design. We will measure the performance in two ways. First, we will mathematically analyze the time complexities of our algorithms. This will usually be done using the asymptotic O ("Big-O") and Θ ("Big-Theta") notations. Second, we will measure the *actual* execution time of *actual* computer programs. Finally, we will observe how the execution times of computer programs correlate with the asymptotic time complexities of the underlying data structures and algorithms.

2 Choice of Programming Languages

The course material will be presented mostly in a language-independent fashion during the lectures. There's, however, necessarily going to be a large programming component (see Section 9), for which we're going to use C and Python. Specifically, for every programming problem that you're given, you'll be required to solve that problem twice — once in C and once in Python. The idea here is the following. Using C will force you to learn the inner workings of the data structures as well as clearly understand the conceptual differences between the *specification* of a data structure and the *implementation* of the said data structure.

Why bother with Python then? Why not use *only* C? Well, in many cases, using Python will allow us to forego focusing on the implementation part and focus solely on the specification or interface part of a data structure. This distinction follows necessarily from the design philosophies of the two languages — C was designed to be a systems programming language whereas Python was designed keeping code readability and brevity in mind. In other words, C is mainly concerned with improving machine efficiency whereas Python is concerned with improving programmer efficiency. In today's world, it is more likely that you'll be using much more Python than C (or than even C++) in your CS job/career. Thus, the experience that you're going to have during this course — implementing and playing with the most common data structures in Python — will come in very handy in your future programming endeavors.

For each of these two languages, an idiomatic usage of that language will be promoted in the class. Thus merely "translating" your C code into Python code (or the other way round) will *probably* not fetch you full credit. In the case of C, pointer handling will be given special attention. For both the languages, good design practices will be discussed.

3 Topics

1. Abstract Data Types (ADTs): Interface and Implementation
2. Sorting and Order Statistics
   (a) Sorting in $O(n^2)$-time
      i. Bubble Sort
      ii. Selection Sort
      iii. Insertion Sort
   (b) Sorting Lower Bounds
   (c) Sorting in Linear Time
      i. Counting Sort
      ii. Radix Sort
      iii. Bucket Sort
   (d) Median and Order Statistics
3. Basic Data Structures
   (a) Arrays and Lists
   (b) Stacks, Queues, and Linked Lists
   (c) Recursion and Stacks
      i. Quick Sort
      ii. Merge Sort
   (d) Direct Access Arrays, Hashing
4. Trees and Graphs
   (a) Binary Search Trees
   (b) Heaps, Heap Sort
   (c) Graph Representation
   (d) Elementary Graph Operations
      i. Breadth-First Search (BFS)
      ii. Depth-First Search (DFS)
      iii. Topological Sorting of Directed Acyclic Graphs (DAGs)
      iv. Connected Components
      v. Biconnected Components
      vi. Spanning Trees
   (e) Minimum Spanning Tree (MST) Algorithms
      i. Kruskal's
      ii. Prim's

(f) Single-Source Shortest Path Algorithms
    i. Bellman-Ford
    ii. Dijkstra's

5. Advanced Data Structures
   (a) Efficient Binary Search Trees
       i. AVL Trees
       ii. Red-Black Trees
       iii. Splay Trees
   (b) Skip Lists
   (c) Priority Queues
       i. Binomial Heaps
       ii. Fibonacci Heaps
       iii. Symmetric Min-Max Heaps
   (d) Data Structures for Disjoint Sets
   (e) B Trees, B+ Trees

4 Learning Outcome

After successfully completing this course, a student should be able to

1. implement the most common data structures in at least two different programming languages, namely, C and Python;
2. design and implement appropriate data structures for the various programming tasks that they encounter in their CS career;
3. in any particular situation, given the choice between multiple different data structures, be able to analyze the trade-offs associated with each of the choices, and make a good decision accordingly;
4. measure the performance of a program empirically;
5. analyze the performance of an algorithm mathematically;
6. solve common programming problems on a white board reasonably fast, and be able to do that in at least two different programming languages, namely, C and Python.

5 Teaching Assistants

To be decided.

6 Class Schedule

Monday and Thursday, 1:30 pm – 3 pm. The place is `AC03-003 (LT)`. Some classes may be held online.

7 Office Hours

To be decided.

8 Textbooks and Readings

Primary Reading Material. The GTG Book for Python [3].

Secondary Reading Materials.

1. The CLRS Book [2].
2. The K&R Book [4].
3. The lecture notes by James Aspnes [1].

9 Evaluation and Grading

The following is a tentative guideline for the grading policy and may undergo a change.

1. **Exams:** There will be biweekly exams — thus there will be 6 exams in total, each contributing 7%, *potentially*, to your final score. Out of the 6 exams, the best 4 scores will be used to calculate your final score. Thus the exams together will contribute 28% to your final score.

   In the exams, you will be tested on your knowledge and understanding of the data structures discussed in the lectures. You'll also be asked to provide *algorithmic* solutions using that knowledge, during which process you may need to design new data structures of your own. The exams will be theoretical in nature, i.e., you'll have to write your algorithms in a pseudocode format and using pen and paper. You'll also be required to mathematically analyze the time and space complexities of your algorithms.

2. **Homework Assignments:** There will be weekly homework assignments — thus there will be 12 homework assignments in total, each contributing, *potentially*, 8% to your final score. Out of the 12 homework assignments, the best 9 scores will be used to calculate your final score. Thus the homework assignments together will contribute 72% to your final score.

   In the homework assignments, you'll be asked to implement the data structures discussed in the lectures. You'll also be asked to solve certain algorithmic problems. Note that, unlike the class-tests, the homework assignments will be mostly programming-based. You're going to have to write actual computer code. In particular, if your code doesn't compile and run successfully, you get a zero on that homework problem, even if the underlying

algorithm is correct. You'll have to solve every problem twice — once using C and once using Python.

Numerical Score to Grade Conversion. Final grades will be assigned according to the "Indicative CGPA to Percentage Band" as shown in Table 1.[1]

10 Submission Policies

We're aware of the fact that you're juggling multiple activities and the assigned deadlines may not always be favorable. In order to meet deadlines, it would be best for you to start on your assignments *as soon as* they are handed out. Starting working on an assignment on the day of the submission is the worst idea you could possibly come up with. Historically, this has resulted in scores that are below average performance. Starting your work late also creates conditions for plagiarism, which is something that you should *definitely avoid* — at all costs.

You are expected to submit the assignments before the assigned deadline. We will enforce the following policies:

1. All assignments will have to be submitted electronically through the *Google Classroom* for the course.

2. The English language component of every assignment, if any, will have to be properly formatted using the LaTeX typesetting system.

3. Late submissions are possible. An assignment is counted as late, as soon as the clock for the assignment submission rolls over. For every extra day that you take to submit, the assignment will lose 25% of its value. Once the assignment loses 100% of its value, it will not be graded.

4. Extensions may be granted only in case of genuine emergencies, e.g., medical/health problems, family-related issues, etc. All such extensions will solely be at the instructor's discretion.

11 Cheating and Plagiarism

- The exams will be conducted in class and will be closed book. The only resource you'll be allowed to consult is your own mind.

- For the homework assignments, feel free to refer to the lectures or any of the *official* reading materials. Looking up or seeking help from *every other* source — including the internet — will be considered instances of plagiarism.

- For most of the assignments, you will be allowed to discuss the problems with your classmates. However, any work that you submit should be *completely* your own, and should be done in an individual setting. One rule of thumb that you can follow is the

[1] In Table 1, the symbol x denotes the aggregate numerical score accumulated — according to the policies outlines above — by you over the entirety of the course duration.

following. While discussing something with your peers, do *not* use any pen and paper (or laptop). That is, do not write down anything during the discussions. Allow yourself to take only *mental notes*, and at a later point in time, when you're in the solitude of your own room, work out the mental notes on paper.

- We will have *zero tolerance* for cheating — be it an assignment or an exam. Any evidence of academic integrity violations or plagiarism will result in an F grade for the entire course. *No exceptions* will ever be made. In addition, you will be reported to the OAA for academic integrity violation.
- If you have any, however small, questions about what counts as plagiarism — and what doesn't — please contact the instructor.

  "But I didn't know < `insert-the-thing-you-did` > was also counted as plagiarism"

  will *not* be considered an acceptable excuse.

References

[1] James Aspnes. Notes on Data Structures and Programming Techniques, May 2022. Accessed on: July 18th, 2022. URL: `http://www.cs.yale.edu/homes/aspnes/classes/223/notes.html`.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, fourth edition, April 2022. URL: `https://mitpress.mit.edu/books/introduction-algorithms-fourth-edition`.

[3] Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. *Data Structures and Algorithms in Python*. Wiley, first edition, March 2013. URL: `https://www.wiley.com/en-us/Data+Structures+and+Algorithms+in+Python-p-9781118290279`.

[4] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, second edition, March 1988. URL: `https://www.cs.princeton.edu/~bwk/cbook.html`.

Final Score	**Letter Grade**
$x \leq 50$	F
$50 < x \leq 53$	D-
$53 < x \leq 56$	D
$56 < x \leq 59$	D+
$59 < x \leq 62$	C-
$62 < x \leq 66$	C
$66 < x \leq 70$	C+
$70 < x \leq 75$	B-
$75 < x \leq 80$	B
$80 < x \leq 86$	B+
$86 < x \leq 93$	A-
$x > 93$	A

Table 1: Numerical Score to Letter Grade Conversion Policy