Introduction to Data Structure and Algorithms

Lecturer: Jalaj Upadhyay Syllabus

Course Description

The course intends to fill the gap in the education of those MITA students who have not been exposed to the basic notions of algorithms and data structures. While those students with a computer science degree have seen elementary versions of the topics covered in the course, others with a background in engineering, science, and business may not have seen these essential materials previously. The course covers, at a graduate level, the most important topics in data structures (arrays, list, stacks, queues, heaps, balanced binary search trees, hash tables, union-find trees. Graph representations). It also covers the most important algorithms (mergesort, quicksort, order statistics, Fast Fourier Transform, modular arithmetic, primality testing, algorithms for public-key cryptosystems, algorithms for blockchain, graphs and networks algorithms, and the notion of NP-completeness.) These notions are fundamental in understanding all other aspects of information technology, and even in some aspects of operations research.

Course Material

The primary book for the course would be *Introduction to Algorithms (4th edition)* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Other resources include:

- Algorithm Design by Jon Kleinberg and Eva Tardos
- Algorithms Illuminated by Tim Roughgarden

Learning Goals and Objectives

To learn and become fluent in the most fundamental algorithms and data structures, to determine which data structure and algorithm is suitable for which situation. In addition, in many job interviews, MITA students are quizzed about basic data structures and algorithms and are expected to know this material flawlessly. The course presents a rigorous training of such topics and prepares students for both interviews and for rational and efficient design and development of software.

Students who complete this course will demonstrate the following:

- Mastery and basic understanding of the notion of algorithm
- Mastery and basic understanding of the notion of data structure
- Recognizing how suitable data structures will aid faster and more efficient algorithms
- Knowing what is the most suitable method and data organization to use in software development
- The ability to precisely analyze the difficulty (complexity) of an algorithmic task

- The understanding of the notion of computational intractability and how to deal with computationally intractable problems in practice

Note that this is **not** an interview preparation course.

Tentative Course Syllabus and Schedule

This is a graduate-level course, so I expect you to do the required reading before the class to have a good engagement. Expect me to prepare properly for each class session. I expect the same of you. Complete all background reading and assignments. You cannot learn if you are not prepared. The minimum expectation is that you have prepared by studying for at least twice as many hours. The Chapters are all from the textbook, but these materials can also be found in various lecture notes. You can also refer to the lecture slides. Reach out to me or your TA if some part of the lecture is unclear. It is important that you keep up with the material covered in the class to ensure full success.

Topic 1: Basics and definitions.

- Simple and familiar algorithms such as minimum finding, selection-sort, insertion sort, merge-sort, sequential and binary search.
- The notions of asymptotic analysis: the concept of Big-O, Big-Ω, and Big-Θ, algebraic properties of Big Oh notation, natural lower bounds for algorithms
- Recursion, recursive algorithms, Solving simple tail recursion equations. Analysis of merge sort.

Reading: Chapters 1,2,3

Topic 2: Divide and Conquer

- More on recurrence relations and recursive algorithms: Computing performance of algorithms through recurrence relations, master theorem
- The notion of average-case analysis of algorithms, the notion of randomized algorithms, and the difference between the two paradigms. Case study the Quicksort algorithm and its analysis, Computing medians and other order statistics in expected linear time
- Other divide and conquer algorithms: Recursive methods for multiplying polynomials, Strassen's matrix multiplication algorithm

Reading: Chapters 4,5,7,9

Topic 3: Data Structure

- The notion of abstract data type, specification vs implementation, what can be done efficiently and what cannot be done efficiently with arrays, linked lists, stacks and queues
- Notion of directed and undirected graphs and their adjacency matrix, and adjacency list representation

- Depth-first search with stacks, breadth-first search with queues, testing connectivity, topological sorting.

Reading: Chapter 10 (sec. 10.1-10.3, skip 10.4 for now) and 22

Topic 4: Greedy Algorithms

- The notion of (unrooted) trees, minimum spanning trees and forests. Prim's and Kruskal's Greedy algorithms, the basic methods
- Introduction to priority queues and heaps as ADT and their array implementation, (*) heapsort and its analysis
- Use of heaps to improve Kruskal's algorithms
- Union-find problem and data structures, improving Prim's algorithm
- Huffman codes and greedy compression
- Dijkstra's algorithm, improvement with heaps, Floyd-Warshal algorithm, Shortest distances as a form of matrix multiplication

Reading: Chapter 6, 21 (skip 21.4), 23, and 16 (sec 16.3).

Topic 5: Dynamic programming

The main point of dynamic programming, when not to use recursion. Simple examples, computing binomial coefficients, Fibonacci numbers, etc.

Reading: Chapter 15 (skip 15.5)

Topic 6: More Advanced Data Structure

- Binary trees, the complexity of search, insert, delete operations and their relation to height and number of nodes,
- Representation of collections by balanced binary search trees, building, insertion, search, delete, merge. Implementation by red-black trees. Analysis of basic operations
- Hash tables, hash functions, and various hashing and addressing techniques and their (probabilistic) analysis
- (time-dependent) k-d trees, and range searching with applications to machine learning
- (time-dependent)B-trees and external search (searching data distributed across many fields.

Reading: Chapters 11, 12, 13, 18, 24, 25

Topic 7: Hardness and Cryptography

- Euclid's algorithm for gcd (greatest common divisor) of two numbers and its analysis
- modular arithmetic, and the notion of a group, repeated squaring algorithm in modular arithmetic
- The RSA public key system, signatures, and the assumption of difficulty of factoring integers, other public key systems based on the discrete logarithm
- Turing machines and undecidability of problems. Diagonalization.
- The precise definition of an algorithm: Boolean algebra, Circuits, implementing Turing machines as Boolean circuits.
- Decision problems, search problems, and optimization problems: Why focusing on decision algorithms is useful
- The first NP-complete problem: Satisfiability problem (SAT), and refinements: 3-SAT
- NP-completeness of CLIQUE, HAMILTONIAN-CYLCE, TSP, COLORING, SUBSET-SUM, KNAPSACK, INTEGER PROGRAMMING, etc

Reading: Chapters 31 and 34

Grading Scheme and General Information

- 4 Assignments: 10% each (in total 40% of the final grade)
- 1 Final exam: 40 % of the final grade
- 5-10 in the class quizzes: 10% of the final grade
- Scribe notes: 10% of the final grade: Sign-up sheet

There is no grade for attendance. The scribe notes can be seen as the proxy for attendance.

General Information about Assignments and Final Exams

Due date There would be *no* extension provided. You automatically have a 48 hours grace period in which you would not be penalized, but help from TA or instructor would not be provided in that 48 hours period. You can submit another version of your assignment during the grace period and only the last submitted assignment would be marked. So, say, if the assignment is due on Friday at 11:59 pm, you can submit new versions until Sunday, at 11:59 pm, but you would not get any help from the TA or the instructor from Friday, at 11:59 pm.

Late policy You will lose 5% of your marks on the assignment per day once the grace period is passed. So make sure to submit at least one version of your assignment before the grace period ends. If there are some extraneous reasons because of which you cannot submit the assignment, reach out to the instructor and TA at least 6 hours before the grace period ends. Sending a message does not automatically provide you an extension – you need to get written approval from the instructor or the TA. Once the 6-hour deadline is passed, no matter the situation, you will lose 5% per day for late assignments – the only exception here would be a medical reason with proper documentation provided by a medical practitioner. This should be university approved.

Collaboration You can collaborate with your fellow students, but your assignments should be written entirely by you. I encourage forming study groups to learn the class material collectively. However, ensure that you understand the material because that would be crucial for your performance in the final exam. The final exam would be based on the material covered in the class. Do not rely only on lecture slides as they are supposed to be a reference. Rely on your own notes and understanding. I encourage solving problems from the textbook and reaching out to me to discuss them.

Information About Scribe Notes

Scribe note is an alternative to attendance. For every class, two or three students will be the designated "scribe," taking careful notes during class, writing them up in the format described below, and sending them to me for posting here. Here is how to be a scribe:

Pick a date and your partners who would be writing the scribe notes together. The choice of week is first come first serve basis. Only one in each group needs to inform me, but send me a message on Canvas (cc to everyone in your group) so that I know the group and the week. It is your duty to pick a week and partners. We would like to have all the lectures scribed, so if there are some lectures missing and there are three people for two weeks, I will randomly assign two of those six people to the week not picked by anyone. An updated list for each week's scribe would be posted here. Take careful notes during the class. Ensure that your notes are written out in the document in complete well-formulated sentences that would be understandable to even those students who have missed the class. The notes should be written in your own words. Do not include any administrative information, but only the technical aspects of the lecture. Prepare the notes into a latex document. You can use Overleaf if you do not want to download any editor. I can provide you with some basic pointers to use Latex. Send via Canvas your scribe notes (with the source latex file and any supporting file you are using) by Thursday 11:59 pm. I will post them as an announcement. Other students would have the opportunity to comment on your scribe notes.

The grade for the scribe notes would also be based on the comments of other students. Your final grades for scribe notes would be divided equally between the quality of your scribed notes (5% which would be the same as your partners) and the quality of comments you made on other students' scribed notes (5%) so that they can improve. I will also provide you feedback on your scribe notes. The 5% of the grade defined by the quality of your scribed notes would be given based on how accessible your scribe notes are for anyone who missed the lecture. Make sure that it does not contain typos and grammatical errors.

The objective of this exercise is to ensure clarity of thought and your own understanding of the material. Do not plagiarize. If your scribed notes are plagiarized, you will be awarded no marks irrespective of your other activities.