CS 610 Spring 2023
Data Structures and Algorithms

Instructor: Dr. Ravi Varadarajan
Email: ravi.varadarajan@njit.edu

Course Goals

The aim of this course is to help students acquire skills in writing efficient programs using various data structures and algorithm design paradigms that they will learn during the course. They will also learn to analyze the time and space complexity of their programs using asymptotic bounds so as to assess how they scale with input sizes. They will learn efficient algorithms in popular problem domains such as sorting, searching, graphs, numerical algorithms and pattern matching.

Course learning outcomes

1. Learn asymptotic notations (big-oh, big-omega, big-theta, little-oh) to analyze the growth rate of time and space complexities of algorithms.
2. Learn to solve recurrence equations that arise in the time-complexity analysis.
3. Learn basic (Lists, Queues, Stacks) and advanced data structures (Heaps, Binary Search Trees, Hash tables, Graphs) and their use in the design of efficient algorithms for solving problems in various domains.
4. Learn different techniques to prove correctness and to analyze time complexity of algorithms using worst-case and average case inputs and also using amortization measures.
5. Learn different algorithm design paradigms (Divide-and-conquer, Greedy, Dynamic programming) for solving problems efficiently.
6. Learn efficient sorting and searching algorithms.
7. Learn efficient algorithms for solving various classical problems in numeric (matrix and polynomial arithmetic, FFTs) and non-numeric processing (pattern matching) areas.
8. Demonstrate knowledge acquired in this course regarding data structures and algorithm design through implementation of a programming project.

Course resources

Required text book:

Algorithm Design: Foundations, analysis, and internet examples by M.T. Goodrich and R.Tamassia, Addison Wesley, 2001 ISBN-0-471-38365-1.

Lecture slides which form the primary source of course material will be posted in Canvas after the corresponding lectures.

Other references:

Introduction to Algorithms, 3rd Edition, MIT Press, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, ISBN-13: 978-0262033848

Course Topics

1. Mathematical foundations (Supplementary materials & **Ch. 1**) – Asymptotic growth functions (big-oh, big-omega, big-theta, little-oh), proof techniques (induction, by contradiction etc.)

2. Algorithm analysis & Elementary data structures (**Ch. 1 & Ch. 2 except 2.5**) - computational model, pseudo-code, running time analysis, amortized complexity, Stacks, Queues, Lists, Binary Trees, Heaps
3. Algorithm design strategies (**Ch. 5**) – Greedy algorithms, Divide-and-conquer, Dynamic Programming
4. Searching (**Ch 2.5** & **Ch. 3**) – Dictionaries, Binary search trees, balanced search trees (Red-Black trees, 2-4 trees), Splay Trees, Skip lists
5. Sorting, selection and set algorithms (**Ch. 4**) – Bucket sort, radix sort, merge sort, quick sort and heap sort, order statistics, lower bound analysis, set Union-Find algorithms
6. Graph data structures and connectivity (**Ch. 6**) – Directed and undirected graph data structures, graph traversals (BFS, DFS), connectivity algorithms (spanning trees, bi-connectivity, strong connectivity)
7. Weighted graph algorithms (**Ch. 7, Ch. 8.1-8.2**) – Minimum spanning trees (Kruskal's and Prim's algorithms), Dijkstra's single-source shortest path algorithms, Floyd-Warshall algorithm for all-pair path problem using closed-semirings, Flow problems
8. Text processing (**Ch. 9.1-9.3**) – Pattern matching algorithms, Tries
9. Numerical processing (**Ch. 10.1 and 10.4, 10.5**) – Strassen's matrix multiplication algorithm, Modulo arithmetic and rings, Fast Fourier Transform, convolutions
10. Advanced topics as time permits (Possible topics include problem complexity, P vs NP, NP-Completeness, Quantum computing etc.)

There may be slight deviations of this syllabus or the schedule during the semester.

Grading Criteria
Written home works (22%), Pop quizzes (8%), 2 Midterm Exams (30%), Final Exam (25%), Programming project (15%).

There are no make-up quizzes or exams. Exceptions for missing homework/project deadlines or exams will be considered on a case-by-case basis provided you submit documentation for the reason to the Dean of Students Office who in turn will contact me.

Programming project has to be implemented in Java or Python. Details about the programming project will be revealed during the course.

Your final grade will be assessed relative to the rest of the class based on a curve. There will not be any rounding done when determining the final grade. Final grade change request will be considered only for correction of grading errors but not for any other reasons. Incomplete grade (I) will not be given except in rare circumstances when the student has been doing well in the course otherwise.

Academic honesty policy

All work that you submit as your own must, in fact be your own.
There should be no collaboration on the home works or the project.

Academic Integrity is the cornerstone of higher education and is central to the ideals of this course and the university. Cheating is strictly prohibited and devalues the degree that you are working on. As a member of the NJIT community, it is your responsibility to protect your educational investment by knowing and following the academic code of integrity policy that is found at:

http://www5.njit.edu/policies/sites/policies/files/academic-integrity-code.pdf.

Please note that it is my professional obligation and responsibility to report any academic misconduct to the Dean of Students Office. ***Any student found in violation of the code by cheating, plagiarizing or using any online software inappropriately will result in disciplinary action. This may include a failing grade of F, and/or suspension or dismissal from the university.*** *If you have any questions about the code of Academic Integrity, please contact the Dean of Students Office at dos@njit.edu"*