RUTGERS
Rutgers Business School
Newark and New Brunswick

Computer Science
COURSE NUMBER: 26:198:685
COURSE TITLE: Introduction to Algorithms and data structure

COURSE MATERIALS

Texts and References**:**

1. **(Required)** T. Corman, C. Leiserson, R. Rivest, and C. Stein *"Introduction to Algorithms", Third edition* MIT Press 2009.

2. **(Recommended)** R. Sedgewick, and K. Wayne *"Algorithms" Fourth edition* , Addison-Wesley, 2011.

The first book is our main text. The second covers more or less the same material, but is shorter and more readable (it is considered an undergraduate text).

CLASS ORGANIZATION & ADMINISTRATION

Required work

1. There will be 6 or 7 homework sets during the semester. They will constitute 50% of your final grade. Also **you will not pass this course if you fail to hand in two homework assignments or more regardless of your total score!**

2. We will have just a final exam. The final constitutes 50% of your grade

Computer Science **(26:198:685)**

COURSE SCHEDULE

Topics

Below is a list of topics to be covered in the course. A star (*) in front of the topic means it will be covered if there is enough time

Topic 1: Basics and definitions. (1)

- The notion of an algorithm and computational resources used to perform it (primarily time and space, but also amount of parallel computational resources will be considered.
- Simple and familiar algorithms such as sums and products of numbers, sums and products of polynomials, and sums and products of matrices as case studies
- The notion of asymptotic analysis, and the concept of Big O
- Use of counting techniques, recursion and summation methods to estimate performance of algorithms. Case study: computing minimum and maximum elements, simple sorting methods such as insertion and selection sorting
- **Reading:** Chapters 1,2,3

Topic 2: Divide and Conquer paradigm and recursion, randomized algorithms, and average case analysis (1)

- Recurrence relations and recursive algorithms: Recursive methods of multiplying polynomials, Strassen's matrix multiplication, Mergesort algorithm, computing performance of algorithms through recurrence relations
- 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. Faster algorithms for verifying a*b=c where a,b, and c are polynomials, or matrices
- Computing medians and other order statistics
- **Readings:** Chapters 4,5,7,9

Topic 3: Other topics involving numbers, polynomials, and matrices (2)

- *Matrix computations, Solving systems of equations, LUP factorization, determinants, permanents (***Read chapter 27)**
- The Fast Fourier Transform, its inverse and its analysis. Applications to polynomial computations. (**Read chapter 30)**

- Cryptography, public key systems, RSA cryptography, primality testing **(Read chapter 31 sections 31.6,31.7, 31.8, 31.9)**

Topic 4: Information storage and retrieval, Searching, and supporting data structures (3)

- Elementary data structures such as array, linked list, stacks and queues.
- Hash tables, hash functions, and various hashing and addressing techniques and their (probabilistic) analysis
- Binary trees, relation to height and number of nodes, priority queues, heaps, Heapsort
- Representation of collections by balanced binary search trees, building, insertion, search, delete, merge. Implementation by red-black trees. Analysis of basic operations
- *B-trees and external search (searching data distributed across many fields.
- **Reading: Chapters 10, 11, 12, 13, 18**

Topic 5: Algorithms and data structures for networks and graphs and networks (3)

- Definition of directed and undirected graphs, adjacency, and incidence matrix representation, depth-first and breadth-first search, topological sorting, strong and weak connectedness, and components,
- The “greedy paradigm” minimum spanning trees and Prim's and Kruskal's algorithms. Other greedy methods (and inadequacy of the method in problems like maximum clique and independent sets, matching, traveling salesman, coloring)
- Shortest paths and the the dynamic programming paradigm. Dijkstra's shortest path algoithm, Floyd-Warshall algorithm, the idea of dynamic programming, application of dynamic programming to other problems (knapsack, TSP, matrix chain product)
- Maximum flow problem, Max-flow-min-cut theorem, Ford and Fulkerson Augmenting path algorithm, Complexity analysis
- **Reading: Chapters 22, 23,24,25, 26***

Topic 6: *String and text algorithms, pattern search, data compression (2)

- *Knuth-Morris-Pratt algorithm, Rabin-Karp algorithm, Boyer-More algorithm for pattern matching

- *Finite automata and regular expressions
- *Trie data structures
- *Huffman trees and data compression
- ***Reading: chapter 32, and additional reading material**

Topic 7: NP-completeness and Computational intractability (2)

- The precise definition of an algorithm: Turing machines
- Decision problems, optimization problems, and functional problems: Why focusing on decision algorithms is useful
- The first NP-complete problem: Satisfiablity problem (SAT), and refinements: 3-SAT
- NP-completeness of CLIQUE, TSP, COLORING, SUBSETSUM, KNAPSACK, INTEGER PROGRAMMING
- **Reading: chapter 34**