BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, Pilani
Pilani Campus
AUGS/ AGSR Division

SECOND SEMESTER 2020-21
COURSE HANDOUT

Date: 12.01.2021

In addition to part-I (General Handout for all courses appended to the time table) this portion gives further specific details regarding the course

Course No. : **CS F211**
Course Title : **Data Structures & Algorithms**

Lecture Instructors: VISHAL GUPTA (Email: vishalgupta@pilani.bits-pilani.ac.in)
L. Rajya Lakshmi (Email: rajya.lakshmi@pilani.bits-pilani.ac.in)

Lab Instructor(s): Vishal Gupta (Email: vishalgupta@pilani.bits-pilani.ac.in)
Rajya Lakshmi (Email: rajya.lakshmi@pilani.bits-pilani.ac.in)
Oshin Rawlley (RS) (Email: p20200063@pilani.bits-pilani.ac.in)
Ankit Agarwal (RS) (Email: p20190021@pilani.bits-pilani.ac.in)
Sameen Ahmad (RS) (Email: p20190412@pilani.bits-pilani.ac.in)

Course Website: ***access via*** **http://nalanda.bits-pilani.ac.in**

1. a. Course Overview:

This course offers an introduction to typical data structures used for representing collections of data and the associated relations. The course has a design focus but realization and performance issues will also be emphasized.

1. b. Course Objectives:

The objectives of the course are to introduce students to:

- common structures for storing collections of data and associated relations along with algorithms for retrieving/modifying such collections
- techniques for designing and implementing such ***data structures*** on modern computers
- formal and experimental techniques for analyzing the performance (time and space requirements) of such data structures.

1.c. Course Scope

The course will cover most common deterministic data structures for linear and non-linear data as well as a few randomized data structures. Implementation techniques will be covered for sequential execution with virtual memory.

2. Text & Reference:

a.Text Book:

T1. Michael T. Goodrich and Roberto Tamassia: *Algorithm Design: Foundations, Analysis and Internet examples* (John Wiley &Sons, Inc., 2002).

b. Reference Book:

R1. Cormen T.H., Leiserson, C.E., Rivest, R.L., and C. Stein. *Introduction to Algorithms, MIT Press,* 3rd Edition, 2009. (*Indian reprint: Prentice-Hall).*

AR: [*See course website for additional material*]

3. Course Plan:

3. a. Lectures – Themes, Topics, and Prior Knowledge:

#	Theme	Description of Topics	Prior Knowledge
I	Introduction	Data Abstraction, Data Modeling, Data Representation	Programming with arrays and records (structures in C)
II	Linear Structures	Lists (static and dynamic), Random vs. Sequential Access, Restricted Access Lists.	Programming with arrays, records, and pointers. Basic understanding of dynamic allocation and linked lists.
III	Dictionaries –Searching and Ranking.	Sorting Algorithms, Searching, Hashing and Hashtables, Bloom Filters.	Basic Probability
IV	Implementation Issues	Dynamic Allocation, Recursive and Iterative implementation of repetition, Recursive data definitions.	Basic understanding of dynamic allocation and linked lists. Basic understanding of recursive procedures.
V	Performance Analysis	Complexity model for algorithms, Complexity Analysis, Performance Measurements and Model, Impact of virtual memory on performance.	Counting techniques, Recurrence relations and techniques for solving them, Order notation (O, Θ, and Ω), growth rates of typical functions (sub-linear, linear, polynomial, exponential)
VI	Non-Linear Data Structures (Trees)	Modeling with trees, Binary Trees and Tree Traversals, Binary Search Trees and Height Balanced Search Trees, General Trees and Tree Traversals, Heaps and Tries, Applications of Trees, External memory data structures.	Basic understanding of dynamic allocation and pointers. Basic understanding of recursive procedures.
VII	Non-Linear Data Structures (Graphs)	Modeling with Graphs, Graph Representations, Basic Graph Traversals, Basic algorithms on Graphs.	Basic Graph Theory and Graph Representations (Adjacency Matrix and Adjacency Lists). Basic understanding of dynamic allocation and pointers.

3.b. Laboratory – Themes and Exercise Topics

Module	Theme	Description
P-I	Basic Tools and Techniques	Dynamic Allocation, Pointers, and Linked Lists; Command Line arguments, Compilation options and commands, Separate Compilation, and Libraries. Profiling, Measuring Time and Space usage.
P-II	Sorting and Hashtables	Pragmatics of Sorting and efficient implementations, Hashing techniques and Hashtable implementations.

P-III	Trees	Binary Search Trees and Height Balancing; General Trees and Traversals; Applications; Tries and Heaps.
P-IV	Graphs	Basic representations of graphs and traversals; implementation of simple graph algorithms.

Prior Knowledge: ***Basic but rigorous programming using* C**

3.c. Lecture Schedule:

Lec.	Theme	Topic	Learning Outcome(s) [*The student should be able to*:]	Reading
1 to 2	I	Course Motivation and Introduction. Data: Modeling, Abstraction, and Representation.	• understand the roles of modeling data, abstraction, and representation in solving problems. • model given data, formulate an abstraction, and choose a representation	-
2 to 3	II, IV	Abstract Data Types and Data Structures: Lists (LIFO, FIFO, Ordered) and Implementation	• define and implement access restricted lists (Stacks and Queues) • implement ordered lists using arrays and linked lists • make implementation choices between: arrays and linked lists; sorted and unsorted lists.	T1 2.1
4	V	Time and Space Requirements. I/O Performance.	• understand time complexity measures • analyze simple algorithms and data structures for time requirements	T1 1.1 to 1.4
5 to 7	III	Divide-and-Conquer Algorithms – Introduction, Insertion Sort and Merge Sort	• apply the principle of divide-and-conquer on simple problems • formulate time complexity of an algorithm as a recurrence relation and solve it • understand and compare basic how sorting algorithms (Insertion Sort and Merge Sort) work and perform	T1 4.1 and T1 5.2 and R1 4.3 to 4.5
			•	
9 to 11	IV	Recursion and Iteration. Transforming recursive procedures to iterative form.	• understand how recursive procedures run internally • define data recursively and write procedures accordingly • transform typical recursive procedures to iterative form • eliminate tail recursion • control stack space in converted (iterative) forms	Class Notes
12 to 15	III, V	Quick Sort – Algorithm, Analysis, Pivot Selection and Randomization, Performance Improvements.	• understand how QuickSort works and the significance of pivot selection on its performance • understand how randomization improves “expected performance” • analyze special cases of QuickSort and explain corresponding performance improvements including stack space reduction • implement an efficient version of QuickSort	T1 4.3 and R1 Ch. 7

16a	V	Lower Bound on Comparison-based Sorting	• argue that comparison based sorting algorithms cannot be designed below a certain complexity	T1 4.4
16b to 17	III	Bin Sort / Bucket Sort, and Radix Sort	• explain how distribution-based sorting algorithms work • implement Bucket Sort, Radix Sort, and their variations.	T1 4.5
18 to 21	III, V	Hashing and Hashtables – Unordered Dictionary, Hash functions and Collision, Separate Chaining, Open-Addressed Hashtables and Probing Techniques, Analyses.	• explain the significance of hash functions and hash tables in providing efficient lookup • implement a few different hash functions and analyze their performance • implement separately-chained hashtables and analyze their performance • implement open-addressed hashtables with different probing techniques and analyze their performance	T1 2.5 and R1 11.2 to 11.4
22	III, V	Bloom Filters – Motivation, Design and Analysis.	• understand the need for a probabilistic data structure • understand and leverage the trade-offs in using a Bloom filter • implement Bloom filters	Addl. Reading
23 to 24	VI	Partially Ordered Data – Modeling using Trees, Binary Trees, Tree Traversals.	• understand how to model partially ordered data • implement binary trees and traversals on binary trees	T1 2.3
25 to 26	V, VI	Ordered Dictionaries: Binary Search Trees – Operations and Analysis, Height Balanced BSTs.	• implement binary search trees • understand the time complexity issue • design and implement a height balancing technique	T1 3.1 to 3.2 and T1 3.5
27 to 28	V, VI	Generalized Trees and Tree Traversals.	• understand how to model data using trees • design and implement trees with fixed and arbitrary branching • implement traversal techniques on trees and use traversals in applications	Class Notes
29 to 30	VI	Ordered Dictionaries: Tries, Analysis, Variants.	• Design and implement Tries and its variants	T1 9.2
31 to 32	VI	Partially Ordered Data – Heaps, Analysis, Applications.	• implement and use heaps	T1 2.4
33	VII	Modeling Binary Relations using Graphs. Graph Representations.	• model binary relations using graphs • understand and choose graph representations for specific problems • implement graph operations using a chosen representation	T1 6.1 to 6.2
34 to 36	V, VII	Graphs – Traversal, Connectivity and Connected Components	• understand and implement algorithms for graph traversal • understand and implement algorithms for testing connectivity and finding connected components	T1 6.3 to 6.4

			• model problems as traversal / connectivity problems and use graph algorithms to solve problems	
37 to 40	V, VII	Weighted Graphs – Modeling, Shortest Paths, and Minimal Spanning Trees. Advanced Algorithms	• understand how to model data using weighted graphs • understand and implement shortest path algorithms (single source and all pairs) • understand and implement an MST algorithm • model problems as path problems and use graph algorithms to solve problems	

Note: *See course website for Class Notes and Additional Reading.* End of Note.

3. d. Pedagogy

Pedagogy

- **Lectures:**
  - Classes are designed to be organic – interaction is essential
  - Socratic approach will be followed i.e. there will be questions raised in class to provoke the students' thinking and derive the answers therefrom.

- **Labs:**
  - Lectures and Labs are tightly woven i.e.
    - it is important to pay attention to lectures to be able to implement the exercises in the lab and
    - it is important to carry out the lab exercises to completely understand lecture material and occasionally for following subsequent lectures.
  - Focus in the labs would be in getting sound and complete implementations but also in learning to analyze performance:
    - Performance analysis is key to this course and
    - theoretical/mathematical approach to performance analysis would be taught in lectures but
    - practical approach must be understood by doing it in labs – profiling programs, measuring time and space usage, plotting and fitting curves, and tuning your program to improve performance are essential tools in an implementer's survival kit!
- **Evaluation:**
  - Evaluation components will assess a range of learning outcomes: basic understanding of the material taught, lab skills learnt, ability to apply what is learnt in lectures and labs, ability to solve new problems by combining several components of content and skills learnt and design skills.

4. Evaluation

4.a. Evaluation Scheme:

Component	Marks [300]	Date & Time	Duration	Remarks
Lab Performance	21 M (7%)	For each Lab Session	3 days each	**Each lab performance will be graded for 3M marks. Students will upload the solution of the lab sheet within three days of the conduct of the lab session.**
Quiz	60 M (20%)	TBA	40 Minutes	Open Book
Mid-Term Test (Written)	93 M (31%)	As per AUGSD	As per AUGSD	Open Book
Comprehensive Exam	126 M (42%)	As per AUGSD	As per AUGSD	Open Book
TOTAL	**300M**			

Note: *As per the date on which this handout is prepared, the whole semester is expected to be online because of COVID19 pandemic. If you will be in campus in between the semester, the components and its type/mode might change. It such case, it would be communicated to you in advance.*

4. b. Make-up Policy:

- **Laboratory Performance:**
    1. **Marks in best 7 labs (out of 8 evaluative ones) will be taken up in final grading. No explicit MakeUP for any lab.**
    2. **All the solutions uploaded will undergo through plagiarism testing using MOSS/JPLAG.**
- **Quiz and Mid Sem:**
    1. Make-up will be ***granted only for genuine reasons*** when the student is physically unable to appear for the quiz/test.
    2. It is the responsibility of the student to communicate a make-up request (along with necessary documentary proof) to the course IC before or during the test/quiz.
    3. Decision of the instructor-in-charge with respect to a make-up request is final.
    4. Make-up of this component ***can be*** conducted using viva-voce examination.
- **Comprehensive Exam:**
    1. Permission for a Make-up for the comprehensive exam will have to be obtained from **Associate Dean, AUGSD** and
    2. **Make-up for the comprehensive exam will usually be scheduled centrally**.

4.c. Fairness Policy:

Any use of unfair means in lab tests, Quiz, mid-term test, or comprehensive exam will be handled strictly. The minimum penalty would be loss of full weight of the component. Students involved in such activity are liable for further sanctions including being formally reported to the Unfair Means committee and being subject to penalties enabled by Unfair Means Rules of the Institute. Unfair means would include copying from or enabling copying by other students; or copying / borrowing material from sources of information not permitted during the tests / exams.

.

6. Notices: All notices concerning this course will be displayed on the course website (on Nalanda) only. If there is a need email would be used on short notice (12 hours) – only BITS Pilani mail id of students would be used.

Instructor -In- Charge
CS F211