Birla Institute of Technology and Science, Pilani, Hyderabad Campus

Department of Computer Sc. and Information Systems

First Semester 2020-2021

CS F211 (Data Structures and Algorithms)

Date: 17th Aug 2020

Course Number	: CS F211 (L:3, P:1, U:4) T, Th, S: 5th hour
Course Title	: Data Structures and Algorithms
Instructor-In-Charge	: Prof. CHITTARANJAN HOTA (hota@hyderabad.bits-pilani.ac.in)
Instructors	: Dr. Mrityunjay Singh, Mr. K Sai Anirudh, Ms. Rashmi Sahay

Scope and Objectives of the Course:

A data structure is a collection of large amounts of data values, the relationships among them, and the functions or operations that can be applied on them. In order to be effective, data has to be organized in a manner that adds to the effectiveness of an algorithm, and data structures such as stacks, queues, linked lists, heaps, trees, and graphs provide different capabilities to organize and manage large amounts of data. While developing a program or an application, many developers find themselves more interested in the type of algorithm used rather than the type of data structure implemented. However, the choice of data structure used for a particular algorithm is always of paramount importance. For example, B-trees have unique abilities to organize indexes and hence are well suited for implementation of databases; Linked lists are well suited for backtracking algorithms like, accessing previous and next pages in a web browser; Tries are well suited for implementing approximate matching algorithms like, spell checking software or predicting text in dictionary lookups on Mobile phones; Graphs are well suited for path optimization algorithms (like in Google maps) or searching in a Social graph (like Facebook). As computers have become faster and faster, the problems they must solve have become larger and more complex, requiring development of more complex programs. This course will also teach students good programming and algorithm analysis skills so that they can develop such programs with a greater degree of efficiency.

The primary objectives of the course are as under:

- Apply various basic data structures such as stacks, queues, linked lists, trees etc. to solve complex programming problems. Understand basic techniques of algorithm analysis.
- Design and implement advanced data structures like graphs, balanced search trees, hash tables, priority queues etc. Apply graph and string algorithms to solve real world problems like finding shortest paths on huge maps or detecting plagiarism percentage.
- Apply basic algorithmic techniques such as brute-force, greedy algorithms, divide and conquer, dynamic programming etc. to solve complex programming problems and examine their efficiency.

At the end of the course, you should understand common data structures and algorithms, be able to develop new data abstractions (interfaces) and use existing library components in C++.

Text Book:

T1: Introduction to Algorithms, TH Cormen, CE Leiserson, RL Rivest, C Stein, 3rd Ed., MIT Press, PHI, 2010.

Reference Books:

R1:Data Structures and Algorithms in C++, Michael T. Goodrich, Roberto Tamassia, David M. Mount, 2nd Edition, 2011, Wiley (e-book in India).

R2:Data Structures & Algorithm Analysis in C++, Mark Allen Weiss, 4th Edition, Pearson, 2014.

R3:Data Structures and Algorithms, Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, 1st Ed., Pearson, 1983.

Lecture Plan:

Lecture#	Learning Objectives	Topics	Reference
1	The role of DS and Algorithms in Computing.	What kinds of problems are solved by algorithms? Journey from problems to programs.	T1 (1), R3(1)
2	Introduction to C++.	Classes: Class Structure, Constructors, Class Friends and Class Members, Standard Template Library (STL), An example C++ program.	R1 (1.5, 1.6)
3-4	To understand the features of Object Oriented Paradigm.	Object Oriented Design: Goals, Principles and Design Patterns; Inheritance and Polymorphism; Interfaces and abstract classes; Templates.	R1 (2.1, 2.2, 2.3)
5-7	Implementing elementary data structures and algorithms. Understanding techniques for Algorithm analysis.	Using arrays, Insertion and removal from a Linked list, generic single linked list, doubly linked lists, circular linked lists, linear and binary recursion.	T1 (10), R1 (3.1, 3.2, 3.3, 3.5)
8-9		Functions: Linear, N-Log-N, Quadratic functions etc., Asymptotic notation and asymptotic analysis, Using Big-Oh notation, Examples of analysis.	T1 (2), T1(3) R1 (4.1, 4.2)
10-12	Implementing more common data structures and algorithms like Stacks, Queues, Deques, Vectors, List ADTs, Sequences, and Trees. Using Amortization to perform a set of push operations on a vector.	Stack ADT, Array-based stack implementation, stack implementation using generic linked list, Applications of stacks: matching tags in an HTML document; Queue ADT, Array-based and circular linked list based implementation.	T1(10), R1 (5.1, 5.2)
13		Double-Ended queue: Deque ADT, Implementing using doubly linked lists, Adapters: Implementing stack using Deque.	T1(10), R1 (5.3)
14		Vector ADT, Simple Array-based implementation; Extendable array based implementation (Amortization) and STL Vectors.	R1 (6.1)
15-16		List ADT: Node based operations and Iterators, doubly linked list implementation, Sequence ADT, Applications: Bubble sort on sequences, and its analysis.	T1(10), R1 (6.2, 6.3, 6.4)
17-18		General Trees: Properties and functions, Traversal algorithms: Pre order, post order traversals, Binary tree: ADTs, Linked and Vector structures for Binary trees, Binary tree traversal, Template function pattern.	T1(10), R1 (7.1, 7.2, 7.3)
19-21	Implementing Advanced data structures like Priority queues, Heaps, Hash tables, Maps, Skip lists, Dictionaries, Search Trees.	Priority Queue ADT, Implementing using Lists, Algorithms suitable for Priority queues, Heap: Complete binary trees and their representation, Implementing Heaps using Priority queue, Heap sort as an example.	T1(6), R1 (8.1, 8.2, 8.3)
22-24		Map ADT, Implementation using Lists, Hash tables: Bucket arrays, hash functions, compression functions, collision-handling schemes, Rehashing into a new table, Implementation of hash tables, Skip lists: Search and update operation implementations.	T1(11), R1 (9.1, 9.2, 9.4)
25		Dictionary ADT: Implementation with location-aware entries.	R1 (9.5)
26-28		Binary Search Trees: Operations and Analysis, AVL Trees: Insertion and deletion, Analysis, Multi-way search trees, Red-Black Trees: Operations and analysis.	T1(12),T1(13) R1 (10.1, 10.2, 10.4, 10.5)
29-30	Understanding various basic	Merge sort: Divide and conquer, merging arrays and lists, running time of merge sort; Quick sort: Randomized quick sort.	T1(4), T1(5) R1 (11.1, 11.2)

31-32	Algorithmic techniques and usage of appropriate data structures along with their applications and analysis.	Sorting through algorithmic lens: Lower bound, Linear time: Bucket and Radix sort, Comparing sorting algorithms.	T1(6), T1(7), T1(8), R1 (11.2, 11.3)
33		Sets: Set ADT, Mergable sets, Partitions; Selection: Prune-and-Search, randomized quick-select.	T1 (7), R1 (11.4, 11.5)
34-35		Strings and Dynamic programming: String operations, Matrix Chain-Product as an example, Applying Dynamic programming to LCS problems.	T1 (15), R1 (12.1, 12.2)
36		Pattern matching algorithms: Brute force, Boyer-Moore algorithm, KMP algorithm.	R1 (12.3)
37		Text Compression and Greedy method: Huffman-Coding algorithm, greedy method; alternatively doing pattern matching using Tries: Standard and Suffix tries.	T1 (16), R1 (12.4, 12.5)
38		Graph Algorithms: Graph ADT, Data structures for graphs: Edge list, Adjacency list, Adjacency matrix.	T1(22), R1 (13.1, 13.2)
39-40		Graph Traversals: DFS, and BFS, Traversing a Diagraph, Transitive closure.	T1 (22), R1 (13.4)
41-42		Shortest path and MST: Dijkstra, Kruskal, and Prim-Jarnik algorithms.	T1(23),T1(24) R1 (13.5, 13.6)

Evaluations:

Component	Duration	Weightage(%)	Date & Time	Nature of the component
Test1 (Quiz)	0.5 hrs.	15%		Open Book
Test2 (Quiz)	0.5 hrs.	15%		Open Book
Test3 (Quiz)	0.5 hrs.	10%		Open Book
Lab evaluations (Two quizzes)	0.5 hrs./quiz	15%	-	Open Book
One group project & demo	0.5 hrs./group	15%	-	Open Book
Comprehensive examination	3 hrs.	30%		Open Book

List of Laboratories: (Every lab sheet will have a detailed C++ program and a to-do-task)

Lab1: Demonstartion of C++ Class structures, Class members, and Standard Template Library (STL) in C++.

Lab2: Demonstration of C++ Inheritance, Interfaces, Abstract classes, Function templates and Class templates.

Lab3: Inserting (at the front) and deleting (from the end) an object from a Singly Linked List; Implementing a doubly linked list with header, and trailer sentinels. Example read of words from a file, discarding any strings that do not look like words using Linked Lists.

Lab4: Implementing linear recursion (reversing an array); Implementing Binary recursion (computing K-th Fibonacci's Number).

Lab5: Write a program to estimate the number of primitive operations executed by a sample C++ program (like assigning a value to a variable, performing an arithmetic operation, indexing into an array

etc.) upto a constant factor. Compare the running times of two algorithms, one implemented as a Quadratic-time algorithm and the other as a Linear-time algorithm for computing prefix averages of a sequence of numbers.

Lab6: Create an implementations of the Stack ADT based on a singly linked list representation; Use templates to produce a generic stack data structure, Create a program that evaluates arithmetic expressions in postfix form; Write a C++ program to match tags in an HTML document using the stack.

Lab7: Implement insert and remove operations in a double ended queue (deque) using a singly linked list; Implemtation of Iterator-based constructors to create a Vector, sort a portion of this vector, permute the elements and insert an element into it.

Lab8: Implement a List ADT using a doubly linked list. Implement an anagram puzzle program using the List ADT implemented in the first part.

Lab9: Write a C++ program to find out the depth and height of a node in a tree; Implement reading an arithmetic expression in prefix form into a binary tree of characters, and then output in infix or postfix using an inorder or postorder traversal.

Lab10: Create an implementation of the Heap ADT using an array representation of a tree, Use inheritance to derive a priority queue class from your heap class and develop a simulation of an operating system's task scheduler using a priority queue; Create a heap sort function based on the heap construction techniques used in your implementation of the Heap ADT.

Lab11: Implement the Hash Table ADT using an array of lists representation; Implement a perfect hash to store selected C++ reserved words.

Lab12: Implement Merge-sort using a vector based merge-sort algorithm; and write a C++ program to implement In-place Quicksort assuming distinct elements are there for sorting

Lab13: Create an implementation of the Weighted Graph ADT using a vertex list and an adjacency matrix, Develop a routine that finds the least cost (or shortest) path between each pair of vertices in a graph, Add vertex coloring and implement a function that checks whether a graph has a proper coloring or not.

Make-up-Policy:
Make-up exams will be strictly granted on prior permission and on genuine grounds only. A request email should reach the I/C on or before the test.

Course Notices:
Course material pertaining to this course will be made available on a regular basis on the course webpage (http://csis.bits-hyderabad.in/csf211) and googleclass page will be used for notices, announcements, grades, quizzes, and googlemeet recordings. Project demos will be taken on the machines owned by students.

Consultation Hour: Monday (5pm to 6pm).

Academic Honesty and Integrity Policy:
Academic honesty and integrity are to be maintained by all the students throughout the semester and no type of academic dishonesty is acceptable.