KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
Deemed to be University
BHUBANESWAR-751024

School of Computer Engineering
Autumn Semester 2018-19

Course Handout

Date: 26/06/2018

Course code	**: CS 2001**
Base-lined version	**: 1.0**
Base-lined date	**: 04/07/2018**
Course Title	**: Data Structure and Algorithm**
Course Coordinator	**: Mr. Rajat Kumar Behera**
Course Faculty	**: Dr. Alok Kumar Jagadev**
Course offered to the School	**: Computer Engineering, Electronic Engineering and Electrical Engineering**

1. Course Description:

Introduction: The course focuses on basic and essential topics in data structures and algorithms, including arrays, linked lists, Stacks, Queues, Trees, sorting algorithms, searching algorithms and graphs.

Course Contents:

Sr#	Major Area	Detailed Area
1	Introduction	Structures and Unions, Pointers, Dynamic Memory Allocation, Algorithm Specification, Space and Time Complexity
2	Arrays	Arrays, Abstract Data Type, Dynamically Allocated Arrays, Polynomials, Two-dimensional Array, Address Calculation, Matrix Addition and Multiplication, Sparse Matrix, Upper & Lower Triangular Matrix, Tridiagonal Matrix
3	Linked List	Singly Linked Lists and Chains, Representing Chains in C, Polynomials, Sparse Matrix, Doubly Linked Lists, Circular & Header Linked lists
4	Stacks and Queues	Stacks, Stacks using Dynamic Arrays and Linked List, Queues, Queue using Dynamic Arrays and Linked List, Circular Queues using Dynamic Arrays and Linked List, Evaluation of Expressions, Priority Queue, Dequeue
5	Trees	Introduction, Binary Trees, Binary Tree Traversals, Threaded Binary Trees, Binary Search Trees, AVL Trees, m-way Search Trees, B-Trees, Introduction to B+-Trees, Tree Operation, Forests
6	Graphs	Graph ADT, Graph Operation – DFS, BFS
7	Sorting	Insertion Sort, Quick Sort, Merge Sort, Heap Sort, Bubble Sort, Selection Sort, Radix Sort
8	Searching	Linear Search, Binary Search, Hashing – Hash Function, Collision Resolution Techniques

2. Course Objective:

- Understand different ways to represent different kinds of data.
- Learn different kinds of operation performed on different representation of data.
- Identify and apply the appropriate data structure and algorithm for a specified application.
- Analyse the performance of algorithm for different operation performed on data.
- Provide solid foundations in foundational aspects of programming – both data structures and algorithms
- Demonstrate the correctness of the algorithm by analysing their computational complexities

3. Course Outcome:

CO #	Detail
CO1	Students will be able to understand the concepts of data structure, data type and array data structure
CO2	Students will be able to analyse algorithms and determine their time complexity
CO3	Students will be able to implement linked data structure to solve various problems
CO4	Students will be able to understand and apply various data structures such as stacks, queues, trees and graphs to solve various computing problems
CO5	Students will be able to implement and apply standard algorithms for searching and sorting
CO6	Students will be able eeffectively choose the data structure that efficiently models the information in a problem.

4. Text Book:

T1. Data Structures using C by Aaron M. Tenenbaum, Yedidyah Langsam, Moshe J. Augenstein. Pearson, 1st Edition

5. Reference Books:

RB1. Data Structures, Schaum's OutLines, Seymour Lipschutz, TATA McGRAW HILL
RB2. Data Structures Using C, Second Edition, Reema Thereja, Oxford University Press
RB3. Fundamentals of Data Structures in C, 2nd edition, Horowitz, Sahani, Anderson-Freed, Universities Press.
RB4. Data Structures A Pseudocode Approach with C, 2nd Edition, Richard F. Gilberg, Behrouz A. Forouzan, CENGAGE Learning, India Edition
RB5. Data Structures and Algorithm Analysis in C, Mark Allen Weiss, Pearson Education, 2nd Edition.

6. Reference Site:

RS1. NPTEL - https://onlinecourses.nptel.ac.in/explorer
RS2. Tutorials Point - https://www.tutorialspoint.com/data_structures_algorithms/
RS3. Geeks for geeks - http://www.geeksforgeeks.org/

7. Pre-requisites:
- Programming in C (CS-1001)
- Mathematics for Computer Science

8. Course Lesson Plan:

Course Lecture No.	Unit	Topics	Day #	Refer to Chapter, See (Book)
1-4	Introduction	• Introduction • Course Coverage	1	Chapter 1 (T1) **Introduction to Data Structure**
		• Structure, Union • Pointers and Dynamic Memory Allocation	2	
		• Algorithm Specification • Algorithm Analysis	3	

Course Lecture No.	Unit	Topics	Day #	Refer to Chapter, See (Book)
		• Time Complexity • Space Complexity • Class Work	4	
5-10	Arrays	• Array Introduction • Row major order and address calculation	5	Chapter 1 (T1) **Introduction to Data Structure**
		• DMA (Dynamic Memory Allocation) • Difference between static and dynamic memory allocation • DMA – 1-D and 2-D arrays • Problem Solving • Pointers to array, Pointer to structure, Array of pointers	6	
		• Array abstract data type (ADT) • Problem Solving	7	
		• Polynomial & its Operation • Matrix Operation	8	
		• Sparse Matrix and its Operation • Class Work	9	
		• Contingency	10	
11-18	Linked List	• Introduction to Linked List • Advantages, Disadvantages, Application • Representation	11	Chapter 4 (T1) **Queues and Lists**
		• Class Work • Types of Linked List	12	
		• Double Linked List • Circular Linked List	13	
		• Linked List Operation – Insertion, Deletion, Insert Last, Delete Last • Class Work	14	
		• Linked List Operation – Insert After, Traverse, Search, Traverse Backward • Class Work	15	
		• Header Linked List & operation • Circular Header Linked List & operation	16	
		• Polynomial • Double Linked List & operation	17	
		• Sparse Matrix • Class Work	18	
19-26	Stacks & Queues	• Introduction to Stack • Stack Application • Stack Representation – Arrays	19	Chapter 2, 4 (T1) **The Stack, Queues and Lists**
		• Stack Representation – Linked List • Stack ADT	20	
		• Arithmetic Expression Evaluation • Class Work	21	
		• Arithmetic Expression Conversion • Class Work	22	
		• Introduction to Queues • Queues Application • Queues Representation – Arrays	23	

Course Lecture No.	**Unit**	**Topics**	**Day #**	**Refer to Chapter, See (Book)**
		• Queues Representation – Linked List • Queues ADT • Class Work	24	
		• Linear Queue Drawback • Circular Queues	25	
		• Deques • Priority Queue • Class Work	26	
27-38	Trees	• Introduction to Trees • Trees Terminology • Class Work	27	Chapter 5 (T1) **Trees**
		• Tress Application • Binary Tree – Full, Complete and Extended Binary Trees • Expression Trees • Class Work	28	
		• Representation of Binary Tree – Linked and Array Representation • Binary Tree ADT	29	
		• Arithmetic Expression Conversion • Class Work	30	
		• Binary Tree Traversal Concept and Algorithm – In-Order, Pre-Order and Post-Order & Level-Order • Binary Tree Construction with different traversal	31	
		• Class Work on Binary Tree • Threaded Binary Tree – Single and Double Threaded	32	
		• Binary Search Tree • BST ADT – Search, insertion	33	
		• BST ADT – Deletion, • Class Work	34	
		• Balanced Binary Tree • AVL Tree • AVL Rotation Techniques, ADT	35	
		• Multi-way Search Tree & ADT • B-Tree & ADT	36	
		• B+ Tree Introduction • Forest	37	
		• Class Work • Contingency	38	
39-42	Graphs	• Introduction to Graph • Graph Terminology • Graph Application	39	Chapter 8 (T1) **Graphs and Their Application**
		• Graph Representation • Class Work	40	
		• Graph Operation – DFS and BFS • Class Work	41	
		• Class Work • Contingency	42	
43-46	Sorting	• Bubble Sort • Insertion Sort • Selection Sort	43	Chapter 6 (T1) **Sorting**

Course Lecture No.	Unit	Topics	Day #	Refer to Chapter, See (Book)
		• Quick Sort • Merge Sort	44	
		• Heap Sort • Radix Sort	45	
		• Class Work • Contingency	46	
47-49	Searching	• Linear Search • Binary Search	47	Chapter 7 (T1) **Searching**
		• Hashing – Hash Function • Class Work	48	
		• Hashing – Collision Resolution Technique • Class Work	49	

9. Evaluation Scheme:

ES No.	Evaluation Component	Duration	Percentage of Evaluation	Date	Course Lecture No.		Mode
					From	To	
1	Mid-Semester Examination	1.5 Hrs	20	TBD	1	22	Closed Book
2	Activity based Teaching and Learning	Through out semester	30	Through out semester	NA	NA	Open Book, Closed Book and Presentation
3	End-Semester Examination	3 Hrs	50	TBD	1	49	Closed Book

Mid-semester question paper comprises of 6 questions and students have to answer any four questions including question no 1, which is compulsory. Weightage for each question is 5. There will be 5 parts in question no 1.

End-semester question paper comprises of 8 questions and students have to answer any six questions including question no 1, which is compulsory. Weightage for 1st question is 10 and 8 for others. There will be 10 parts in question no 1.

10. Activity based Teaching and Learning:

Considering the guidelines circulated and after discussing with the faculty members, following activity based teaching and learning is proposed:

10.1. Activity List

Component wise distributions of the activities are listed below.

Problem Solving	Critical Thinking	Quiz
15	10	5

Considering the guidelines circulated and after discussing with the faculty members, following component wise description of each activity list is proposed:

10.2. Problem solving (15 marks): Assignment

Assignments have to be solved in a group/individual and mentioned below for reference only. Faculties are free to give their own assignments and evaluation is to be done by respective assigned subject teacher. Subject teacher have to decide the number of groups and students for each group. Students are expected to write the solution in the writing pad and submit the soft copy to the subject teacher.

Assignment-1 (Introduction)

- Find the time complexity of the following code segment

Q.

```
for (i = 1; i <= n; i = i*2)
{
 for (j = n; j >= 1; j = j/2)  { some statement }
}
```

Q.

```
for (i = 1; i <= n; i = i*2)
{
 for (j = n; j >= 1; j = j/2)  { some statement }
}
```

Q.

```
for (i = 1; i <= n; i = i*2)
{
 for (j = n; j >= 1; j = j/2)  { some statement }
}
```

- Write an recursive algorithm to print from 1 to n (where n > 1)
- Write an recursive algorithm to find the kth smallest element of a set S
- Write an recursive algorithm to sum the list of numbers
- Find the time and space complexity of the following code segment

Q.

```
int Factorial (int n)
{
  if n == 0 then
    return 0;
  else
    return n * Factorial(n – 1)
}
```

Q.

```
int Fib(int n)
{
  if n <= 1 then
    return 1;
  else
    return Fib(n-1) + Fib (n – 2)
}
```

Q.

```
int GCD(int x, int y)
{
  if y == 0 then
    return x
  else if  x >= y AND y > 0
    return GCD (y, x % y)
   else
     return 0;
}
```

Assignment-2 (Arrays)

- Write an algorithm to add the original sparse matrix with the transpose of the same matrix.
- Write an algorithm to multiply two sparse matrices.
- Design an efficient data structure to store data for lower and upper triangular matrix.
- Design an algorithm to convert a lower triangular matrix to upper triangular matrix.
- Design an efficient data structure to store data for tri-diagonal matrix.
- Write an algorithm that takes as input the size of the array and the elements in the array and a particular index and prints the element at that index.
- Write down the algorithm to sort elements by their frequency.
- Write down the algorithm to add two polynomials of single variable.
- Write down the algorithm to multiply two polynomials of two variables.
- A program P reads in 500 random integers in the range [0..100] presenting the scores of 500 students. It then prints the frequency of each score above 50. What would be the best way for P to store the frequencies?
- Write down the algorithm to delete all the vowels in a character array.
- Write down the algorithm to print all the elements below the minor diagonal in a 2-D array.
- Write an algorithm to find a triplet that sum to a given value
- Given an array arr, write an algorithm to find the maximum j – i such that arr[j] > arr[i]
- Write an algorithm to replace every element in the array with the next greatest element present in the same array.

Assignment-3 (Linked List)

- Design algorithm/develop pseudocode/write C code to add a given value K to each element in the LIST.
- Design algorithm/develop pseudocode/write C code to deletes the last node from the LIST.
- Design algorithm/develop pseudocode/write C code to delete the 1st node from the list.
- Design algorithm/develop pseudocode/write C code which interchanges the Kth and K+1st elements
- Design algorithm/develop pseudocode/write C code to swap nodes in a linked list without swapping data.
- Design algorithm/develop pseudocode/write C code to reverse the nodes in a linked list.
- Design algorithm/develop pseudocode/write C code to create a linked list from a given linked list. The new linked list must contain every alternate element of the existing linked list.
- Design algorithm/develop pseudocode/write C code to count the number of times a given key occurs in a linked list
- Let a linked list consists of n number of nodes, where each node consists of an unique number, a priority number(in between 1 to 5), and pointer to next node Design the algorithm/develop pseudocode/write C code to divide the nodes into different linked list where each linked consists of nodes having same priority.
- Design algorithm/develop pseudocode/write C code to delete n nodes after m nodes of a linked list
- Design algorithm/develop pseudocode/write C code to check if a singly linked list is palindrome
- Design algorithm/develop pseudocode/write C code to search an element in a linked list, if found delete that node and insert that node at beginning. Otherwise display an appropriate message.
- Design algorithm/develop pseudocode/write C code to add, subtract and multiply 2 polynomials.
- Design algorithm/develop pseudocode/write C code to delete last occurrence of an item from linked list

- Given a singly linked list with nodes L0 -> L1 -> … -> Ln-1 -> Ln. Design the algorithm/develop pseudocode/write C code to rearrange the nodes in the list so that the new formed list is : L0 -> Ln -> L1 -> Ln-1 -> L2 -> Ln-2 …

Assignment-4 (Stack & Queues)

- Write an algorithm to convert an infix expression into it equivalent postfix expression. Explain the execution of algorithm using the following expression (A+B) * C – (D * E) î (F + G + (H * M))
- Write pseudo code to check whether a given postfix expression is correctly parenthesized
- Evaluate the postfix expression: 5 3 2 * + 7 9 /4 * 2 / - 6 + 2 –
- Write C functions for insertion, deletion, traversal operation for circular queues
- Write C functions for insertion, deletion, traversal operation for input restricted deques
- Write an algorithm to copy the data elements of one stack to other without changing the order and without using any other data structure
- Write C functions for insertion, deletion, traversal operation for priority queues
- Write an algorithm to check for balanced parentheses (,), {, }, [and] in an expression
- Write recursive pseudo code or recursive function to display the string in reverse order.
- Write an algorithm to convert postfix expression to infix expression.
- Write an algorithm to convert prefix expression to infix expression.
- We are given stack data structure, the task is to implement queue using only given stack data structure.
- We are given queue data structure, the task is to implement stack using only given queue data structure.
- Write an algorithm or c code segment for insertion, deletion, peek and traversal of priority queue using dynamic arrays
- Write an algorithm or c code segment for Postfix to Infix conversion
- Write an algorithm or c code segment for Prefix to Infix conversion
- Write an algorithm or c code segment for Postfix to Prefix conversion
- Write an algorithm or c code segment for Postfix to Prefix conversion
- Write C functions for insertion, deletion, traversal operation for input restricted deques

Assignment-5 (Trees)

- A binary tree has 9 nodes. The inorder and preorder traversal of T yields the following sequence of node. Inorder: E, A, C, K, F, H, D,B,G and Preorder: F, A, E, K, C, D, H, G, B. Draw the tree T
- Suppose T is a binary tree. Write a recursive & non-recursive algorithm to find the number of leaf nodes, number of nodes and depth in T. Depth of T is 1 more than the maximum depths of the depths of the left and right subtrees of T.
- Suppose T is a binary tree. Write a recursive & non-recursive algorithm to find the height of T.
- Insert the following keys in the order to form the binary search tree J, R, D, G, T, E, M, H, P, A, F, Q
- Insert the following keys in the order to form the binary search tree 50, 33, 44, 22, 77, 35, 60, 40
- Suppose a binary tree T is in memory. Write the pseudo code to delete all terminals or leaf nodes in T
- Write a C function to copy the binary tree T to T'
- Suppose ROOTA points to a binary tree T1. Write the pseudo code which makes a copy T2 of the tree T1 using ROOTB as pointer
- Write algorithm to insert and delete the nodes in B tree.
- Write a program to check whether the binary tree is a binary search tree
- Write the non-recursive function for inorder traversal, preorder traversal, and postorder traversal of a binary tree.

- Write a function to display all the paths from root to leaf nodes in a binary tree.
- Write a programme to insert a node into BST both in recursive and non-recursive manner.
- Write a programme to display the nodes of a tree in level wise(first we need to display 0th level node, then 1th level nodes and so on).
- Write a program to check whether a binary tree is an AVL tree.

Assignment-6 (Graphs)

Write C code snippet (function) to

- Represent the graph using Adjacency Matrix
- Represent the graph using Adjacency List
- Find if there is a path between pair of vertices in an undirected and directed graph
- Find the number of isolated vertices in an undirected graph.
- Check if a given undirected graph is a tree (an undirected graph is a tree is there is cycle and is connected)
- Find a mother vertex in a graph. Mother is a vertex v such that all other vertices in G can be reached by a path from v. There can be more than one mother vertices in a graph. Need to output anyone of them.
- Detect a cycle in directed graph
- Find the bridges in the undirected graph. An edge in an undirected connected graph is a bridge if removing it disconnects the graph

Assignment-7 (Searching & Sorting)

- Let A[1], A[2], . . . , A[n] be an array containing very large positive integers. Describe an efficient algorithm to find the minimum positive difference between any two integers in the array. What is the complexity of your algorithm? Explain.
- Design an efficient algorithm to sort 5 distinct keys.
- Let A = A[1], . . . , A[n] be an array of n distinct positive integers. An inversion is a pair of indices i and j such that i < j but A[i] > A [j]. For example in the array [30000, 80000, 20000, 40000, 10000], the pair i = 1 and j = 3 is an inversion because A [1] = 30000 is greater than A[3] = 20000. On the other hand, the pair i = 1 and j = 2 is not an inversion because A [1] = 30000 is smaller than A [2] = 80000. In this array there are 7 inversions and 3 non-inversions. Describe an efficient algorithm that counts the number of inversions in any array. What is the running time of your algorithm?
- We have a N (very large number of) sales records. Each record consists of the id number of the customer and the price. There are k customers, where k is still large, but not nearly as large as N. We want create a list of customers together with the total amount spent by each customer. That is, for each customer id, we want to know the sum of all the prices in sales records with that id. Design a sensible algorithm for doing this.
- What is the average and worst time complexity for insertion, deletion and access operation for the hash table?
- Mathematically compute the worst case time complexity of binary search

10.3. Critical thinking (10 marks): Mini-Project

Critical thinking process is related to demonstrating the mini-project and is the group wise activity. The group has to submit the source code and 2 pages report capturing design aspect of the project like data structure used, algorithm used with space and time complexity and the input and output of the project. Following mini-projects are proposed and for reference:

Sr#	**Unit**	**Problem Description**
1	Arrays	Write a menu driven program using DMA and pointers to - insert, delete and update an item at any given index. - rotate the array by d elements - move all odd elements to the end/ beginning - move all even elements to the end/ beginning - find the largest and smallest in the array

		The program should handle erroneous condition.
2	Liked List	Write a menu driven program (covering single, double and circular kinked list) to - insert, delete and update a key at beginning, end or at any intermediate position. - reverse the list - split into 2 halves - search a key - Find the length - Remove the duplicates - detect loop The program should handle erroneous condition.
3	Stacks	Write a menu driven program (covering array and linked based) to - insert, delete an item. - reverse the stack - delete the middle most item - conversion (prefix to infix, postfix to prefix, postfix to infix etc) - expression evaluation The program should handle erroneous condition.
4	Queues	Write a menu driven program (covering array, linked based, Linear queue, Circular Queue and Priority Queue) to - insert, delete an item. - reverse the stack - delete the middle most item The program should handle erroneous condition.
5	Trees	Write a menu driven program (covering binary search trees, threaded binary tree, m-way search tree) to - insert, delete an item. - in-order, pre-order, level-order and post-order traversal - search for the specific item The program should handle erroneous condition.
6	Graph	Write a menu driven program (covering Adjacency Matrix, Adjacency List, Cyclic and Acyclic) to - insert, delete nodes and edges. - perform BFS and DFS traversal - search for the node and edge The program should handle erroneous condition.
7	Searching and Sorting	**Q.** Write a menu driven program (covering all searching techniques) to search for an item in the array **Q.** Write a menu driven program (covering all sorting techniques) to sort the array in ascending and descending order

10.4. Quiz (5 marks): Mini-Project

Two/Three quizzes with easy, moderate and difficulty level will be conducted at the mid and end of semester according to the standard of GATE. Sample quiz is shown for reference only. Faculties are free to give their own questions in the quiz. Evaluation is to be done by respective assigned subject teacher.

- Consider a linear array int array[3][2] = {{45,23},{333,21},{90,45}} in 16-bit OS and the base address of the array is 1000 and is presented in memory as Row Major. What is the address of the element located at the index [1][1] ___________________?
- Consider the tree arcs of a BFS traversal from a source node W in an unweighted, connected, undirected graph. The tree T formed by the tree arcs is a data structure for computing.

  (A) the shortest path between every pair of vertices.

  (B) the shortest path from W to every vertex in the graph.

  (C) the shortest paths from W to only those nodes that are leaves of T.

  (D) the longest path in the graph