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

School of Computer Engineering
Autumn Semester 2023

Course Handout

1. **Course code** : CS 21001
2. **Course Title** : Data Structures
3. **LTP Structure** :

L	T	P	Total	Credit
3	1	0	4	4

4. **Course Faculty** :

Name : Dr. Jagannath Singh
Email ID : jagannath.singhfcs@kiit.ac.in
Contact No. : 9861085883
Instructor *Chamber:* Faculty Room: 11, Ground Floor, Block-B, Campus-15
Tentative Consulting Hours: Tuesday (9-10 AM), Wednesday (9-10 AM).

5. **Base-lined date** : 10/08/2023
6. **Course offered to the School** : School Computer Engineering, School Electrical Engineering, School Electronics Engineering

7. **Course Objective:**

- To find the time complexity and space complexity of algorithms
- To design and implement operations on arrays, stacks, queues, and linked lists
- To understand the complex data structures such as tree and graph
- To understand various techniques of sorting and searching
- To solve real life problems

8. **Course Outcome:** At the end of the course, the students will be able to:

CO #	Detail
CO1	Remember the concept of data structure and choose appropriate data structures to represent data items for real world problems.
CO2	Understand the time and space complexity, string representation and manipulations, and pattern matching.
CO3	Understand the data structures such as arrays, linked lists, stacks and queues, dynamic storage management, garbage collection and compaction to develop solution for real-world scenarios.
CO4	Understand non-linear data structures such as trees and graph to develop solution for real world scenarios.
CO5	Apply algorithms for solving problems like sorting, searching, hashing and symbol table.
CO6	Analyze the data structure that efficiently models the solution to a problem.

9. Course Contents

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.

Sr#	Major Area	Detailed Area	CO
1	Introduction	Development of Algorithms, Notations and analysis, Storage structures for arrays, Sparse matrices, Stacks and Queues: Representations and applications.	CO1, CO2, CO3
2	Linked List, Stacks, and Queues	Linked Lists, Linked stacks and queues, Operations on polynomials, Doubly linked lists, Circularly linked lists, Dynamic storage management, Garbage collection and compaction.	CO2, CO3
3	Trees	Tree representation, Binary Trees, Binary search trees, Tree traversal, Expression manipulation, Symbol table construction, Height balanced trees, AVL trees.	CO4, CO6
4	Graph	Graphs, Representation of graphs, BFS, DFS, Topological sort, String representation and manipulations, Pattern matching.	CO4, CO6
5	Sorting and Searching	Sorting Techniques: Selection, Bubble, Insertion, Merge, Heap, Quick, Radix sort, Linear search, Binary search, Hash table methods.	CO5, CO6

10. Text Book:

1. Sartaj Sahni, "Data Structures, Algorithms and Applications in C++", Universities Press (I) Pvt. Ltd., 2008.

11. Reference Books:

1. J. P. Tremblay, P. G. Sorenson, "An Introduction to Data Structures with Applications", Second Edition, Tata McGraw Hill, 1981.
2. M. Tenenbaum, Augestien, "Data Structures using C", Third Edition, Pearson Education, 2007.
3. Mark Allen Weiss, "Data Structures and Algorithm Analysis in C", Second Edition, Addison-Wesley Educational Publishers, 2006.

12. Reference Site:

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

14. Lesson Plan:

Lect No.	Unit	Topics	Lesson #
1-4	Introduction	• Course coverage • Introduction	1
		• Algorithm specification • Algorithm analysis	2
		• Time complexity • Space complexity • Class work	3
		• Problem solving	4
		• Array Introduction • 1-D array address calculation • Row and column major order and address calculation	5-6

		• Array abstract data type(ADT) • Problem solving	7
		• Polynomial & its operation • Matrix operations	8
		• Sparse matrix and its operation • Class work	9
		• String representation and manipulations	10
		• Pattern matching	11
		• Tutorial class	12
13-22	Linked List	• Introduction to Linked List • Advantages, Disadvantages, Applications • Representation	13-14
		• Class work • Types of Linked List	15
		• Single Linked List Operation – Traversal, Insertion, Deletion, Insert Last, Delete Last • Class work	16-17
		• Double Linked List Operations • Class work	18
		• Circular and Header Linked List: Operations • Class Work	19
		• Polynomial Representation • Polynomial Operations	20
		• Dynamic storage management • Garbage collection and compaction	21
		• Tutorial Class	22
23-32	Stacks & Queues	• Introduction to Stack • Stack Application • Stack Representation – Arrays	23
		• Stack Representation – Linked List • Stack ADT	24
		• Arithmetic Expression Evaluation • Class Work	25
		• Arithmetic Expression Conversion • Class Work	26
		• Introduction to Queues • Queues Application • Queues Representation – Arrays	27
		• Queues Representation – Linked List • Queues ADT • Class Work	28
		• Linear Queue Drawback • Circular Queues	29
		• Deques and its Operation	30
		Mid Semester Examination	
		• Priority Queue • Class Work	31
		• Tutorial Class	32
33-42	Trees	• Introduction to Trees • Trees Terminology • Class Work	33

		• Tress Application • Binary Tree – Full, Complete and Extended Binary Trees • Expression Trees • Class Work	34
		• Representation of Binary Tree – Linked and Array Representation • Binary Tree ADT	35
		• Arithmetic Expression Conversion • Class Work	36
		• Binary Tree Traversal Concept and Algorithm – In-Order, Pre- Order and Post-Order & Level- Order • Binary Tree Construction with different traversal	37
		• Class Work on Binary Tree • Threaded Binary Tree – Single and Double Threaded	38
		• Binary Search Tree • BST ADT – Search, insertion	39
		• BST ADT – Deletion, • Class Work	40
		• Balanced Binary Tree • AVL Tree • AVL Rotation Techniques, ADT	41
		• Tutorial Class	42
43-47	Graphs	• Introduction to Graph • Graph Terminology • Graph Application	43
		• Graph Representation • Class Work	44
		• Graph Operation – DFS and BFS, Topological sort • Class Work	45-46
		• Tutorial Class	47
48-51	Sorting	• Bubble Sort • Insertion Sort • Selection Sort	48
		• Quick Sort • Merge Sort	49
		• Heap Sort • Radix Sort	50
		• Tutorial Class	51
52-55	Searching, Hashing	• Linear Search • Binary Search	52
		• Hashing – Hash Function • Symbol table construction • Class Work	53-54
		• Hashing – Collision Resolution Technique • Tutorial Class	55

13. Assessment plan for active based teaching learning:

Considering the guidelines circulated and after discussing with the faculty members, following activity based teaching and learning is proposed to have the uniformity of subject delivery in all sections:

Assessment Components:

Sr #	Assessment Component	Time	Weightage/ Marks	Schedule
1	Mid-Semester Examination	1 and half Hrs	20	Refer Details in Student Handbook
2	Activity based Teaching and Learning	Throughout semester	30	Throughout semester
3	End-Semester Examination	3 Hrs	50	Refer Details in Student Handbook

Activity Calendar

Sl. No.	Activity Name & Course Outcome Mapping	Activity Type & Submission Mode	Marks	Tentative Schedule
1	Class Participation [CO-1, 2, 3, 4, 5 & 6]	Individual / During Class Hours	5	In-Class
2	Quiz/ Class Test –I [CO-1 & 4]	Individual / Google Class Room	5	2nd Week of August'23
3	Quiz/ Class Test –II [CO-2, 3 & 4]	Individual / Google Class Room	5	1st Week of September'23
4	Quiz/ Class Test –III [CO-3 & 4]	Individual / Google Class Room	5	2nd Week of October '23
5	Assignment [CO-3 4, 5, 6]	Group (Mini Project) / Google Class Room	5	3rd Week of October'23
6	Quiz/ Class Test –IV [CO-3, 4, 5]	Individual / Google Class Room	5	1st Week of November '23
7	Quiz/ Class Test –V [Extra] [CO-1, 2, 3, 4, 5 & 6]	Individual / Google Class Room	5	3rd Week of November'23

Component wise distributions of the activities are listed below

Quiz (Problem Solving and Critical Thinking)	Interactive focus (Viva)	Assignments Problem Solving and Critical Thinking
Module wise or content wise several quizzes will happen (Each quiz will be of 5 marks)	Faculty Choice based Activity (during the class) (5 Mark)	**Assignment (Mini Project)** (15 Mark) (Based on Problem Solving and Critical Thinking)

14. Attendance: Every student is expected to be regular (in attendance) in all lecture classes, tutorials, labs, tests, quizzes, seminars etc and in fulfilling all tasks assigned to him / her. Attendance will be recorded and 75% attendance is compulsory.

15. Makeup:

- No make-up examination will be scheduled for the mid semester examination. However, official permission to take a make-up examination will be given under exceptional circumstances such as admission in a hospital due to illness / injury, calamity in the family at the time of examination.
- A student who misses a mid-semester examination because of extenuating circumstances such as admission in a hospital due to illness / injury, calamity in the family may apply in writing via an application form with supporting document(s) and medical certificate to the Director General of the School for a make-up examination.
- Applications should be made within five working days after the missed examination.

16. Discussion of Mid Semester performance: Performance of the mid semester examination will be discussed in the class room

17. Pre-end semester total marks: To be announced and discussed in the class.

18. **Course Management System:** SAP Portal - is a software system designed to facilitate teachers in the management (instructional content, assessment and documentation) of the courses for their students, both teachers and students can monitor the system. Though usually considered as a tool and often used to complement the face-to-face classroom.

19. **Assignments (15 marks):**

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)

```
Q.
int Factorial (int n) {
        if n == 0 then
                return 0;
        else
                return n * Factorial(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 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 a program to add the original sparse matrix with the transpose of the same matrix.
- Write a program to multiply two sparse matrices using 3-tuple method.
- Design an efficient algorithm to convert a lower triangular matrix to upper triangular matrix.
- Write an algorithm to sort elements by their frequency in decreasing order.
- Write an algorithm to add two single variable polynomials of degree n.
- Write an algorithm to multiply two 2-variable polynomials of degree n.
- A program P reads in 500 arbitrary 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 an algorithm to delete all the vowels in an array of characters.
- Write an algorithm to print all the elements below the minor diagonal in a 2-D array.
- Write an algorithm to find all triplets that sum to a given value in an array.
- 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 to interchanges the Kth andK+1st elements without swapping their values.
- Design algorithm to create a new linked list from a given linked list. The new linked list must contain every alternate element of the existing 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 to divide the nodes into different linked list where each linked consists of nodes having same priority.
- Design algorithm to delete n nodes after m nodes of a doubly linked list.
- Design algorithm to check if a singly linked list is palindrome or not.
- Design algorithm to search an element in a doubly linked list,if found delete that node and insert that node at beginning. Otherwise display an appropriate message.
- Write a menu driven program to add, subtract and multiply 2 single variable polynomials.
- Design algorithm 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 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 deque.
- 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 operations for priority queues.
- Write an algorithm to check for balanced parentheses (,), {, }, [and] in an infix expression.
- Write a recursive algorithm 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.
- Design algorithm to implement queue using only given stackdata structure.
- Design algorithm to implement stack using only given queuedata structure.
- Write an algorithm for insertion, deletion, peek and traversal of priorityqueue using dynamic array.
- Write an algorithm to convert a postfix expression to an infix.
- Write an algorithm to convert a prefix expression to an infix.
- Write an algorithm to convert a postfix expression to prefix.

- Write C functions for insertion, deletion, traversal operation for output restricted deque.

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, A. 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 keys: J, R, D, G, T, E, M, H, P, A, F, Q in the order to form the binary search tree.
- Insert the keys: 50, 33, 44, 22, 77, 35,60, 40 in the order to form the binary search tree.
- Suppose a binary tree T is in memory. Write the pseudo code to delete all terminals/ leaf nodes in T.
- Write a C function to copy the binary tree T to T'.
- 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 or not.
- 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 program to insert a node into BST both in recursive and non-recursive manner.
- Write a program 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 or not.

Assignment-6 (Graphs)

Write 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).
- 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

20 Critical thinking (5 marks): Mini-Project / Viva

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 linkedbased) 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

21 Quiz (10 marks):

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.

i. 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]_____?

ii. 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

Dr. Jagannath Singh
Course Faculty