Gyanmanjari Innovative University

Course Syllabus
Gyanmanjari Institute of Technology
Semester-3 (B. Tech.)

Subject: Data Structures: Algorithms and Efficient Problem Solving - BET2CS13304

Type of course: Professional Core

Prerequisite: Basic Knowledge of JAVA

Rationale:

This course is designed to learn essential concepts of Data Structures in computer science that help in organizing, managing, and storing data efficiently for easy access and modification. They are broadly classified into linear data structures like arrays, stacks, and queues, where elements are arranged sequentially, and non-linear data structures such as trees and graphs, which represent more complex relationships. A strong grasp of these structures is vital for designing effective algorithms. Additionally, understanding searching methods (like linear and binary search), sorting algorithms (such as bubble sort and quick sort), and hashing techniques plays a crucial role in optimizing program performance and solving computational problems efficiently. The Java programming language is used as the teaching vehicle for this course.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
4	0	2	5	100	50	150

Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation, CCE-Continuous and Comprehensive Evaluation.

Course Content:

Sr. No	Course Content	Hrs.	% Weightage
1	**Basic Concept of Data Structures:** ***Theory Topics:*** Data Structure Basic Concepts, Problem Analysis, Types of Data Structures Linear and Non- Linear, differentiate primitive and non-primitive data structures, Introduction to Algorithms, Key features of an algorithm, Analysis Terms (For the definitions purpose only): a. Time Complexity and Space Complexity, Arrays: Representation of single- and two-dimensional arrays with two array addition, subtraction and multiplication, Searching Techniques: Linear/Sequential Search and Binary Search. ***Practical:*** 1) Write a program to create an array with five elements and display each of them. 2) Write a program that will display array in reverse order. 3) Write a program for addition and subtraction of two arrays and store the result in a new separate array. 4) Write a program to find sum and average of all array elements. 5) Write a program to find the highest and lowest element in an array. 6) Write a program to copy one array to another array. 7) Write a program to split one array into two parts. 8) Write a program to create 3x3 matrix and display in matrix form. 9) Write a program to perform addition of 3x3 matrix. 10) Write a program to perform subtraction of 3x3 matrix. 11) Write a program to perform multiplication of 3x3 matrix. 12) Write a program to calculate the sum of each row in a 3x3 matrix. 13) Write a program to calculate the sum of each column in a 3x3 matrix. 14) Write a program to perform linear search for an element in a list. 15) Write a program to perform binary search for an element in a list.	18	20%

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Smart Inventory System Using Arrays and Search:** Develop a basic inventory system using arrays and suitable searching techniques.	20	-
2	**Active Learning Activity (ALA) – Data Structure Quiz:** Attempt a unit-wise MCQ quiz consisting of a maximum of five questions.	-	05
3	**Viva:** Answer five oral questions based on the concepts covered in the unit.	-	05
	Total	20	10

Examination Style:

1. **Smart Inventory System Using Arrays and Search (20 Marks)**
   - Students will simulate a basic inventory or product-tracking system using arrays. They must analyse the given problem, store and manage product information using arrays, and perform suitable array operations such as addition, subtraction, and multiplication. Students will implement linear search and binary search techniques to locate product records and provide a written explanation of the algorithms used. They must also explain the time and space complexity of the implemented operations. The activity will assess students' problem-solving ability, understanding of arrays, searching techniques, algorithms, and complexity analysis.
2. **Active Learning Activity (ALA) – Data Structure Quiz of Unit 1 (05 Marks)**
   - A unit-wise multiple-choice quiz consisting of a maximum of five questions will be conducted. The questions will be based on the concepts covered in Unit 1, including problem analysis, arrays, array operations, algorithms, linear search, binary search, and time and space complexity. The quiz will assess students' conceptual understanding and recall of the unit.
3. **Viva of Unit 1 (05 Marks)**
   - Five viva questions will be asked individually from Unit 1 to assess students' recall, understanding, and ability to explain the concepts of arrays, array operations, algorithms, linear search, binary search, and time and space complexity.

2

Stack and Queues:

Theory Topics:

Stack: Linear Data Structure, Stack Concept, Array representation of stack, Operations on Stack: PUSH, POP, PEEK, DISPLAY, Stack Applications: Infix, Postfix, Prefix expressions, Evaluation of Postfix Expression, Recursive Functions: Factorial, Queue: Queue Concept, Array representation of queue, Operations on Queue: ENQUEUE, DEQUEUE, DISPLAY, Limitation of Simple Queue, Types of Queue: Simple Queue, Circular Queue, Double Ended Queue(DEQUES), Priority Queue, Application of Queues, Differentiate simple queue and circular queue.

Practical:

1. Write a program to implement stack operations using arrays: PUSH, POP, PEEP and DISPLAY.
2. Write a program to evaluate an expression in to Infix.
3. Write a program to evaluate an expression in to Postfix.
4. Write a program to evaluate an expression in to Prefix.
5. Write a program to convert an infix arithmetic expression into postfix notation.
6. Write a program to perform the following operation on a simple queue using Arrays: ENQUEUE, DEQUEUE, DISPLAY
7. Write a program to perform the following operation on a circular queue using Arrays: ENQUEUE, DEQUEUE, DISPLAY

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Code Optimization:** Optimize and refactor Java programs related to data structures without changing their output.	20	-
2	**Active Learning Assignment – Problem - Solving Using Programming:** Solve the assigned problem in a group of two and upload the solution to the GMIU Web Portal.	-	10
	Total	20	10

18

20%

	Examination Style: 1. **Code Optimization (20 Marks)** • Students will be provided with two suboptimal Java programs based on data structure concepts, similar to unsolved examples from reference books. They must analyse, optimize, and refactor the programs to improve their performance, readability, structure, and efficiency without changing the expected output. The evaluation will assess students' understanding of data structures, algorithmic efficiency, logical reasoning, coding practices, and their ability to identify and remove unnecessary or inefficient operations. 2. **Active Learning Assignment – Problem-Solving Using Programming (10 Marks)** • Students will work in groups of two and receive problem statements during the laboratory session. Each group must analyse and solve its assigned problem using an appropriate programming approach. The completed work must be uploaded to the GMIU Web Portal within the specified timeline and should include the problem statement, source code, and program output.		
3	**Linked List:** ***Theory Topics:*** Linked list representation, Types of Linked List: Singly Linked List, Doubly Linked List, Circular Linked List, Operations on Linked List, Creation of node, Insert node at first, Insert node at last, Insert after a given node, insert before a given node, Insert at given position, Delete node at first, Delete node at last, Delete a given node value, Search a specific node based on value, Count the number of nodes in linked list, Display Linked List, Linked implementation of Stack, Linked implementation of Queue, Applications of Linked List. ***Practical:*** 1. Write program perform the following operations on a singly linked list: Creation of node, Insert node at first, Insert node at last, Insert after a given node, insert before a given node, Insert at given position, Delete node at first, Delete node at last, Delete a given node value, Search a specific node based on value, Find the sum of elements of the list, Count number of the nodes in the linked list, Reverse the linked list. 2. Write a program perform the following operations on a doubly linked List: Creation of node, Insert node at first, Insert node at last, Insert after a given node, insert before a given node, Insert at given position, Delete node at first, Delete node at last, Delete a given node value,	18	20%

Search a specific node based on value, Find the sum of elements of the list, Count number of the nodes in the linked list.

3. Write a program to create a singly linked list in LIFO fashion (As Stack).
4. Write a program to create a singly linked list in FIFO fashion (As Queue).
5. Write a program perform the following operations on a circular singly linked List: Insert an element, Delete an element, Find the sum of elements of the list, Count number of the nodes in the linked list, Search a given elements in the linked list.
6. Write a program perform the following operations on a circular doubly linked List: Insert an element, Delete an element, Find the sum of elements of the list, Count number of the nodes in the linked list, Search a given elements in the linked list.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Linked List LifeLine – Hospital Patient Tracker** Students will simulate a **hospital's patient admission system** using various **types of linked lists**. The program should handle real-world operations using linked list.	20	-
2	**Active Learning Assignment Data Structure Interactive Visualization:** Find out interactive visualization of data structures. This can help you visualize how data is stored and manipulated within different structures. Upload a soft copy on GMIU Web Portal. (Group of two students)	-	10
	Total	20	10

Examination Style:

1. **Linked List Lifeline – Hospital Patient Tracker (20 Marks)**
   - Students will develop a hospital patient-admission and tracking system using appropriate linked-list structures. The program should support real-world operations such as admitting a patient, displaying patient records, searching for a patient, updating patient information, discharging a patient, and traversing the complete patient list. Students may apply singly linked lists, doubly linked lists, or circular linked lists

	based on the given requirements. The evaluation will assess their understanding of linked-list operations, dynamic memory management, problem-solving ability, program correctness, and handling of boundary conditions. 2. **Active Learning Assignment – Data Structure Interactive Visualization of Unit 3 (10 Marks)** • Students will work in groups of two and explore an interactive visualization tool related to data structures. They must study and demonstrate how data is stored, accessed, inserted, deleted, and manipulated within the selected data structure. Each group must prepare a soft copy containing the name of the visualization tool, the selected data structure, screenshots, and a brief explanation of the observed operations. The completed activity must be uploaded to the **GMIU Web Portal**.		
4	**Trees and Graph:** ***Theory Topics:*** Non-Linear Data Structure: Tree and Graph, Basic Terminology: Tree, Root, Node, Edge, Parent, Child, Siblings, Leaf Node, Internal Node, Degree, Level, Height, Depth, Path, Sub Tree, directed tree, un-directed tree, Binary Tree: Complete Binary Tree and Strict Binary Tree, Tree Linked List Representation, Binary Tree Traversal: Inorder, Preorder, Postorder, Operations on a Binary Tree: Insertion, Deletion, Search, Display using tree traversal, Types of Trees: Binary Search Tree, AVL/ Height Balanced Tree, Threaded Binary Tree, Applications of Tree. **Graph:** Basic Terminology: Graph, Undirected, Directed, Path, Closed Path, Simple Path, Cycle, Connected Graph, Complete Graph, Weighted Graph, Loop, Degree of the node, Adjacent Node, Graph Representation: Adjacency Matrix and Linked List, Traversal of Graph: Breadth First Search (BFS) and Depth First Search (DFS) ***Practical:*** 1. Write a program to create a binary search tree and display its elements in inorder traversal. 2. Write a program to create a binary search tree and display its elements in preorder traversal. 3. Write a program to create a binary search tree and display its elements in postorder traversal. 4. Write a program to delete an element from a binary search tree. 5. Write a program binary search tree with all operations.	**18**	20%

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Binary Search Tree Operations:** Build and perform essential operations on a Binary Search Tree using city names or city codes.	20	-
2	**Thinking, Pairing, and Sharing (TPS):** Individually solve a tree or graph problem, discuss it with a partner, and present the final solution to the class.	-	10
	Total	20	10

Examination Style:

1. **Binary Search Tree Operations (20 Marks)**
   - Students will develop a Binary Search Tree using city names or city codes as node values. The program should support operations such as inserting city nodes, searching for a city using its name or code, deleting a specified node, and displaying the tree using suitable traversal techniques such as inorder, preorder, and postorder traversal. The evaluation will assess students' understanding of Binary Search Tree properties, node insertion, searching, deletion, traversal techniques, program correctness, and handling of different tree conditions.
2. **Thinking, Pairing, and Sharing (TPS) for Think:**
   - Students will participate in the Thinking, Pairing, and Sharing activity based on tree and graph concepts. During the **Think** stage, each student will individually analyse and solve a given problem, such as constructing a binary tree, performing tree traversal, or applying DFS or BFS on a graph. During the **Pair** stage, students will work in groups of two to compare their solutions, discuss alternative approaches, and refine their understanding. During the **Share** stage, each pair will present its solution and key observations to the class. The faculty member will review the responses, clarify misconceptions, and explain the correct approach.

5	**Sorting and Hashing:** ***Theory Topics:*** **Sorting:** Introduction, Types of Sorting: Insertion Sort, Selection Sort, Bubble Sort. Merge Sort, Quick Sort, **Hashing:** Introduction to Hashing, Hash Table, Hashing Applications	18	20%

Practical:

1. Write a program to implement an insertion sort.
2. Write a program to implement a selection sort.
3. Write a program to implement a bubble sort.
4. Write a program to implement a merge sort.
5. Write a program to implement a quick sort.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Code Optimization:** Optimize and refactor Java programs related to data structures without changing their expected output.	20	-
2	**Sorting Showdown:** Demonstrate and compare different sorting algorithms through a group-based practical activity.	-	10
	Total	20	10

Examination Style:

1. **Code Optimization (20 Marks)**
   - Students will be given two suboptimal Java programs related to Data Structure topics (like an unsolved example from a reference book). They are required to optimize and refactor the code for better performance, readability, and efficiency, without changing the output.

2. **Sorting Showdown for Unit – 5 (10 Marks)**
   - **Form Groups –** Assign each group a sorting algorithm (e.g., Bubble, Merge, Quick). **Simulate Sorting –** Use number cards to perform the sort manually, step by step. **Record Observations –** Note comparisons, swaps, and time taken (optional). **Present –** Each group explains and demonstrates their algorithm. **Discuss –** Compare sorting methods as a class (efficiency, use cases).

Distribution of Marks (Revised Bloom's Taxonomy)						
Level	Remembrance (R)	Understanding (U)	Application (A)	Analyze (N)	Evaluate (E)	Create (C)
Weightage %	10%	15%	20%	10%	15%	30%

Suggested Specification table:

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcome:

After learning the course, the students should be able to:	
CO1	Apply fundamental concepts of data structures, algorithms, arrays, and matrices.
CO2	Implement stack and queue operations and their applications using arrays.
CO3	Apply linked list variants and operations for dynamic data management.
CO4	Implement tree and graph representations, traversals, and applications.
CO5	Apply sorting and hashing techniques for efficient data organization and retrieval..

Instructional Method:

The course delivery method will depend upon the requirement of content and needs of students. The teacher in addition to conventional teaching method by black board, may also use any of tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

From the content 10% topics are suggested for flipped mode instruction.

Students will use supplementary resources such as online videos, NPTEL/SWAYAM videos, e-courses, Virtual Laboratory.

The internal evaluation will be done on the basis of the Active Learning Assignment.

Practical/Viva examination will be conducted at the end of semester for evaluation of performance of students in laboratory.

Reference Books:

[1]. Introducing Data Structures with Java, First Edition David Cousins, Pearson.
[2]. Data Structures and Algorithms in Java™, Sixth Edition, Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, Wiley.
[3] Think Data Structures Algorithms and Information Retrieval in Java, First Edition, Allen B. Downey, O'Reilly
[4] Data Structures and Algorithms Using Java, William McAllister, Pearson.
[5] Data Structures With JAVA, Special Indian Edition (2nd Edition), John Hubbard, McGraw Hill.
[6] Data Structures With JAVA , John R. Hubbard and Anita Huray, Pearson.
[7] Illustrating Data Structures through C, First Edition, Mansi Bosamia and Artidevi Gohil, creative crows publishers LLP.

Suggested Rubrics:

Suggested Assessment Guidelines

Module-1: Fundamentals of Data Structures, Arrays and Searching

Smart Inventory System Using Arrays and Search (20 Marks)		
Criteria	**Description**	**Marks**
Problem Analysis and System Design	Clear understanding of the inventory problem, identification of required product details, and logical planning of the solution.	05
Array Operations and Data Management	Correct use of arrays to add, update, delete, display, and manage product records.	05
Searching Techniques	Accurate implementation of linear search and binary search for locating product records.	05
Program Execution and Explanation	Correct output with a clear explanation of the algorithm and its time and space complexity.	05
	Total	20

Module-2: Stack and Queue Data Structures

Code Optimization (20 Marks)		
Criteria	**Description**	**Marks**
Identification of Inefficient Code	Correct identification of redundant statements, inefficient logic, and inappropriate data structure operations.	05
Code Optimization and Refactoring	Effective modification of the programs to improve performance, structure, readability, and efficiency.	05
Output Correctness	Preservation of the expected functionality and output after optimization.	05
Code Quality and Explanation	Properly structured code with meaningful naming and a clear explanation of the improvements made.	05
	Total	20

Module-3: Linked List Data Structures

Linked List Lifeline – Hospital Patient Tracker (20 Marks)		
Criteria	**Description**	**Marks**
Problem Analysis and Linked List Selection	Clear analysis of the hospital scenario and selection of an appropriate linked-list structure.	05
Patient Record Operations	Correct implementation of patient admission, searching, updating, deletion, and discharge operations.	05
Traversal and Boundary Handling	Accurate traversal and display of records with proper handling of empty lists and boundary conditions.	05
Program Execution and Explanation	Correct program execution, expected output, and clear explanation of linked-list operations.	05
	Total	20

Module-4: Trees and Graphs

Binary Search Tree Operations (20 Marks)		
Criteria	**Description**	**Marks**
BST Construction and Insertion	Correct construction of a Binary Search Tree using city names or codes while maintaining BST properties.	05
Search and Deletion Operations	Accurate implementation of searching and deletion operations for different node conditions.	05
Tree Traversals	Correct implementation and display of inorder, preorder, and postorder traversals.	05
Program Execution and Explanation	Correct output, appropriate handling of tree conditions, and clear explanation of BST operations.	05
	Total	20

Module-5: Sorting and Hashing Techniques

Code Optimization (20 Marks)		
Criteria	**Description**	**Marks**
Identification of Inefficient Code	Correct identification of inefficient logic, unnecessary operations, and performance-related issues.	05
Optimization and Refactoring	Effective restructuring of the programs to improve performance, readability, and efficiency.	05
Output Correctness	Maintenance of the original functionality and expected output after optimization.	05
Code Quality and Explanation	Properly structured code with a clear explanation of modifications and efficiency improvements.	05
	Total	20