Kadi Sarva Vishwavidyalaya, Gandhinagar

25CAMJ204: Data Structures and Algorithms

Program: B.E. Computer Engineering / Computer Science and Engineering / Information Technology (Year - 2)

Teaching Scheme: 3L - 0T - 4P (5 Credits)

Prerequisite: Students should have basic knowledge of procedural programming using C, including variables, control structures, functions, pointers and fundamental problem-solving skills.

Rationale: The subject enables students to understand how data can be efficiently organized, stored, and accessed to solve complex computational problems. By learning various data structures, students develop essential skills in algorithmic thinking, problem solving, and software development efficiency.

1. **Course Objectives:**

The Purpose of learning this course is

- To develop a clear understanding of fundamental concepts of data structures and algorithm design principles including time and space complexity analysis using asymptotic notations.
- To enable students to design and implement linear data structures such as stacks, queues, linked lists, and apply them to solve real-world computational problems such as expression evaluation and memory management.
- To impart knowledge of non-linear data structures including trees and graphs, and their applications to optimize data storage and retrieval.

2. To solve network and connectivity problems (through Graph theory) to find shortest paths and minimum spanning **trees** in complex networks.

- To optimize search and sort operations to determine the most efficient method for organizing and accessing large datasets.

2. Teaching and Evaluation Scheme:

Teaching Scheme					Examination Scheme				
L	**T**	**P**	**C**	**Hrs / Week**	**IE**	**EE**	**CIA**	**Practical**	**Total Marks**
3	-	4	5	7	40	60	30	20	150

IE: Internal Exam
EE: External Exam (End Semester)
CIA: Continuous Internal Assessment
Practical: Practical Exam (End Semester)

3. Detailed Syllabus

Unit	Topic	Lectures	% weigh	Outcomes
1	**Introduction and Linear Data Structure 1:** Arrays and Structures: Abstract Data Type (ADTs), Dynamically Allocated Arrays and Structure, Asymptotic Notations of Time and Space Complexity. List ADT: Singly linked lists, Circular linked lists, Doubly linked lists, List operations (Insertion, Deletion, Copy, and Traversal), Applications of lists demonstrating lightweight and efficient data handling in real-world computing systems.	12	28%	CO1, CO5
2	**Linear Data Structure 2: Stacks and Queues:** Stack ADT: Stack Operations, Polish notation and Reverse polish, Evaluating arithmetic expressions,	11	24%	CO2, CO5

	Conversion of Infix to postfix expression, Applications of Stack, Tower of Hanoi. Queue ADT: Queue Operations, Circular queues and Priority Queues for optimized resource allocation in real-time systems, deQueue, Applications of queues.			
3	**Searching, Sorting and Hashing Techniques:** Searching: Linear search, Binary search Sorting: Bubble sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort. Hashing: Hash Functions, Open Addressing, Double Hashing, Collision Resolution Techniques. Techniques emphasizing on reducing computational cost and energy consumption through optimized algorithm selection	11	24%	CO3, CO5
4	**Non-Linear Data Structures – Trees and Graphs:** Tree ADT: Tree traversals, Binary Tree ADT, Threaded Binary Trees, AVL Trees, B-Tree, Heap, Applications of heap for efficient data storage, indexing, and retrieval in scalable systems. Graphs: Definition, Representation of Graph, Types of Graphs, Breadth-First Search, Depth First Search, Minimum Spanning Tree.	11	24%	CO4, CO5

4. List of Experiments:

No	Details	Hours	CO
1	In a classroom of 10 students, the teacher wants to find out who scored the highest marks in the final exam. The teacher gives you a list of marks as input and pass these marks to a function findTopper() that returns the highest marks.	2	CO1, CO5
2	A company has an employee whose details (ID, Name, and Salary) are stored together. The HR department wants to calculate the annual bonus (10% of salary) using a function that accepts a structure as argument.	2	CO1, CO5
3	A small library maintains details of 5 books — title, author, and price. Create a digital record using required data structures. Display all the books whose price is greater than Rs. 500.	2	CO1, CO5
4	In a small city hospital, a doctor wants to keep digital records of patients. Each patient has details such as name, age, and temperature. Whenever a patient visits, the nurse enters their details and calls a function that uses a pointer to structure to update or display the patient's record. The doctor requires: 1. A function to input patient details (using structure pointer). 2. A function to display the details. A function to update the temperature of the patient and print an alert.	2	CO1, CO5
5	At a college, the registrar maintains a list of enrolled students for a particular course. The registrar wants a program to manage the student list dynamically. Each student has: • Roll Number (unique ID) Operations in Detail: 1. Add a Student: A new student can be added at the beginning, middle, or end of the list depending on priority. 2. Remove a Student: Admission cancel of any student	8	CO1, CO5

No	Details	Hours	CO
	3. Display All Students: Show the list of all currently enrolled students in order. Search a Student by Roll Number		
6	In a small café, a cup counter can hold a maximum of 5 cups at a time. Cups are always managed in a strict top-to-bottom order — the newest cup is always placed on top, and only the top cup can be taken first. The café staff performs the following actions during the day: 1. Add a Cup: When new clean cups arrive, they are placed on the top of it—but only if the counter has space (max 5 cups). 2. Serve a Cup: When a customer orders coffee, the topmost cup is taken from the counter. (If there are no cups left, they can't serve anyone.) 3. Check a Cup: The manager can inspect a cup at a certain position from the top to ensure it's clean. 4. Replace a Cup: If a cup at a given position from the top is cracked, it can be replaced without disturbing others. Show All Cups: The manager can display all cup IDs currently on the counter, from top to bottom.	2	CO2, CO5
7	Write a program to convert the following infix expression into its postfix form using a stack. Sample Input: (A+B)∗(C+D/E)−(F+G∗H)	4	CO2, CO5
8	Solve the Tower of Hanoi problem using recursion. The program should display the sequence of moves required to transfer all disks from the source rod to the destination rod using an auxiliary rod.	2	CO2, CO5
9	At an ice cream truck, only 5 kids can wait in line at once. Kids arrive one by one and are served in the same order they come. Actions: 1. Join Line: A new kid joins at the end. 2. Serve Ice Cream: The first kid gets served and leaves. 3. Next Kid: See who's next to be served. Show Line: Display all kids currently waiting.	2	CO2, C05
10	In a library, a librarian has a row of books arranged on a shelf. Each book has a unique ID number. The librarian wants to find a specific book by its ID. She starts from the first book and checks one by one until the desired book is found. If the book is found, she notes its position on the shelf; if not, she reports it's not available.	2	CO3, C05
11	An event planner has a list of invited guests sorted alphabetically by name. When a guest arrives, the planner wants to quickly check if the guest is on the list. Write a program that takes the guest's name as input and checks whether it is present in the list. If found, display the position in the list; if not, print "Guest not invited". Optimize your search so that the program does not check every name one by one.	2	CO3, C05
12	A hospital maintains patient IDs to quickly access records. To keep the system efficient, smaller IDs go to the left and larger IDs to the right in a hierarchical structure. Write a program to insert new patient IDs into the system and Search for a patient by ID.	4	CO4, C05
13	A company maintains a hierarchy of employees. Each manager can have multiple subordinates. The CEO is at the top. Represent the company hierarchy as a tree. Implement functions to traverse the tree in different orders: 1. Visit manager before subordinates. 2. For a binary hierarchy, visit left subordinate, manager, right subordinate. Visit subordinates before the manager.	6	CO4, C05

<table>
<tr><th>No</th><th>Details</th><th>Hours</th><th>CO</th></tr>
<tr><td>14</td><td>A robot is exploring a maze. It starts at the entrance and wants to visit every path to find a treasure. The robot explores one path as far as it can go before backtracking to try other paths. Represent the maze as a graph and implement DFS to visit all locations. Display the order in which the robot visits each location.</td><td>4</td><td>CO4, C05</td></tr>
<tr><td>15</td><td>A rescue team needs to reach all houses in a flooded city. They start from the main station and want to visit all nearby houses first, then move on to houses farther away.</td><td>4</td><td>CO4, C05</td></tr>
<tr><td>16</td><td>Case Study:
Apply five different sorting algorithms (Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, and Quick Sort) in ascending order. Test them on five arrays with different sizes and characteristics. Record and compare the execution time taken by each algorithm on each array. Interpret the results to understand which algorithm performs better under different conditions.
Provided Input Data
<table>
<tr><th>Array</th><th>Size (n)</th><th>Description</th><th>Example Values</th></tr>
<tr><td>A1</td><td>10</td><td>Small random array</td><td>[9, 2, 5, 1, 8, 4, 3, 7, 6, 0]</td></tr>
<tr><td>A2</td><td>100</td><td>Medium size random array</td><td>Random integers between 1–1000</td></tr>
<tr><td>A3</td><td>1000</td><td>Large random array</td><td>Random integers between 1–10000</td></tr>
<tr><td>A4</td><td>1000</td><td>Nearly sorted array</td><td>First 950 sorted, last 50 shuffled</td></tr>
<tr><td>A5</td><td>1000</td><td>Reverse sorted array</td><td>[1000, 999, 998, …, 1]</td></tr>
</table></td><td>12</td><td>CO3, CO5</td></tr>
</table>

5. Course Outcomes (COs)

After completion of the course, the student will be able to:

CO No.	Course Outcome (CO)	RBT Level (Cognitive Domain)
CO1	Explain the concepts of Abstract Data Types (ADTs) and implement arrays, structures, and linked lists while analyzing their time and space complexity.	Understand
CO2	Apply stack and queue operations to solve computational problems such as expression evaluation, conversion, and resource management.	Apply
CO3	Compare and implement various searching, sorting, and hashing techniques for sustainable and efficient data access and storage.	Analyze
CO4	Construct and traverse tree and graph data structures to solve complex non-linear data representation problems	Apply
CO5	Design and implement efficient solutions to real-world problems using appropriate data structures and algorithms in programming languages.	Analyze, Apply

6. Mapping of COs with POs & PSOs:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	–	–	–	–	–	–	–	–	–	3	3
CO2	3	3	2	–	2	–	–	–	–	–	–	3	3
CO3	3	3	2	2	2	–	–	–	–	–	–	3	3
CO4	3	3	–	2	2	–	–	–	–	–	–	3	3
CO5	3	3	3	2	2	–	–	–	1	1	1	3	3
Avg.	**3**	**2.8**	**1.4**	**1.2**	**1.83**	**0.00**	**0.00**	**0.00**	**0.33**	**0.33**	**0.33**		

7. Reference Reading:

7.1 Text Books

1. Jean-Paul Tremblay and Paul G. Sorenson, *An Introduction to Data Structure Applications*, Tata McGraw Hill
2. Tanenbaum, *Data Structures using C & C++*, PHI
3. Robert L. Kruse, *Data Structures and Program Design in C*, PHI
4. Data Structures & Algorithms Made Easy (Narasimha Karumanchi)

7.2 Reference Books

5. Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed, *Fundamentals of Data Structures in C*, Universities Press
6. Reema Thareja Data Structures Using C, Oxford University Press

7.3 Online References

7. Data Structures and Algorithms Design
   Prof. Nitin Saxena, IIT Kanpur
   https://nptel.ac.in/courses/106104697
8. Programming, Data Structures and Algorithms using Python
   Prof. Madhavan Mukund, Chennai Mathematical Institute
   https://nptel.ac.in/courses/106106145DSA Self-Paced Course,
   https://www.geeksforgeeks.org/courses/dsa-self-paced
9. Data Structure and Algorithms using Java
   Prof. Debasis Samanta, IIT Kharagpur
   https://nptel.ac.in/courses/106105225
10. Fundamental Data Structures & Algorithms using C language, Udemy.
    https://www.udemy.com/course/data-structures-stack-queue-linkedlist/

8. Course Outcomes Mapping Table

CO	Course Outcome (CO)	POs/PSOs Map
CO1	Explain the concepts of Abstract Data Types (ADTs) and implement arrays, structures, and linked lists while analyzing their time and space complexity.	PO1, PO2, PSO1, PSO2
CO2	Apply stack and queue operations to solve computational problems such as expression evaluation, conversion, and resource management.	PO1, PO2, PO3 PSO1, PSO2
CO3	Compare and implement various searching, sorting, and hashing techniques for efficient data access and storage.	PO1, PO2, PO3, PO4, PO5 PSO1, PSO2
CO4	Construct and traverse tree and graph data structures to solve complex non-linear data representation problems	PO1, PO2, PO4, PSO1, PSO2
CO5	Design and implement efficient solutions to real-world problems using appropriate data structures and algorithms in programming languages.	PO1, PO2, PO3, PO4, PO5, PO9, PO 10, PO11, PSO1, PSO2

Sustainable Development Goals (SDG) Integration

Integration of Sustainable Development Goals (SDG 9: Industry, Innovation, and Infrastructure) in the Data Structures syllabus is achieved by embedding principles of efficient system design within existing topics. Concepts such as arrays, linked lists, trees, graphs, and algorithms are taught with a focus on time–space efficiency, scalability, and real-world applications like database systems, network optimization, and large-scale data processing. Through case-based learning, practical assignments, and assessments, students are encouraged to design resource-efficient and high-performance solutions. This approach ensures that learners understand the role of data structures in building sustainable and scalable technological infrastructure.

SDG Mapping with Course Outcomes

CO	Course Outcome	Relevant SDGs	Justification
CO1	Explain the concepts of Abstract Data Types (ADTs) and implement arrays, structures, and linked lists while analyzing their time and space complexity.	SDG 4, SDG 9	Supports efficient data representation and foundational computational skills.
CO2	Apply stack and queue operations to solve computational problems such as expression evaluation, conversion, and resource management.	SDG 8, SDG 9	Enables efficient task handling and resource optimization.
CO3	Compare and implement various searching, sorting, and hashing techniques for efficient data access and storage.	SDG 9, SDG 12	Improves processing efficiency and optimal resource use.
CO4	Construct and traverse tree and graph data structures to solve complex non-linear data representation problems	SDG 9, SDG 11	Aids in modeling and optimizing network systems.
CO5	Design and implement efficient solutions to real-world problems using appropriate data structures and algorithms in programming languages.	SDG 8, SDG 9	Promotes scalable and industry-oriented solutions.

9. Evaluation Scheme:

- The course evaluation shall comprise Internal Examination (IE) and End Semester Examination (EE) for theory, along with Continuous Internal Assessment (CIA) and External Practical Examination.
- The Internal Examination (40 marks) will be conducted through one or more mid-semester tests covering the prescribed syllabus.
- The End Semester Examination (60 marks) will assess the overall understanding of the course with emphasis on problem-solving and analytical ability.
- The Practical Examination (20 marks) shall be conducted externally and will include implementation, performance, and viva-voce.
- Students are required to secure minimum qualifying marks in each component separately as per university rules.

10. Pedagogical Framework:

- The course adopts a blended teaching approach combining lectures, coding demonstrations, and interactive problem-solving sessions.
- Emphasis is placed on learning by doing through extensive laboratory work, assignments, and mini-projects.
- Conceptual understanding is reinforced with real-world examples and algorithm visualization techniques.
- Students are encouraged to engage in collaborative learning, discussions, and peer problem-solving.
- Continuous feedback through assessments and lab performance supports progressive skill development and mastery.
- Integrates Sustainable Development Goals by embedding efficiency, scalability, and sustainability into teaching through case studies, problem-solving, and practical implementation, enabling students to design resource-efficient and real-world solutions.

11. Innovative Practices:

- Integration of algorithm visualization tools and coding platforms to enhance conceptual understanding and engagement.
- Use of problem-based learning (PBL) with real-world scenarios to strengthen analytical and design skills.
- Incorporation of mini-projects and case studies to connect data structures with practical applications.
- Encouragement of competitive programming and coding challenges to improve speed and accuracy.
- Adoption of peer learning, code reviews, and flipped classroom techniques to promote active learning and collaboration.
- Practices that lays emphasis on energy-efficient and optimized data structure utilization through comparative analysis of algorithms.

12. Self-Learning Component:

- Students are encouraged to engage in self-paced learning through online coding platforms (NPTEL/SWAYAM), tutorials, and technical resources to strengthen their understanding of algorithms and data structures. (E.g. Programming, *Data Structures And Algorithms Using Python* by Prof. Madhavan Mukund (IIT Madras) - https://nptel.ac.in/courses/106106145)
- Regular practice of coding problems beyond the syllabus helps in improving problem-solving skills and logical thinking.
- Learners can explore advanced topics such as dynamic programming, graph algorithms, and optimization techniques independently.
- Participation in coding competitions, hackathons, and open-source contributions promotes continuous self-improvement.
- This approach fosters lifelong learning habits, enabling students to keep pace with evolving technologies and industry demands.