|  |  |
| --- | --- |
| **dynaBPokhara University**  **Faculty of Science and Technology** | |
| Course Code: CMP 227 (3 Credits) | Full Marks: 100 |
| Course Title: Data Structure and Algorithms(3-0-2) | Pass Marks: 45 |
| Nature of the Course: Theory/Practical | Total Lectures: 48 hours |
| Level: Bachelor | Year: II / Semester: III | Program: Bachelor of Computer Application |

|  |  |  |
| --- | --- | --- |
| **1. Course Description:** | | |
| This course introduces students to fundamental data structures and algorithms. It covers core concepts, analysis techniques, and practical implementation. Students will learn to design efficient algorithms, analyze their performance, and apply them to problem-solving. The course emphasizes hands-on practice through coding assignments and projects. | | |
| **2. General Objectives**: | | |
| The general objectives of this course are as follows:   1. To familiarize students with fundamental data structures (stacks, queues, linked lists, trees, graphs) and algorithms, including sorting and searching techniques. 2. To equip students to analyze algorithm efficiency using time complexity metrics (Best, Worst, and Average case). 3. To enhance problem-solving and critical thinking skills through practical assignments, projects, and the comparison of recursive and iterative methods. 4. To prepare students for industry challenges by integrating modern tools, techniques, and the application of fundamental algorithms to modern technological fields. | | |
| **3.** **Methods of Instructions:** | | |
| * Lecture and discussion * Practical * Demonstration * Presentation | | |
| **4. Course Contents:** | | |
| **Specific Objectives** | **Contents** | |
| **Unit 1: Introduction to Data Structures and Algorithms 7 hours** | | |
| * Describe how data structures and algorithms (DSA) contribute to effective problem-solving. * Compare time complexity and asymptotic notations (Big O, Omega, Theta) to analyze algorithms. * Explore about the impact of dynamic memory allocation on performance. | 1.1. Introduction to Data Structures   1. Data Types and Data Structures   1.1.2 Classification of Data Structures  1.1.3 ADTs (Abstract Data Types)  1.2 Introduction to Algorithms   * + 1. Definition and Characteristic   1.2.2 Need/Importance of DSA  1.3. Algorithm design  Divided and Conquer, Greedy Algorithm, Backtracking, Dynamic Programming  1.4 Algorithm Analysis  1.3.1 Time Complexity  1.3.2 Best, Worst and Average Case  1.3.3 Rate of Growth  1.3.3 Asymptotic Notation (Big O, Omega and Theta) | |
| **Unit 2: Stack, Queues and Recursion 8 hours** | | |
| * Explain the concept of stacks as an ADT. * Explore the role queue types (linear, circular, priority). * Implement the recursive problems. | 2.1. Concept of stacks and Stack ADT  2.1.1 Stack operations  2.1.2 Stacks implementation using arrays  2.1.3 Applications of stacks:  2.1.3.1 Conversion from infix to postfix  2.1.3.2 Evaluation of postfix  2.1.4 Recursion  2.1.4.1 Concept of recursion  2.1.4.2 Recursive vs. iterative solutions  2.1.4.3 Recursive problems: Factorial, Fibonacci sequence, GCD, Tower of Hanoi  2.1.5 Recursion and stack  2.2. Concept of queues and Queue ADT  2.2.1 Queue operations  2.2.2 Queue Implementation using arrays  2.2.3 Types of Queues  2.2.4. Linear and circular  2.2.5 Introduction to priority queues and double ended queue. | |
| **Unit 3: Linked List 8 hours** | | |
| 1. Implementation of about different types of linked lists: Singly Linked List, Doubly Linked List, and Circular Linked List. 2. Describe basic operations in linked lists such as node creation, insertion, and deletion at various positions. 3. Implement stacks and queues using linked lists, with a focus on doubly and circular linked lists. | 3.1 List –Definition and List ADT, array Implementation  3.2 Linked List – Definition and its operations  3.3 Types of Linked Lists:  3.3.1 Singly Linked List  3.3.2 Doubly Linked List  3.3.3 Circular Linked List  3.4 Implementation of Singly Linked Lists:  3.4.1 Node Creation  3.4.2 Node Insertion (Beginning, End, Specified Position)  3.4.3 Node Deletion (Beginning, End, Specified Position)  3.6 Linked List implementation of Stack and Queue | |
| **Unit 4: Tree 8 hours** | | |
| 1. Implement and analyze various tree traversal algorithms for binary trees. 2. Implement binary search trees, including insertion, deletion, and searching operations. 3. Explain the need for balanced trees, search trees, and their applications. | 4.1 Tree concepts and terminology  4.2 Binary trees:  4.2.1 Definition and types  4.2.2 Tree representations (array, linked-based)  4.2.3 Tree Traversal (Pre, Post and In Orders)  4.2.4 Construction of binary tree from traversal  4.2.5 Application of Tree: Huffman Algorithm  4.3 Binary search trees:  4.3.1 Construction of BST  4.3.2 Basic Operation on BST (Insert, delete, search) Node | |
|  | 4.4 Introduction to balanced trees  4.4.1 Need for balanced trees  4.4.2 Definition of AVL (Adelson-Velskii and Landis) trees, insertion, deletion and Rotation operations  4.5 B-tree Definition and application | |
| **Unit 5: Sorting** | | **6 hours** |
| 1. Analyze and implement various sorting algorithms, including simple and advanced techniques. 2. Compare the efficiency of different sorting algorithms using time complexity analysis. 3. Implement the workings of distribution-based sorting algorithms. | 5.1 Definition and Classification of Sorting algorithm (internal, external, stable, unstable)  5.2 Bubble Sort  5.3 Selection Sort  5.4 Insertion Sort  5.5 Merge Sort  5.6 Quick Sort  5.7 Radix Sort  5.8 Heap Sort as priority queues | |
| **Unit 6: Searching Algorithm and Hashing** | | **4 hours** |
| 1. Implement and analyze Binary Search for efficient data retrieval. 2. Apply the fundamentals of Hashing, including Hash Functions and Hash Tables. 3. Explore Collision Resolution Techniques: Separate Chaining, Linear Probing, Quadratic Probing | 6.1Concept of Searching algorithms  6.2 Linear and Binary Search  6.3 Hashing:  6.3.1 Concept of Hashing, Hash functions, Hash Table  6.4 Collision resolution techniques:  6.4.1. Open Hashing: Separate Chaining  6.4.2. Closed addressing : Linear Probing , Quadratic Probing, Double Hashing  6.5. Load Factor and Rehashing | |
| **Unit 7: Graphs** | | **4 hours** |
| 1. Describe graph representations through its types. 2. Master graph traversal techniques (DFS, BFS, topological sort). 3. Apply graph algorithms for finding minimum spanning trees (Kruskal's, Prim's) and shortest paths (Dijkstra's). 4. Apply graph concepts to real-world problems and applications. | 7.1. Graph concepts and representations (adjacency matrix, adjacency list) and types  7.2. Graph traversals:  7.2.1. Depth-first search (DFS)  7.2.2. Breadth-first search (BFS)  7.2.3. Topological Sort  7.3. Minimum Spanning Tree- Kruskal’s Algorithm and Prim’s Algorithm  7.4. Shortest path problems- Dijkstra's algorithm | |
| **5. Practical Work:**  Students should focus on manually implementing these data structures from scratch. Students are encouraged to implement these ADTs without use of Java’s built-in libraries. At the end of practical works, student can identify the performance among the implementation.   1. Implement stack, linear and circular queue using arrays. 2. Implement Infix to postfix conversion using stack 3. Implement recursive algorithms- Factorial, Fibonacci series and Tower of Hanoi 4. Implement singly link list, binary search trees and implement traversal algorithms. 5. Implement graph traversal algorithms (DFS, BFS). 6. Implement sorting algorithms (quick sort, merge sort, heap sort). 7. Implement searching algorithms (sequential, binary search). 8. Implement hash system (separate chaining, liner and quadratic). 9. Implement Dijkstra's algorithm for shortest path. 10. Implement Kruskal's or Prim's algorithm for minimum spanning tree. | | |
| **6. Evaluation System and Students’ Responsibilities:** | | |
| **6.1 Evaluation System:** | | |
| In addition to the formal exam(s) conducted by the Office of the Controller of Examination of Pokhara University, the internal evaluation of a student may consist of class attendance, class participation, quizzes, assignments, presentations, written exams, etc. The tabular presentation of the evaluation system is as follows.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Internal Evaluation** | **Weight** | **Marks** | **External Evaluation** | **Marks** | | **Theory** |  | **30** | **Semester End examination** | **50** | | Attendance / Class Participation | 10% |  | | Assignments | 20% |  | | Project Work/Presentations | 10% |  | | Term Exam | 60% |  | | **Practical** |  | **20** | | Attendance and Lab Participation | 10% |  | | Lab Report | 20% |  | | Lab Examination | 40% |  | | Viva Examination | 30% |  | | **Full marks=50+50** | | | | | | | |
| **6.2 Students’ Responsibilities**: | | |
| To be eligible for the Semester End Examinations, each student must secure at least 45% marks in the internal evaluation with 80% attendance in the class to appear in the Semester End Examination. Failing to obtain such score will be given NOT QUALIFIED (NQ) and the student will not be eligible to appear in the End-Term examinations. Students are advised to attend all the classes and complete all the assignments within the specified time period. If a student does not attend the class(es), it is his/her sole responsibility to cover the topic(s) taught during the period. If a student fails to attend a formal exam, quiz, test, etc. there won’t be any provision for a re-exam. | | |
| **7. Prescribed Books and References:** | | |
| **Text Books**   1. Langsam, Y., Augustin, M.J. and Tanenbaum, A.M: Data Structure Using C and C++, Prentice Hall of India 2. Michael T. Goodrich and Roberto Tamassia *Data Structures and Algorithms in Java, 0-471-73884-0, Fourth Edition, John Wiley & Sons, Inc.* | | |
| **Reference Books**   1. Rowe, G.W.: Introduction to Data Structure and Algorithms with C and C++, Prentice Hall of India 2. Kruse, R. L., & Ryba, A. J. (1998). *Data structures and program design in C++*. Prentice Hall, India.. 3. Brassard, G., & Bratley, P. (1996). *Fundamentals of algorithmic*. Prentice-Hall, India. | | |