

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**WORK INTEGRATED LEARNING PROGRAMMES**

**COURSE HANDOUT**

**Part A: Content Design**

|  |  |
| --- | --- |
| **Course Title** | DATA STRUCTURES AND ALGORITHMS DESIGN |
| **Course No(s)** | SE ZG519/SS ZG519 |
| **Credit Units** | 5 |
| **Course Author** | Febin. A.Vahab |
| **Version No** | 2.0 |
| **Date** | 01/10/2020 |

**Course Description**

|  |
| --- |
| The course covers design, implementation and applications of basic and advanced data structures including trees, graphs, bloom filters. The course also covers algorithm design techniques like greedy, dynamic, map reduce etc. using examples from sorting, searching, graph theory, networking and number theory. The complexity issues are also discussed further. |

**Course Objectives**

* Equip students with the ability to analyse the performance and correctness of algorithms using theoretical foundations and asymptotic notations, preparing them for advanced study and professional application in algorithm design.
* Enable students to proficiently analyse and solve recursive algorithms through the understanding and application of recurrence relations, the Master Theorem, and various methods for solving recurrences.
* Ensure that students gain a deep understanding of both elementary and advanced data structures, including their implementation, applications, and optimization techniques, to solve complex computational problems effectively.
* Provide students with the knowledge and skills to implement and utilize advanced data structures such as hash tables, AVL trees, k-d trees, and Bloom filters, focusing on collision handling, rehashing, and other optimization techniques.
* Train students in employing various algorithm design techniques such as greedy methods, divide and conquer, and dynamic programming to develop efficient solutions for a wide range of computational problems.
* Foster a comprehensive understanding of computational complexity theory, including the classification of problems into P and NP classes, the concept of NP-completeness, and the application of polynomial-time reducibility to significant computational problems.

**Learning Outcomes:**

At the end of the course students must be able to

* Demonstrate the ability to analyse the efficiency of algorithms using asymptotic notation and characterize their run-time complexities. Understand and apply concepts such as best case, average case, and worst case scenarios, as well as the correctness of algorithms.
* Develop the skills to analyse recursive algorithms using recurrence relations and the Master Theorem. Solve recurrence relations using methods such as substitution and recursion trees to specify the runtime of recursive algorithms.
* Implement and utilize various data structures including stacks, queues, lists, trees (binary trees, AVL trees, k-d trees), heaps, and graphs. Analyse and optimize the operations associated with these data structures for different applications.
* Understand and implement advanced data structures such as hash tables, including collision handling methods, and Bloom filters. Apply these data structures to solve complex problems efficiently.
* Employ algorithm design techniques such as the greedy method, divide and conquer, and dynamic programming to solve various computational problems. Implement algorithms for problems like the knapsack problem, shortest path problems, and task scheduling.
* Understand the fundamental concepts of computational complexity, including P and NP classes, NP-completeness, and polynomial-time reducibility. Apply these concepts to problems such as CNF SAT and the Clique problem, and comprehend the implications of the Cook-Levin theorem.

**Text Book(s)**

|  |  |
| --- | --- |
| **No** | **Author(s), Title, Edition, Publishing House** |
| T1 | Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition) |

**Reference Book(s) & other resources**

|  |  |
| --- | --- |
| **No** | **Author(s), Title, Edition, Publishing House** |
| R1 | Introduction to Algorithms, TH Cormen, CE Leiserson, RL Rivest, C Stein, Third Ed,  2009, PHI |
| R2 | Data Structures, Algorithms and Applications in Java, Sartaj Sahni, Second Ed, 2005, Universities Press |

**CONTENT STRUCTURE**

|  |  |  |
| --- | --- | --- |
| **No** | **Title of the Module** | **References** |
| M1 | **Analyzing Algorithms**  1.1. Theoretical Foundation  1.1.1. Algorithms and it’s Specification  1.1.2. Random Access Machine Model  1.1.3. Notion of best case, average case and worst case  1.1.4. Notion of Algorithm Correctness  1.2. Characterizing Run Time  1.2.1. Use of asymptotic notation  1.2.2. Big-Oh, Omega and Theta Notations  1.3. Analyzing Recursive Algorithms  1.3.1. Recurrence relations  1.3.2. Specifying runtime of recursive algorithms  1.3.3. Master Theorem  1.3.4 Solving Recurrences: Substitution Method,  Recursion Tree Method | T1: 1.1, 1.2  T1:1.1.4  R1: 4.3,4.4,4.5 |
| M2 | **Elementary Data Structures**   1. Stacks ADT , Implementation and Applications 2. Queues ADT , Implementation and Applications 3. Amortized Analysis – Stack, Queue operations- Aggregate Method 4. List ADT , Implementation and Applications | R1:10.1  R1:17.1  R1:10.2 |
| M3 | **Non-Linear Data Structures**  3.1. Trees  3.1.1. Terms and Definition  3.1.2. Tree ADT  3.1.3. Applications  3.2. Binary Trees  3.2.1. Properties  3.2.2. Representations (Array Based and Linked Structure)  3.2.3. Binary Tree traversal (In Order, Pre Order, Post Order)  3.2.4. Applications  3.3. Heaps  3.3.1. Definition and Properties  3.3.2. Representations (Array Based and Linked)  3.3.3. Insertion and deletion of elements  3.3.4. Heap sort  3.3.5. Priority Queue  3.4. Graphs  3.4.1. Terms and Definitions  3.4.2. Properties  3.4.3. Representations (Edge List, Adjacency list, Adjacency Matrix)  3.4.4. Graph Traversals (Depth First and Breadth First Search )  3.5.5. Applications  3.5. Directed Graph and Reachability-  Floyd-Warshall’s Transitive Closure | T1: 2.3  R2:6  R1: 22.1, 22.2,22.3  R1:25.2 |
| M4 | **Dictionaries**  4.1. Dictionary ADT , Applications  4.2. Hash Tables  4.2.1. Notion of Hashing and Collision  4.2.2. Methods for Collision Handling  4.2.2.1. Separate Chaining  4.2.2.2. Notion of Load Factor  4.2.2.3. Rehashing  4.2.2.4. Open Addressing [ Linear & Quadratic Probing, Double Hash]  4.2.2.5. Applications  4.3. Universal Hashing  4.4. Introduction to Bloom Filters, Applications  4.5. Binary Search Tree  4.5.1. BST Operations  4.5.2. Applications  4.6. AVL trees  4.7. Rank and Range Queries, Performance  4.6 k-d Trees  4.6.1 Representation  4.6.2 Range and NN Queries | R2:11  [Bloom Filter](https://www.eecs.harvard.edu/~michaelm/NEWWORK/postscripts/BloomFilterSurvey.pdf)  R1: 12  T1:3.1  T1:3.2  T1:12.1  T1:12.3.2 |
| M5 | **Algorithm Design Techniques**  5.1. Greedy Method  5.1.1. Design Principles and Strategy  5.1.2. Fractional Knapsack Problem  5.1.3. Minimum Spanning Tree  5.1.4. Shortest Path Problem - Djikstra’s Algorithm  5.1.5. Task Scheduling Problem  5.2. Divide and Conquer  5.2.1. Design Principles and Strategy  5.2.2. Integer Multiplication Problem  5.2.3. Merge Sort  5.2.4. QuickSort  5.3. Dynamic Programming  5.3.1. Design Principles and Strategy  5.3.2. Matrix Chain Product Problem  5.3.3. All-pairs Shortest Path Problem  5.3.4. 0/1 Knapsack Problem | T1: 5.1, 7.3,7.1.1  T1: 5.2.2, 4.1,4.3  T1: 5.3,7.2 |
| M6 | **Complexity Classes**  6.1. Definition of P and NP classes and examples  6.2. Understanding NP-Completeness: CNF SAT  6.3. Cook-Levin theorem  6.4. Polynomial time Reducibility:  6.4.1 CNF SAT  6.4.2 Clique | T1: 13 |

**Part B: Session Plan**

|  |  |
| --- | --- |
| **Academic Term** | **First Semester 2024-2025** |
| **Course Title** | **DATA STRUCTURES AND ALGORITHMS DESIGN** |
| **Course No** | **SE ZG519/SS ZG519** |
| **Lead Instructor** | Rajib Ranjan Maiti |

***SESSION CONTENTS***

|  |  |  |
| --- | --- | --- |
| **Session**  **(#)** | **List of Topic Title**  **(from content structure in Course Handout)** | **Text/Ref Book** |
| 1 | **Analyzing Algorithms**  **Theoretical Foundation**   * Algorithms and it’s Specification * Random Access Machine Model * Notion of best case, average case and worst case * Notion of Algorithm Correctness | T1: 1.1, 1.2 |
| 2 | **Analyzing Algorithms (Continued…)**  **Characterizing Run Time**   * Use of asymptotic notation * Big-Oh, Omega and Theta Notations   **Analyzing Recursive Algorithms**   * Recurrence relations * Specifying runtime of recursive algorithms * Master Theorem | T1:1.1.4  R1: 4.3,4.4,4.5 |
| 3 | **Elementary Data Structures**   * Stacks ADT , Implementation and Applications * Queues ADT , Implementation and Applications * Amortized Analysis -Stack, Queue operations-Aggregate Method * List ADT , Implementation and Applications | R1:10.1  R1:17.1  R1:10.2 |
| 4 | **Non-Linear Data Structures**  **Trees**   * Terms and Definition * Tree ADT * Applications   **Binary Trees**   * Properties * Representations (Array Based and Linked Structure) * Binary Tree traversal (In Order, Pre Order,Post Order) * Applications | T1: 2.3 |
| 5 | **Heaps**   * Definition and Properties * Representations (Array Based and Linked) * Insertion and deletion of elements * Heap sort * Priority Queue | R2:6 |
| 6 | **Graphs**   * Terms and Definitions * Properties * Representations (Edge List, Adjacency list, Adjacency Matrix) * Graph Traversals (Depth First and Breadth First Search ) * Applications | R1: 22.1, 22.2,22.3 |
| 7 | * Directed Graph and Reachability-Floyd-Warshall’s Transitive Closure   **Dictionaries**   * Dictionary ADT , Applications * Hash Tables * Notion of Hashing and Collision   **Methods for Collision Handling**   * Separate Chaining * Notion of Load Factor * Rehashing | R1:25.2  R2:11 |
| 8 | **Methods for Collision Handling (Continued…)**   * Open Addressing [ Linear &Quadratic Probing, Double Hash] * Applications   Universal Hashing  Introduction to Bloom Filters, Applications | R2:11 |
| 9 | **Binary Search Tree**   * BST Operations * Applications * AVL trees | T1:3.1,3.2 |
| 10 | * Rank and Range Queries, Performance   **k-d Trees**   * Representation * Range and NN Queries | T1:12.1  T1:12.3.2 |
| 11. | **Algorithm Design Techniques**  **Greedy Method**   * Design Principles and Strategy * Fractional Knapsack Problem | T1: 5.1 |
| 12 | **Greedy Method (Continued…)**   * Minimum Spanning Tree * Shortest Path Problem - Djikstra’s Algorithm | T1: 7.3,7.1.1 |
| 13 | **Divide and Conquer**   * Design Principles and Strategy * Integer Multiplication Problem * Merge Sort | T1: 5.2.2, 4.1 |
| 14 | **Dynamic Programming**   * Design Principles and Strategy * Matrix Chain Product Problem | T1: 5.3 |
| 15 | * All-pairs Shortest Path Problem   **Complexity Classes**   * Definition of P and NP classes and examples | T1: 7.2  T1: 13 |
| 16 | **Complexity Classes (Continued…)**   * Understanding NP-Completeness: CNF SAT * Cook-Levin theorem   Polynomial time Reducibility:   * CNF SAT * Clique | T1: 13 |

***TUTORIAL SESSION CONTENTS***

*# There should be 4 tutorial sessions planned ,each with a duration of 1.5 hours.*

|  |  |  |
| --- | --- | --- |
| **Webinar(#)** | **Topic** | **References** |
| 1 | Solving recurrence equations–Substitution Method, Recursion Tree Method | R1:4.3,4.4 |
| 2 | Greedy Method: Task Scheduling Problem | T1:5.1.2 |
| 3 | Divide and Conquer: Quick Sort | T1:4.3 |
| 4 | Dynamic Programming: 0/1 Knapsack Problem | T1:5.3.3. |

***Select Topics and Case Studies from business for experiential learning***

|  |  |  |
| --- | --- | --- |
| **Topic No.** | **Select Topics in Syllabus for experiential learning** | **Access URL** |
| TBD | TBD | TBD |

**Evaluation Scheme**:

Legend: EC = Evaluation Component; AN = After Noon Session; FN = Fore Noon Session

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Type | Duration | Weight | Day, Date, Session, Time |
| EC-1 | Quiz-I/ Assignment-I | Online | - | 10% | September 1-10, 2024 |
| Quiz-II |  |  | 10% | October 10-20, 2024 |
| Quiz-III/ Assignment-II |  |  | 10% | November 1-10, 2024 |
| EC-2 | Mid-Semester Test | Closed Book | 2 hours | 30% | Saturday, 21/09/2024 (FN) |
| EC-3 | Comprehensive Exam | Open Book | 2 ½ hours | 40% | Saturday, 30/11/2024 (FN) |

***Note*** *- Evaluation components can be tailored depending on the proposed model.*

***Important Information***

Syllabus for Mid-Semester Test (Closed Book): Topics in Weeks 1-7

Syllabus for Comprehensive Exam (Open Book): All topics given in plan of study

Evaluation Guidelines:

1. EC-1 consists of either two Assignments or three Quizzes. Announcements regarding the same will be made in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
3. For Open Book exams: Use of prescribed and reference text books, in original (not photocopies) is permitted. Class notes/slides as reference material in filed or bound form is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam. The genuineness of the reason for absence in the Regular Exam shall be assessed prior to giving permission to appear for the Make-up Exam. Make-Up Test/Exam will be conducted only at selected exam centers on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self-study schedule as given in the course handout, attend the lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.