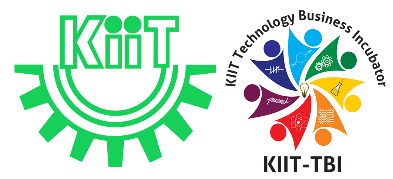
****

**School of Computer Engineering**

**Kalinga Institute of Industrial Technology (KIIT)**

**Deemed to be University**

**Bhubaneswar-751024**

**Day-wise Lesson Handouts**

**Design and Analysis of Algorithms – CS 2012 (L-T-P-Cr: 2-1-0-3)**

**Semester:** 5th

**Discipline:** B.Tech. (CS, IT, CSSE, CSCE)

**Session:** Autumn 2023

**Instructor:**

**Name :** Lipika Mohanty

**Chamber :** Faculty Chamber 8,Ground Floor, Block-B, Campus-15

**Contact Number :** +91-9643535698

**Email :** [lipika.mohantyfcs@kiit.ac.in](mailto:dayal.beherafcs@kiit.ac.in)

**Time Table:**

**CSE 37**

**Tuesday: 3pm -4pm( A-LH 004, Block-A)**

**Wednesday: 8am-9am (A-LH 104, Block-A)**

**Thursday: 10am-11am (A-LH 009, Block-A)**

**Course Objective:**

Design and Analysis of Algorithms is a core course, to 3rd year B.Tech (CS, CSCE, and IT) students. The objective of this course is

* To understand the importance of algorithm
* To analyze the complexity of an algorithm in terms of time and space complexities
* To understand various problem solving techniques
* To learn about amortized analysis of algorithms
* To design and implement various programming paradigms and its complexity

**Day-wise Lesson Handouts:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Lecture No.** | **Topics** | |
| **Week - 1** | 1 | Concepts in algorithm analysis & design. It’s motivation. Difference between Algorithm and Programs, Characteristics of algorithms, Performance Analysis: Space and Time Complexity | |
| 2 | Pseudocode Conventions, Analysis of Linear Search by step count method, Analysis of Insertion Sort by step count method (Incremental Approach), Best-case, Worst-case and Average-case Analysis. | |
| 3 | Amortized Complexity, Growth of functions, Asymptotic Notations () | |
| **Week - 2** | 4 | Asymptotic Notations (), Asymptotic Notation Properties | |
| 5 | Recurrences, solving recurrences using Iterative, Substitution method | |
| 6 | Divide-and-Conquer Approach, Merge Sort Algorithm | |
| **Week – 3** | 7 | Analyzing divide-and-conquer algorithms, Analysis of merge-sort | |
| 8 | Solving recurrences using Recursion-tree method | |
| 9 | Solving recurrences using Master method | |
| **Week - 4** | 10 | Change of Variable method, More recurrence examples | |
| 11 | Binary Search and its complexity analysis | |
| 12 | Quicksort Algorithm, Performance of quicksort (Worst-case, Best-case and Balanced Partitioning) | |
| **Week - 5** | 13 | | Randomized Algorithms: An informal description |
| 14 | | Randomized Quicksort and its complexity analysis |
| 15 | | Introduction to Heap, Maintaining the heap property, Building a heap |
| **Week - 6** | 16 | | Heap-sort algorithm and its complexity analysis |
| 17 | | Application of Heap: Priority Queues |
| 18 | | Overview of Greedy paradigm, An Activity Selection Problem: Recursive, Iterative greedy algorithm |
| **Week - 7** | 19 | | Elements of greedy strategy |
| 20 | | Knapsack Problem, Difference between Fractional Knapsack and 0/1 Knapsack |
| 21 | | Huffman codes, Constructing a Huffman code (tree) |
| **Week - 8** | 22 | | Overview of Dynamic Programming paradigm, Divide and Conquer vs Dynamic Programming, Greedy vs Dynamic Programming |
| 23 | | Matrix Chain Multiplication |
| 24 | | Elements of dynamic programming, Tabulation vs Memoization |
| **Week - 9** | 25 | | Longest Common Subsequence |
| 26 | | Dis-joint Set Data Structure |
| 27 | | Representation Of Graph and its terminology |
| **Week - 10** | 28 | | Graph Traversals: BFS |
| 29 | | Graph Traversals: DFS |
| 30 | | Single Source Shortest Path: Dijkstra’s Algorithm |
| **Week - 11** | 31 | | Single Source Shortest Path: Bellman-Ford Algorithm |
| 32 | | All Pair Shortest Path: Floyd-Warshall Algorithm |
| 33 | | Introduction to Spanning Tree |
| **Week - 12** | 34 | | Minimum Cost Spanning Tree: Kruskal’s Algorithm |
| 35 | | Minimum Cost Spanning Tree: Prim’s Algorithm |
| 36 | | Complexity Classes: P, NP, NP-Hard and NP-Complete |
| **Week - 13** | 37 | | NP-Complete Problems: The Clique problems |
| 38 | | Revisions |

**Course Outcome:**

|  |  |
| --- | --- |
| CO1: | analyse the asymptotic performance of algorithms |
| CO2: | understand different algorithm design techniques |
| CO3: | apply important algorithm design paradigms and methods of analysis |
| CO4: | demonstrate familiarity with major algorithms and data structures |
| CO5: | modify existing algorithms to apply in common engineering design situations. |
| CO6: | understand different classes of problems P, NP, NP-Complete and NP-Hard. |

**Text books:**

* T. H. Coreman, C. E. Leiserson, R. L. Rivest, C. Stein, “Introduction to Algorithms”, PHI.
* E. Harwitz, S. Sahani, S. Rajsekharan, Galgotia “Fundamentals of Computer Algorithms”, Galgotia Publication.

**Reference books:**

* J. Kleinberg, E. Tardos, “Algorithms Design”, Pearson International 1st Edition.
* Michael Goodrich, Roberto Tamassia, “Algorithm Design: Foundations, Analysis & Internet Examples”, John Wiley & Sons.

**Grading Policy:**

* Activities (6 No.s) : **30 Marks**
* Mid-semester exam : **20 Marks**
* End-semester exam : **50 Marks**