|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Design and Analysis of Algorithms 2017 curriculum** | | | | | |
| **Credit Hours:** | **3** | **Prerequisites:** | **Data Structures and**  **Algorithms** | | |
| **Course Learning Outcomes (CLOs):** | | | |  |  |
| **At the end of the course the students will be able to:** | | | | **Domain** | **BT**  **Level\*** |
| 1. Explain what is meant by “best”, “expected”, and “worst”  case behavior of an algorithm | | | |  |  |
| 2. Identify the characteristics of data and/or other conditions  or assumptions that lead to different behaviors. | | | |  |  |
| 3. Determine informally the time and space complexity of  simple algorithms | | | |  |  |
| 4. List and contrast standard complexity classes | | | |  |  |
| 5. Use big O, Omega, Theta notation formally to give  asymptotic upper bounds on time and space complexity of algorithms | | | |  |  |
| 6. Use of the strategies(brute-force, greedy, divide-and- conquer, and dynamic programming) to solve an  appropriate problem | | | |  |  |
| 7. Solve problems using graph algorithms, including single-  source and all-pairs shortest paths, and at least one minimum spanning tree algorithm | | | |  |  |
| 8. Trace and/or implement a string-matching algorithm | | | |  |  |
| \* BT= Bloom’s Taxonomy, C=Cognitive domain, P=Psychomotor domain, A=  Affective domain | | | | | |
| Course Content: | | | | | |
| Introduction; role of algorithms in computing, Analysis on nature of input and size of input Asymptotic notations; Big-O, Big Ω, Big Θ, little-o, little-ω, Sorting Algorithm analysis, loop invariants, Recursion and recurrence relations; Algorithm Design Techniques, Brute Force Approach, Divide-and-conquer approach; Merge, Quick Sort, Greedy approach; Dynamic programming; Elements of Dynamic Programming, Search trees; Heaps; Hashing; Graph algorithms, shortest paths, sparse graphs, String matching;  Introduction to complexity classes; | | | | | |
| Teaching Methodology: | | | | | |
| Lectures, Written Assignments, Semester Project. | | | | | |
| Course Assessment: | | | | | |
| Sessional Exam, Home Assignments, Quizzes, Project, Final Exam | | | | | |
| Reference Materials: | | | | | |
| 1. Introduction to Algorithms (3rd edition) by Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein 2. Algorithm Design, (1st edition, 2013/2014), Jon Kleinberg, Eva Tardos, 3. Algorithms, (4th edition, 2011), Robert Sedgewick, Kevin Wayne | | | | | |

**2023 curriculum**

Course Name: **Analysis of Algorithms**

Credit Hours: 3 (3-0)

Contact Hours: 3-0

Pre-requisites: Data Structures

Course Introduction:

Detailed study of the basic notions of the design of algorithms and

the underlying data structures. Several measures of complexity are

introduced. Emphasis on the structure, complexity, and efficiency of

algorithms.

CLO No. Course Learning Outcomes Bloom Taxonomy

CLO-1 Explain what is meant by “best”, “expected”, and “worst” case

behavior of an algorithm

CLO-2 Identify the characteristics of data and/or other conditions

or assumptions that lead to different behaviors.

CLO-3 Determine informally the time and space complexity of simple

algorithms

CLO-4 List and contrast standard complexity classes

CLO-5 Use big O, Omega, Theta notation formally to give asymptotic

upper bounds on time and space complexity of algorithms

CLO-6 Use of the strategies(brute-force, greedy, divide-and- conquer,

and dynamic programming) to solve an appropriate problem

CLO-7 Solve problems using graph algorithms, including single- source

and all-pairs shortest paths, and at least one minimum spanning tree

algorithm

CLO-8 Trace and/or implement a string-matching algorithm

Course Outline:

Introduction; role of algorithms in computing, Analysis on nature of

input and size of input Asymptotic notations; Big-O, Big Ω, Big Θ,

little-o, little-ω, Sorting Algorithm analysis, loop invariants,

Recursion and recurrence relations; Algorithm Design Techniques, Brute

Force Approach, Divide-and-conquer approach; Merge, Quick Sort, Greedy

approach; Dynamic programming; Elements of Dynamic Programming, Search

trees; Heaps; Hashing; Graph algorithms, shortest paths, sparse

graphs, String matching; Introduction to complexity classes.

Reference Materials: (or use any other standard and latest books)

1. Introduction to Algorithms (3rd edition) by Thomas H. Corman,

Charles E. Leiserson, Ronald L. Rivest and Clifford Stein

2. Algorithm Design, (1st edition, 2013/2014), Jon Kleinberg, Eva

Tardos,

1. Algorithms, (4th edition, 2011), Robert Sedgewick, Kevin Wayne

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Theory of Automata 2017 curriculum** | | | | | |
| **Credit Hours:** | 3 | **Prerequisites:** | None | | |
| **Course Learning Outcomes (CLOs):** | | | |  |  |
| At the end of the course the students will be able to: | | | | **Domain** | **BT**  **Level\*** |
| 1. Explain and manipulate the different concepts in automata theory and formal languages such as formal proofs,  automata, regular expressions, Turing machines etc; | | | |  |  |
| 2. Prove properties of languages, grammars and automata with  rigorously formal mathematical methods | | | |  |  |
| 3. Design of automata, RE and CFG | | | |  |  |
| 4. Transform between equivalent NFAs, DFAs and REs | | | |  |  |
| 5. Define Turing machines performing simple tasks. | | | |  |  |
| 6. Differentiate and manipulate formal descriptions of  languages, automata and grammars with focus on regular  and context-free languages, finite automata and regular expressions. | | | |  |  |
| \* BT= Bloom’s Taxonomy, C=Cognitive domain, P=Psychomotor domain, A=  Affective domain | | | | | |
| Course Content: | | | | | |
| Finite State Models: Language definitions preliminaries, Regular expressions/Regular languages, Finite automata (FAs), Transition graphs (TGs), NFAs, Kleene’s theorem, Transducers (automata with output), Pumping lemma and non-regular language Grammars and PDA: CFGs, Derivations, derivation trees and ambiguity, Simplifying CFLs, Normal form grammars and parsing, Decidability, Context sensitive languages, grammars and linear bounded automata (LBA), Chomsky’s hierarchy of grammars Turing Machines Theory: Turing machines, Post machine, Variations on TM, TM  encoding, Universal Turing Machine, Defining Computers by TMs. | | | | | |
| Teaching Methodology: | | | | | |
| Lectures, Written Assignments, Practical labs, Semester Project, Presentations | | | | | |
| Course Assessment: | | | | | |
| Sessional Exam, Home Assignments, Quizzes, Project, Presentations, Final Exam | | | | | |
| Reference Materials: | | | | | |
| 1. Introduction to computer theory, Daniel I. A. Cohen, 2nd Edition 2. Automata, Computability and Complexity: Theory and Applications, by Elaine Rich, 2011 3. An Introduction to Formal Languages and Automata, by Peter Linz, 4th edition, Jones & Bartlett Publishers, 2006 4. Theory of Automata, Formal Languages and Computation, by S. P. Eugene, Kavier, 2005, New Age Publishers | | | | | |