| **Subject Code**  **20CST-311** | **Design and Analysis of Algorithms** | **L** | **T** | **P** | **S** | **C** |
| --- | --- | --- | --- | --- | --- | --- |
| **Total Contact Hours : 45 Hours** | **3** | **0** | **0** | **0** | **3** |
|  | | | | | |
| **Pre-requisites** | Studied Data Structures, C/C++ in Previous semesters | | | | | |
| **Co-requisites** | **--** | | | | | |
| **Anti-Requisites** | **--** | | | | | |

**DAA Theory syllabus :**

**Course Objectives:**

* To understand meaning and characteristics of algorithms
* To study different algorithm design techniques.
* To implement different algorithm design techniques for solving engineering and related problems.

**Course Outcomes:**

* Apply the knowledge of efficiency evaluation of algorithm with respect to time and space complexity of algorithms.
* Describe the various algorithm development approaches to solve the problems like divide and conquer, graph based, tree based, etc.
* Evaluate the complexity of the algorithms to evaluate the efficiency and effectiveness as greedy strategy, dynamic programming strategy and will able to gain knowledge about backtracking, branch and bound and string matching techniques to deal with some hard problems.
* Analyze the various classes for complex problems like P, NP, and NP-Complete and Correlate existing algorithms to improve efficiency.
* Analyze the various techniques for algorithm design and apply the knowledge to solve complex engineering problems.

**Contents of the Syllabus**

**UNIT-1 [15h]**

**Chapter-1 (Algorithms and Program Performance)**

Designing and analyzing algorithms, Time and Space complexity, Average and worst case Analysis, Asymptotic notations, recurrence equations and their solution: substitution method, recursion-tree method, master method.

**Chapter-2 (Review of Data Structures)**

Arrays, Stacks, Queues, Pointers, Linked Lists (One –way, Two-way and circular Two-way), Hashing, Trees (BST, B Tree, balanced trees (AVL, Red black trees)), Heaps, Graphs

**Chapter-3 (Sorting algorithm)**

Sorting in linear time: counting sort, radix sort, bucket sort

**UNIT-II [15h]**

**Chapter-4 (Divide and conquer & Greedy algorithms)**

Divide and conquer: The General method, Binary search, Finding maximum and minimum of a sequence of numbers, 2 way Merge sort, Quick sort, Selection sort, Strassen’s matrix multiplication.

Greedy algorithms: The general method, Fractional Knapsack problem, Minimum cost spanning tree: Prim’s Algorithm, Kruskal Algorithm; Huffman coding, Optimal merge patterns.

**Chapter-5 (Dynamic programming)**

The general method, 0/1 knapsack, Subset Sum problem, Change making problem, optimal binary search tree, Matrix-chain Multiplication, Longest common Subsequence Problem, Travelling salesman problem. Comparison of Divide & Conquer and Dynamic Programming techniques.

**Chapter-6 (Backtracking & Branch and Bound)**

Backtracking: The general method, N-queen’s problem, sum-of-subsets, Hamiltonian cycles.

Branch and Bound: Branch and Bound method, 0/1 Knapsack problem, Travelling salesperson problem.

**UNIT-III [15h]**

**Chapter-7 (Graph Algorithms)**

Representation of Graphs, Depth First Search, Breadth First search, Topological sort, Single source shortest path: Dijkstra Algorithm & Bellman Ford Algorithm. All-pair shortest paths: Floyd Warshall Algorithm, Minimum Spanning Tree: Sollin’s algorithm.

**Chapter-8 (Computational complexity)**

Basic concepts, P and NP-classes, proof of NP-hard and NP-completeness.

**Chapter-9 (Miscellaneous topics)**

Euclid Algorithm for GCD of 2 numbers, modulo arithmetic, Chinese remainder theorem, string manipulation/matching algorithms: Rabin Karp algorithm, KMP (Knuth-Morris-Pratt) algorithm, Boyer-Moore algorithm; Convex Hull.

**TEXT BOOKS**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
2. Horowitz, Sahni and Rajasekaran, “*Fundamentals of ComputerAlgorithms”*, University Press (India), 2nd edition.

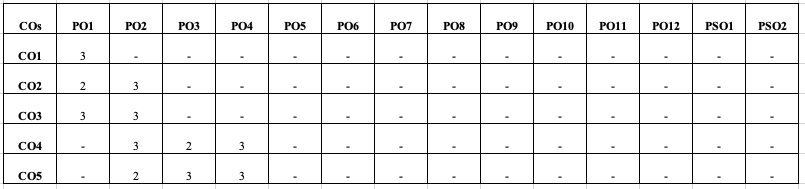
**REFERENCE BOOKS**

1. Tanenbaum, Augenstein, &Langsam, “*Data Structures using C and C++*”, Prentice Hall of India.
2. Brassard, Bratley, “*Fundamentals of Algorithms*”, Prentice Hall of India.
3. Knuth “*The Art of Computer Programming, Volume 1: Fundamental Algorithms*” (Addison-Wesley, Third Edition).
4. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.
5. Kruse, “*Data Structures & Program Design*”, Prentice Hall of India.
6. Aho, Haperoft and Ullman, ”*The Design and analysis of Computer Algorithms*”, Pearson

Education India.

# Mode of Evaluation: The performance of students is evaluated as follows:

|  | **Theory** | |
| --- | --- | --- |
| **Components** | **Continuous Internal Assessment (CAE)** | **Semester End Examination (SEE)** |
| **Marks** | **40** | **60** |
| **Total Marks** | **100** | |



**DAA Lab Syllabus :**

| **Subject Code**  **20CSP-312** | **Design & Analysis of Algorithms Lab** | **L** | **T** | **P** | **S** | **C** |
| --- | --- | --- | --- | --- | --- | --- |
| **Total Contact Hours : 45 Hours** | **0** | **0** | **2** | **0** | **1** |
|  | | | | | |
| **Pre-requisites** | **Basics of Computers and C/C++** | | | | | |
| **Co-requisites** | **--** | | | | | |
| **Anti-Requisites** | **--** | | | | | |

**Course Objectives:**

* To understand the meaning and characteristics of algorithms.
* To study different algorithms' designs.
* Implementing various algorithm design techniques for engineering and related problems.

**Course Outcomes:**

* Apply the knowledge of algorithm design techniques to solve the problems of searching, sorting, and graph algorithms.
* Design the algorithm using advanced techniques for solving complex problems with real-life examples.
* Develop the solution to a real-time problem using various tools like flowcharts, algorithms, programs, etc.
* Utilize modern engineering tools for algorithm techniques to implement algorithms for complex engineering problems like divide and conquer, greedy approach, etc.
* Develop algorithms to solve real-time problems like finding the shortest path, and will be able to see function in multi-disciplinary teams through mini projects based on various problems.

**List of Experiments**

* + 1. **List of Practical’s (Graded)**

**UNIT-I**

1. Calculate the greatest common divisor (GCD) of two numbers using code and analysis.
2. The code implements a power function in O(logn) time.
3. In O(n) time complexity, find the frequency of elements in a given array.
4. (i) Code for inserting and removing elements at the start and end of a doubly and circular linked list.

         (ii) Using templates, write code to push and pop elements, check Isempty and Isfull, and return the top element in stacks.

**UNIT-II**

1. Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming.
2. To implement a subset-sum problem using dynamic programming
3. Code to implement 0-1 Knapsack using Dynamic Programming

**UNIT-III**

1. Code and analyze to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, or (ii) to find a path from source to goal in a maze.
2. Code and analyze to find the shortest paths in a graph with positive edge weights using Dijkstra’s algorithm.
3. Code and Analyze to find all occurrences of a pattern P in a given string S.

**I.**  **List of Practical’s (Non Graded) : Additional Programs**

1. Code to Insert and Delete an element at the beginning and end in Singly Linked List.

2. Code for enqueue, dequeue, Isfull and Isempty operation in queues using templates.

3. Code and analyze to sort an array of integers using Quick sort.

4. Code and analyze to sort an array of integers using Merge sort

5. Code to find the height of a Binary tree with Analysis

6. Code to find the ignored successor in an Binary Search Trees with complexity Analysis.

7. To implement maximum and minimum problem using divide and conquer strategy.

8. To implement binary search using divide and conquer strategy.

9. To implement LCS Problem using Dynamic Programming.

10. Code and analyze to do a breadth-first search (BFS) on an undirected graph. Implementing an application of BFS such as (i) to find connected components of an undirected graph, OR (ii) to check whether a given graph is bipartite.

11. Code and analyze to find shortest paths in a graph with arbitrary edge weights using Bellman-Ford algorithm.

12. To implement KMP (Knuth-Morris-Pratt) algorithm.

13. To implement Prim’s algorithm for minimum spanning tree.

14. To implement Kruskal’s algorithm for minimum spanning tree.

**TEXT BOOKS**

* 1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
  2. Horowitz, Sahni and Rajasekaran, “*Fundamentals of ComputerAlgorithms”*, University Press (India), 2nd edition.

**REFERENCE BOOKS**

1. Tanenbaum, Augenstein, &Langsam, “*Data Structures using C and C++*”, Prentice Hall of India.
2. Brassard, Bratley, “*Fundamentals of Algorithms*”, Prentice Hall of India.
3. Knuth “*The Art of Computer Programming, Volume 1: Fundamental Algorithms*” (Addison-Wesley, Third Edition).
4. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.
5. Kruse, “*Data Structures & Program Design*”, Prentice Hall of India.
6. Aho, Haperoft and Ullman, ”*The Design and analysis of Computer Algorithms*”, Pearson

Education India.

# Mode of Evaluation: The performance of students is evaluated as follows:

|  | **Theory** | |
| --- | --- | --- |
| **Components** | **Continuous Internal Assessment (CAE)** | **Semester End Examination (SEE)** |
| **Marks** | **40** | **60** |
| **Total Marks** | **100** | |

