SEMESTER-5

DEPARTMENT OF COMPUTER SCIENCE
[UG Programme for Bachelor in Computer Science (Honours)]

DISCIPLINE SPECIFIC CORE COURSE - 13 (DSC-13) : Algorithms and Advanced Data Structures

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course
		Lecture	Tutorial	Practical/ Practice		
DSC 13 Algorithms and Advanced Data Structures	**4**	**3**	**0**	**1**	Pass in Class XII	DSC 07 Data Structures with C++, DSC 10 Design and Analysis of Algorithms

Learning Objectives

This course is designed to build upon the fundamentals in data structures and algorithm design and gain exposure to more data structures and algorithms for new problems.

Learning outcomes

On successful completion of the course, students will be able to:

- Comprehend and use data structures for lists.
- Use hash tables for dictionaries.
- Comprehend and use data structures and algorithms for string matching.
- Apply disk based data structures.
- Implement and analyze advanced data structures and algorithms for graphs.
- Describe the purpose of randomization in data structures and algorithms.

Unit 1 (4 hours)

List and Iterator ADTs: Vectors, Lists, Sequences

Unit 2 (6 hours)

Hash Tables, Dictionaries: Hash Functions, Collision resolution schemes.

Unit 3 (8 hours)

Strings: String Matching: KMP algorithm; Tries: Standard Tries, Compressed Tries, Suffix Tries, Search Engines

Unit 4 (8 hours)

More on Trees: 2-4 Trees, B Trees

Unit 5 (8 hours)

More on Graphs: Bellman Ford Algorithm, Union Find Data Structures - application Kruskal's algorithm

Unit 6 (6 hours)

Randomization: Randomized Quicksort, Randomized Select, Skip lists

Unit 7 (5 hours)

Network Flows: Ford Fulkerson algorithm for max flow problem.

Essential/recommended readings

1. Goodrich, M.T, Tamassia, R., & Mount, D. *Data Structures and Algorithms Analysis in C++*, 2nd edition, Wiley, 2011.
2. Cormen, T.H., Leiserson, C.E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 4th edition, Prentice Hall of India, 2022.
3. Kleinberg, J., Tardos, E. *Algorithm Design*, 1st edition, Pearson, 2013.
4. Drozdek, A. *Data Structures and Algorithms in C++*, 4th edition, Cengage Learning. 2012.

Practical List : (30 Hours)

Practical exercises such as

1. Write a program to sort the elements of an array using Randomized Quick sort (the program should report the number of comparisons).
2. Write a program to find the ith smallest element of an array using Randomized Select.
3. Write a program to determine the minimum spanning tree of a graph using Kruskal's algorithm.
4. Write a program to implement the Bellman Ford algorithm to find the shortest paths from a given source node to all other nodes in a graph.

5. Write a program to implement a B-Tree.
6. Write a program to implement the Tree Data structure, which supports the following operations:
   I. Insert
   II. Search
7. Write a program to search a pattern in a given text using the KMP algorithm.
8. Write a program to implement a Suffix tree.

DISCIPLINE SPECIFIC CORE COURSE – 14 (DSC-14): Theory of Computation

Credit distribution, Eligibility and Prerequisites of the Course

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course
		Lecture	Tutorial	Practical/ Practice		
DSC 14 Theory of Computation	**4**	**3**	**0**	**1**	Pass in Class XII	DSC04 Object Oriented Programming with C++ / GE1a Programming using C++ /A course in C/C++ at plus 2 level

Learning Objectives

This course introduces formal models of computation, namely, finite automaton, pushdown automaton, and Turing machine; and their relationships with formal languages. make students aware of the notion of computation using abstract computing devices. Students will also learn about the limitations of computing machines as this course addresses the issue of which problems can be solved by computational means (decidability vs undecidability

Learning outcomes

On successful completion of the course, students will be able to:

- design a finite automaton, pushdown automaton or a Turing machine for a problem at hand.