SYMBIOSIS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Batch: 2014 - 2018 **Branch: Computer Science** **Semester: V**

Course Code - 07211507CS

Course Name - Design And Analysis of Algorithms Lab

Credit Points	Practical Hrs/Week	Total marks
01	2	25

Objective	•1 To study and perform analysis of algorithms. •2 To study techniques/strategies in design of algorithms
Prerequisites	Data Structures

Assignment list for practical:

Assignment 1:

1. Write a menu driven program to implement binary search (recursive & non-recursive), determine the time required to search the element. Repeat the experiment for different values of n, the number of elements in the list and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.
2. Sort a given set of elements using the Merge sort and Insertion sort determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator**.** Also display the partial pass-wise sorting done. The efficiency of Sort can be measured by determining the CPU runtime of an implementation of the algorithm to sort a number of elements versus the CPU runtime it takes an implementation of the Insertion Sort algorithm. Sort algorithm with the following array sizes: 5, 10, 15, 20, 30, 50, 100, 1000, 5000, 10000, 15000, and 20000. Each array size to be run ten times and the average of the CPU runtimes to be taken

Assignment 2:

1. Sort a given set of elements using the Quick sort method. (using randomized pivot & middle element pivot chosen as the pivot element)
2. Determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n. The elements can be generated using the random number generator. Also display the partial pass-wise sorting done. Sort algorithm with different array sizes.

Assignment 3:

Implement the “Heapsort” algorithm. The “heapify” process should be implemented as a separate method

Assignment 4:

Create implementations for the Kruskal’s & Prim’s Algorithm for finding the Minimum Cost Spanning Tree for a given graph. You may input the graph from file and use adjacency matrix.

Assignment 5:

Implement Dijkstra’s Algorithm to find a path of minimum total weight (cost) from a starting node to all the other nodes in the weighted graph.

Assignment 6:

WAP to perform Optimal Merge Pattern using heaps.

Assignment 7:

Implement Floyd’s Algorithm to find a shortest path between each pair of vertices in a weighted graph. Display the results as a distance matrix.

Assignment 8:

WAP to perform Knapsack problem.

Version 1: Given a set S of n items, such that each item i has a positive benefit bi and a positive weight wi , the goal is to find the maximum benefit subset that does not exceed a given weight W, **not** allowing for fractional items.

Version 2: find the maximum benefit subset that does not exceed a given weight W, allowing for fractional items.

Both the versions to print the list of Items that were included in the knapsack + total value in the Kanpsack.

Text Books	1. Bressard, "Fundamental of Algorithm.", PHI. 2. Horowitz and Sahani, "Fundamentals of computer Algorithms", Galgotia.