DEPARTMENT OF DATA SCIENCE AND COMPUTER APPLICATIONS

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

IVth SEMESTER B. TECH DATA SCIENCE AND ENGINEERING

LAB MANUAL

FOR THE COURSE

DSE 2264 DESIGN AND ANALYSIS OF ALGORITHMS

(March – June 2022)

Instructors:

Dr. Savitha G.
Assistant Professor,
Dept. of DSCA,
MIT, Manipal

Dr. Abhilash K. Pai
Assistant Professor,
Dept. of DSCA,
MIT, Manipal

I. COURSE OBJECTIVES

1. To write programs for solving various problems with different algorithm design techniques and data structures
2. To measure the complexities involved in an algorithm or the resulting program
3. To make a decision regarding the right algorithm or the resulting program for an environment involving the nature of input, configuration of computational device and/or any other conditions.

II. COURSE OUTCOMES

At the end of this course, students will have the:

1. Ability to design one or more algorithms for a problem using appropriate data structures
2. Convert the algorithm into a program which is efficient
3. Ability to determine the complexity of various algorithms or the resulting program

III. ASSESSMENT PLAN

1. Continuous Evaluation 60%

Total of 3 regular evaluations which will be carried out in alternate weeks. Each evaluation is for 20 marks of which will have the following split up: Record: 6 Marks; Viva: 10 Marks; Execution: 4 Marks; Total = 20 Marks

Total Internal Marks: 3 * 20 =60 Marks

2. Lab Examination 40%

- Examination of 2 hours duration (Max. Marks: 40). Program Write up: 15 Marks; Program Execution: 25; Marks Total: 15+25 =40 Marks

IV. INSTRUCTIONS TO THE STUDENTS

Pre- Lab Session Instructions

1. Students should carry the Observation Record to every lab session.
2. Be in time and follow the institution dress code.
3. Must Sign in the log register provided.
4. Students should attend the lab as per the batch-wise allotment and answer the attendance for the corresponding lab.
5. Adhere to the rules and maintain the decorum.

In- Lab Session Instructions

1. Follow the instructions on the allotted exercises.
2. All programs must be written in C++ programing language by using the "vi" editor.
3. Show the program and results to the instructors on completion of experiments.
4. On receiving approval from the instructor, copy the program and results in the Observation Record.
5. Prescribed textbooks and class notes can be kept ready for reference if required

General Instructions for the exercises in Lab

1. Implement the given exercise individually and not in a group.
2. The programs should meet the following criteria:
    - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
    - Programs should perform input validation (Data type, range error, etc.) and give appropriate error messages and suggest corrective actions.
    - Comments should be used to give the statement of the problem and every function should indicate the purpose of the function, inputs and outputs.
    - Statements within the program should be properly indented.
    - Use meaningful names for variables and functions.
    - Make use of constants and type definitions wherever needed.

- Programs should include necessary time analysis part (Operation count /Step count method)

3. Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
4. The exercises for each week are divided under two sets:
    - Lab exercises - to be completed during lab hours
    - Additional Exercises - to be completed outside the lab or in the lab to enhance the skill
5. In case a student misses a lab class, he/ she must ensure that the experiment is completed during the repetition lab with the permission of the faculty concerned.
6. Questions for lab tests and examination are not necessarily limited to the questions in the manual but may involve some variations and / or combinations of the questions.
7. You may write scripts/programs to automate the experimental analysis of algorithms and compare with the theoretical result.
8. For each question in the lab exercise, the observation record must include the following:
    a) The lab exercise question number, its description, and the date (right side).
    b) The appropriate algorithm for the corresponding exercise (left side).
    c) The analysis part, which must contain the theoretical analysis of the algorithm (left side).
    d) The program part, which includes the line-by-line C++ code.
    e) The output part, where the output for more than one forms of input must be included wherever possible (left side).
    f) The order of growth profile for the basic operation of the algorithm (left side).
9. You may use spreadsheets to plot the graph and then copy the profile on to the observation record.

<u>The students should NOT</u>

- Bring mobile phones or any other electronic gadgets to the lab.
- Go out of the lab without permission.

Lab No-1:

Fundamentals of Algorithmic Problem Solving + Brute Force-I

L1. 1) Write an algorithm for finding the Greatest Common Divisor (GCD) of two numbers using Euclid's algorithm. Determine the time and space complexity of the algorithm. Write a program to implement the Euclid's algorithm. Plot the order of growth graph for the number of times the basic operation of this algorithm will be executed for different input sizes.

L1. 2) Write a program to find GCD using consecutive integer checking method and analyze its time efficiency.

L1. 3) Write a program to sort a set of integers using selection sort and bubble sort algorithms and analyze their time efficiencies. Plot the order of growth graph for the number of times the basic operation of this algorithm will be executed for different input sizes.

L1. 4) Write a program to implement brute-force string matching. Analyze its time efficiency.

Additional Questions:

L1. a) Write a program for computing $\lfloor \sqrt{n} \rfloor$ for any positive integer and analyze its time efficiency. Besides assignment and comparison, your program may only use the four basic arithmetic operations.

L1. b) Write a program to implement recursive solution to the Tower of Hanoi puzzle and analyze its time efficiency.

L1. c) Write a program to compute the n^{th} Fibonacci number recursively and analyze its time efficiency.

Lab No-2:

Brute Force-II

L2. 1) Write a program to implement Knapsack problem using brute-force design technique and analyze its time efficiency.
Knapsack Problem: Given *n* items of known weights $w_1, w_2 \ldots w_n$ values $v_1, v_2, \ldots v_n$ and a knapsack of capacity *B*, find the most valuable subset of items that fit into the knapsack.

L2. 2) Write a program for assignment problem by brute-force technique. Analyze its time efficiency.

L2. 3) Write a program for depth-first search of a graph. Identify the push and pop order of vertices.

L2. 4) Write a program for breadth-first search of a graph.

Additional Questions:

L2. a) Write a program to check whether a graph is bipartite or not using:

i) DFS to check for bipartite

ii) BFS to check for bipartite

Note: A graph is said to be bipartite if all its vertices can be partitioned into two disjoint subsets X and Y so that every edge connects a vertex in X with a vertex in Y.

L2. b) Write a program to construct a graph for the following maze. One can model a maze by having a vertex for a starting point, a finishing point, dead ends, and all the points in the maze where more than one path can be taken, and then connecting the vertices according to the paths in the maze. Also find the solution to the maze using Depth-First-Search.

Lab No-3:

Decrease-and-Conquer + Divide-and-Conquer-I

L3. 1) Write a program to sort a set of numbers using insertion sort and analyze its time efficiency.

L3. 2) Write a program to determine the Topological sort of a given graph using:

i) Depth-First technique

ii) Source removal technique

L3. 3) Write a program to sort given set of integers using Quick sort and analyze its efficiency.

L3. 4) Write a program to sort given set of integers using Merge sort and analyze its efficiency.

Additional Questions:

L3. a) A student management system should be developed by a software company. There are certain dependent and independent modules that must be developed by K teams. The unit testing and integration testing is done before the deployment of the product. Design an algorithm using decrease and conquer technique such that it gives a schedule of tasks in the order in which it must be executed.

L3. b) Write a program to find a^n where $n > 0$ using divide and conquer strategy.

L3. c) Modify the algorithm such that it checks whether there is any task dependency between the teams.

L3. d) Write a program to implement binary-search using divide and conquer strategy.

Lab No-4:

Divide-and-Conquer-II + Transform-and-Conquer-I

L4. 1) Write a program to determine the height of a binary search tree and analyze its time efficiency.

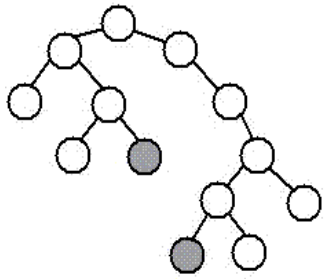
L4. 2) Write a program to create a binary search tree and display its elements using all the traversal methods and analyse its time efficiency.

L4. 3) Write a program to create a AVL tree for a set of integers and analyze its time efficiency. For the AVL tree created, insert an element 6.

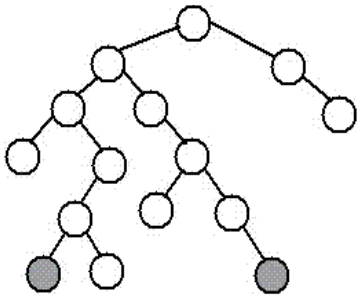
L4. 4) Write a program to create a 2-3 tree for a set of integers and analyze its time efficiency. For the 2-3 tree created, insert an element 6.

Additional Questions:

L4. a) Write a program to find diameter of a binary tree. Diameter of a binary tree is the longest path between any two nodes.
For e.g. consider the following two binary trees. The diameter in each tree is 9. In the first one, longest path passes through root whereas in the second it does not.



diameter, 9 nodes, through root



diameter, 9 nodes, NOT through root

L4. b) Write a program to find the total number of nodes in a binary tree and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

L4. c) Write a program for finding and deleting an element of a given value in a Heap.

Lab No-5:

Transform-and-Conquer-II + Space and Time Tradeoffs

L5. 1) Write a program to sort the list of integers using heap sort with bottom-up max heap construction and analyze its time efficiency. Prove experimentally that the worst-case time complexity is O (n log n).

L5. 2) Write a program to sort set of integers using comparison counting algorithm.

L5.3) Write a program to implement Horspool's algorithm for String Matching and find the number of key comparisons in successful search and unsuccessful search.

L5. 4) Write a program to construct the Open hash table. Find the number of key comparisons in successful search and unsuccessful search.

Additional Questions:

L5. a) Write a program to create a heap for the list of integers using top-down heap construction algorithm and analyze its time efficiency. Obtain the experimental results for order of growth and plot the result.

L5. b) Write a program to construct the closed hash table. Find the number of key comparisons in successful search and unsuccessful search.

L5. c) Write a program to implement Boyer-Moore algorithm for String Matching and find the number of key comparisons in successful search and unsuccessful search.

L5. d) Write a program to sort the elements using distribution counting method.

Lab No-6:

Dynamic Programming

L6. 1) Write a program to find the Binomial Co-efficient using Dynamic Programming.

L6. 2) Write a program to compute the transitive closure of a given directed graph using Warshall's algorithm and analyse its time efficiency.

L6. 3) Write a program to implement Floyd's algorithm for the All-Pairs-Shortest-Paths problem for any given graph and analyse its time efficiency.

L6. 4) Write a program to implement 0/1Knapsack problem using bottom-up dynamic programming.

Additional Questions:

L6. a) Write a program to implement **0/1** knapsack problem using memory functions.

L6. b) Write a program that finds the composition of an optimal subset from the table generated by the bottom-up dynamic programming algorithm for the knapsack problem.

Lab No-7:

Greedy Strategy

L7. 1) Write a program to find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.

L7. 2) Write a program to find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm and analyse its time efficiency.

L7. 3) Write a program to find shortest path from a given vertex to other vertices in a given weighted connected graph, Using Dijkstra's algorithm and analyse its time efficiency.

Additional Questions:

L7. a) Write a program to implement Huffman tree construction algorithm.

L7. b) Write a program to find a maximum spanning tree – a spanning tree with the largest possible edge weight of a weighted connected graph.

L7. c) Write a program to implement the greedy algorithm for the change-making problem, with an amount n and coin denominations $\mathbf{d_1 > d_2 > ... > d_m}$ as its input.

Lab No-8:

Backtracking & Branch-and-Bound

L8. 1) Write a program for N-queens problem using backtracking technique.

L8. 2) Write a program to find the solution to the subset-sum problem for
S= {1, 2, 5, 6, 8} and d=9, using backtracking.

L8. 3) Write a program to implement Knapsack problem using branch and bound technique.

Additional Questions:

L8. a) Write a program for finding Hamiltonian circuit for the graph, using backtracking.

L8. b) Write a program to implement assignment problem using Branch and Bound.

REFERENCES:

1. Anany Levitin, Introduction to The Design and Analysis of Algorithms, 3rd Edition, Pearson Education, India, 2012.

2. Ellis Horowitz, SartajSahni, Dinesh Mehta, Fundamentals of Data Structures in C++, 2nd Edition, GalgotiaPublications, Reprint 2013

3. Thomas H. Cormen, Charles E. Leiserson, Ronal L, Rivest, Clifford Stein, Introduction to Algorithms, Phi, 2nd Edition, 2006.