Pandit Deendayal Energy University

School of Technology

Department of Computer Science & Engineering

Even Semester 2024-2025

LIST OF PRACTICAL FOR DAA

BY

DR. ADITYA SHASTRI

Practical No. 1 and 2

Implement the following sorting in any programming language.

a) Insertion sort
b) Selection sort
c) Merge sort
d) Quick sort

Now, measure the execution time and the number of steps required to execute each algorithm in best case, worst case, and average case.

Practical No. 3

Use singly linked lists to implement integers of unlimited size. Each node of the list should store one digit of the integer. You should implement addition, subtraction, multiplication, and exponentiation operations. Limit exponents to be positive integers.

What is the asymptotic running time for each of your operations, expressed in terms of the number of digits for the two operands of each function?

Practical No. 4

Implement a city database using unordered lists. Each database record contains the name of the city (a string of arbitrary length) and the coordinates of the city expressed as integer x and y coordinates. Your program should allow following functionalities:

a) Insert a record,
b) Delete a record by name or coordinate,
c) Search a record by name or coordinate.
d) Pint all records within a given distance of a specified point.

Implement the database using an array-based list implementation, and then a linked list implementation. Perform following analysis:

a) Collect running time statistics for each operation in both implementations.
b) What are your conclusions about the relative advantages and disadvantages of the two implementations?
c) Would storing records on the list in alphabetical order by city name speed any of the operations?
d) Would keeping the list in alphabetical order slow any of the operations?

Experiment No. 5 [Greedy Approach]

1. Implement Kruskal's algorithm for MST by Union Find Data Structure. Take a graph of 'n' nodes and 'm' edges and print the edges that will be present in MST and the cost of the MST. The following graph can be taken as an example to test your algorithm.

   $G = (V, E)$ where $|V| = 4$ and $|E| = 5$

   $E = \{0-1, 0-2, 0-3, 1-3, 2-3\}$

   $Wt = [10, 6, 5, 15, 4]$

2. Implement interval scheduling algorithm. Given $\boldsymbol{n}$ events with their starting and ending times, find a schedule that includes as many events as possible. It is not possible to select an event partially. For example, consider the following example:

Event	Starting time	Ending time
A	1	3
B	2	5
C	3	9
D	6	8

   Here, maximum number of events that can be scheduled is 2. We can schedule B and D together.

Practical No. 6 [Divide and Conquer]

Implement both a standard $O(n^3)$ matrix multiplication algorithm and Strassen's matrix multiplication algorithm. Using empirical testing, try and estimate the constant factors for the runtime equations of the two algorithms. How big must $\boldsymbol{n}$ be before Strassen's algorithm becomes more efficient than the standard algorithm?

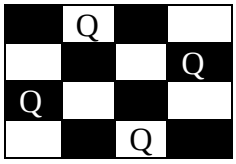
Experiment No. 7 [Dynamic Programming]

Implement the Floyd Warshall Algorithm for All Pair Shortest Path Problem. You are given a weighted diagraph $G = (V, E)$, with arbitrary edge weights or costs $\boldsymbol{c_{vw}}$ between any node $\boldsymbol{v}$ and node $\boldsymbol{w}$. Find the cheapest path from every node to every other node. Edges may have negative weights. Consider the following test case to check your algorithm:

$\boldsymbol{v}$	$\boldsymbol{w}$	$\boldsymbol{c_{vw}}$
0	1	-1
0	2	4
1	2	3
1	3	2
1	4	2
3	2	5
3	1	1
4	3	-3

Practical No. 8 [Backtracking]

Solve the $\boldsymbol{n}$ queens' problem using backtracking. Here, the task is to place $\boldsymbol{n}$ chess queens on an n x n board so that no two queens attack each other. For example, following is a solution for the 4 Queen' problem.

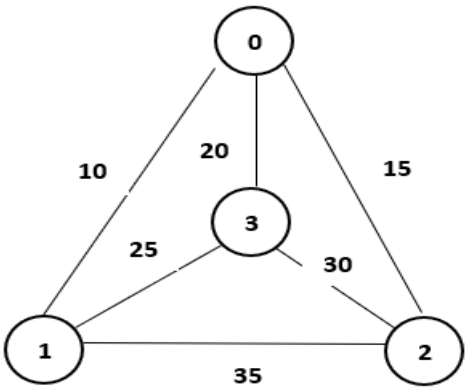


Practical No. 9 [Branch and Bound]

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible tour that visits every city exactly once and returns to the starting point.

Solve this problem using branch and bound technique.

For example, consider the following graph:



A Travelling Salesman Problem (TSP) tour in the graph is $0 - 1 - 3 - 2 - 0$. The cost of the tour is $10 + 25 + 30 + 15 = 80$.

Practical No. 10

To design and solve given problems using different algorithmic approaches and analyze their complexity.

1. Your friends are starting a security company that needs to obtain licenses for n different pieces of cryptographic software. Due to regulations, they can only obtain these licenses at the rate of at most one per month. Each license is currently selling for a price of \$100. However, they are all becoming more expensive according to exponential growth curves: in particular, the cost of license j increases by a factor of $r_j > 1$ each month, where r_j is a given parameter. This means that if license j is purchased t months from now, it will cost $100r_j t$. We will assume that all the price growth rates are distinct; that is, $r_i \neq r_j$ for licenses $i \neq j$ (even though they start at the sameprice of \$100).

The question is: Given that the company can only buy at most one license a month, in which order should it buy the licenses so that the total amount of money it spends is as small as possible?
Give an algorithm that takes the n rates of price growth $r_1, r_2, \ldots, r_n$, and computes an order in which to buy the licenses so that the total amount ofmoney spent is minimized. The running time of your algorithm should be polynomial in n.

2. Suppose you are given an array A with n entries, with each entry holding a distinct number. You are told that the sequence of values $A[1], A[2], \ldots, A[n]$ is unimodal. That is, for some index p between 1 and n, the values in the array entries increase up to position p in A and then decrease the remainder of the way until position n. (So if you were to draw a plot with the array position j on the x-axis and the value of the entry $A[j]$ on the y-axis, the plotted points would rise until x-value p, where they'd achieve their maximum value, and then fall from there on). You'd like to find the "peak entry" p without having to read the entire array - in fact, by reading as few entries of A as possible. Show how to find the entry p by reading at most $O(log\ n)$ entries of A.