Q.1. Coin Tossing

Through the simulation, show that probability of getting HEAD by tossing a fair coin is about 0.5. Write your observation from the simulation run.

Q.2. Performance analysis of Bubble Sort

Write the program to implement two different versions of bubble sort(BUBBLE SORT that terminates if the array is sorted before n-1th Pass. Vs. BUBBLE SORT that always completes the n-1th Pass) for randomized data sequence.

Q.3. Average case analysis for Sorting Algorithms

For each of the data formats: random, reverse ordered, and nearly sorted, run your program say **SORTTEST** for all combinations of sorting algorithms and data sizes and complete each of the following tables. When you have completed the tables, analyze your data and determine the asymptotic behavior of each of the sorting algorithms for each of the data types *(i)* ***Random data, (ii) Reverse Ordered Data, (iii) Almost Sorted Data*** **and (iv)** ***Highly Repetitive Data*** . select the suitable no of elements for the analysis that supports your program.

Q.4. Variants of QUICK SORT

Compare the performance of **variants of quick sort** algorithm for instance characteristic n=10, ..., 1000. Use the finding from Q3, [cross-over point where insertion sort shows the better performance over quick sort] Modify your sorting algorithm in the previous problem to stop partitioning the list in QUICKSORT when the size of the (sub)list is less than or equal to 12 and sort the remaining sublist using INSERTIONSORT. Your counter will now have to count compares in both the partition function and every iteration of INSERTIONSORT. Again, run the experiment for 50 iterations and record the same set of statistics. Compare your results for the two different sorting techniques and comment upon your results.

Q.5. STRASSENS's Matrix multiplication

Practical implementation of Strassens's matrix multiplication algorithm is usually switched to the brute force method after matrix sizes become smaller than some crossover point. Run an experiment to determine such crossover point on your computer system.

Q.6. QUICK SELECT

Use the **QUICK SELECT** algorithm to find **3rd largest element** in an array of n integers. Analyze the performance of **QUICK SELECT** algorithm for the different instance of size 50 to 500 element. Record your observation with the *number of comparison made* vs. *instance*.

Q.7. Iterative Binary Search

Write programs to implement recursive and iterative versions of binary search and compare the performance. For a appropriate size of n=20(say), have each algorithm find every element in the set. Then try all n+1 possible unsuccessful search. The performance, of these versions of binary search are to be reported graphically with your observations.

Q.8. Matrix Chain Multiplication Comparision

Given a matrix chain $A_1 \ldots A_n$ with the dimension of each of the matrices given by the vector **p** = <12,21,65,18,24,93,121,16,41,31,47,5,47,29,76,18,72,15>. (n=17) Write and run both the dynamic programming and memorized versions of this algorithm to determine the minimum number of multiplications that are needed (use type ***longint***) and the factorization that produces this best case number of multiplications. Run each of the two programs over an appropriately large number of times (put each in a loop to run repeatedly x times) and obtain the

times at the beginning and end of the run. Use these times to determine the comparative runtimes of the two algorithms.

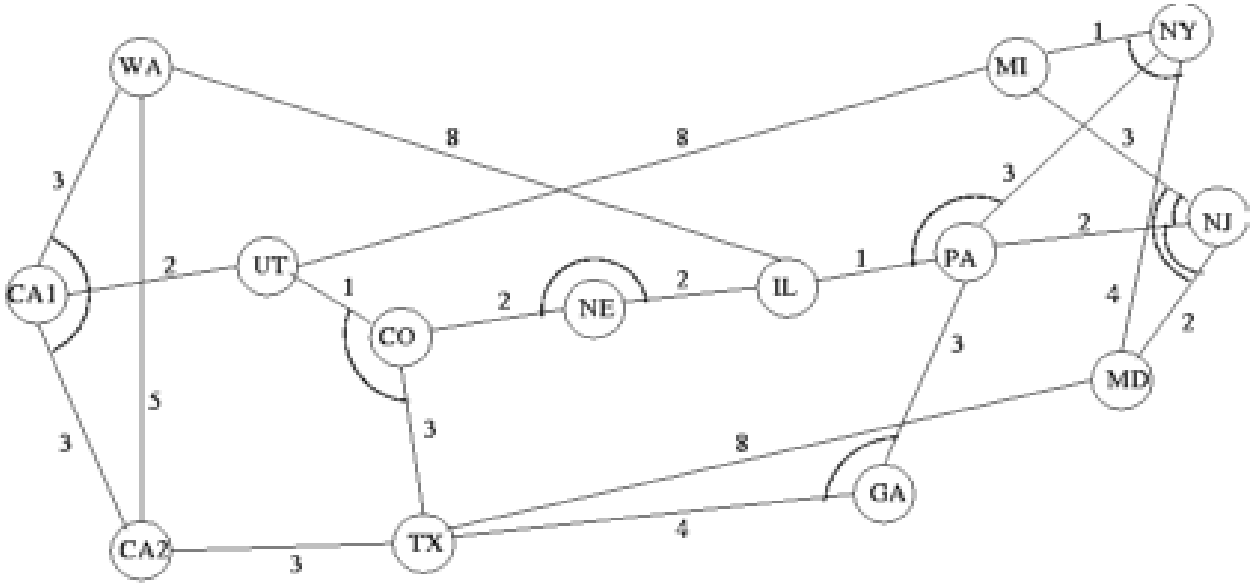
Q.9. BINOMIAL COEFFICIENTS

Computing the **binomial coefficients** *C*(*n*, *k*) defined by the following recursive formula:

$$C(n,k) = \begin{cases} 1, & \text{if } k = 0 \text{ or } k = n; \\ C(n-1,k) + C(n-1,k-1), & \text{if } 0 < k < n; \\ 0, & \text{otherwise.} \end{cases}$$

Write the program with three different algorithm to compute **binomial coefficients** *C*(*n*, *k*) and compare them?

Q.10. SPANNING TREE



Write a program obtain minimum cost spanning tree the above NSF network using Prim's algorithm, Kruskal's algorithm and Boruvka's algorithm. Use the appropriate data structure to store the computed spanning tree for another application (broadcasting a message to all nodes from any source node).

Q.11. 0-1 KNAPSACK PROBLEM

Write a program that computes **optimal solution** to the 0–1 Knapsack Problem using dynamic programming? You may test your program with the following example:

There are n = 5 objects with integer weights w[1..5] = {1,2,5,6,7}, and values v[1..5] = {1,6,18,22,28}. Assuming a knapsack capacity of 11).

Compare your solution to a greedy algorithm that computes sub-**optimal solution** the 0–1 Knapsack Problem.

Q.12. STRING MATCHING ALGORITHM

Given two strings P and T over the same alphabet set Σ, determine whether P occurs as a substring in T (or find in which position(s) P occurs as a substring in T). The strings P and T are called pattern and text respectively. Compare the efficiency of three string matching algorithms (Brute-Force Algorithm, Knuth-Morris-Pratt and Boyer-Moore Algorithm) by varying pattern length [1-15] for n=5000.

Q.13. MATRIX CHAIN MULTIPLICATION

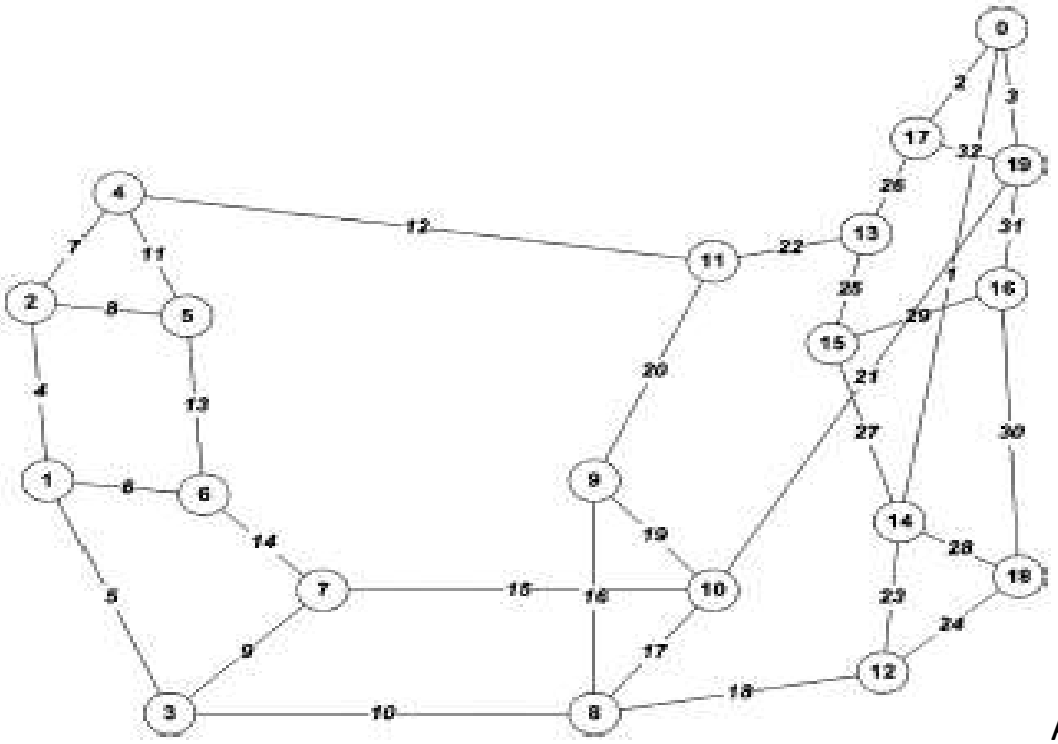
Write a program to compute the best ordering of matrix multiplication. Include the routine to print the actual ordering.

Q.14. TSP TOUR

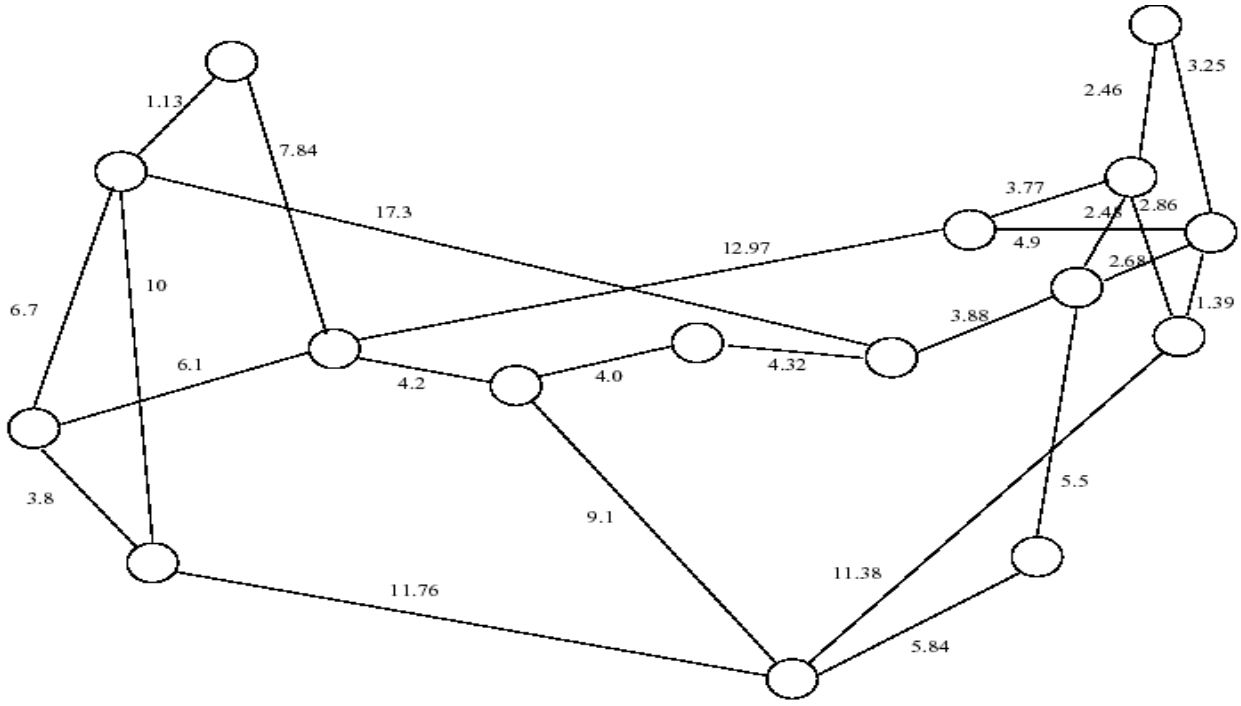
Write a program to verify that a candidate solution for a TSP tour is correct. The input for this program will consist of a graph, with the cities as the vertices and the connecting roads as edges, and the certificate. The graph may be expressed as an N $\times$ N matrix with the (i, j)th entry indicating the distance between the cities numbered *i* and *j*. If no road directly connects this pair of cities, the matrix entry will be 0. The certificate consists of a sequential list of the cities visited during the tour and the target value that the tour is not to exceed. Demonstrate that your program decides this question in O(P(N)) time.

Q.15. PATHING ALGORITHM

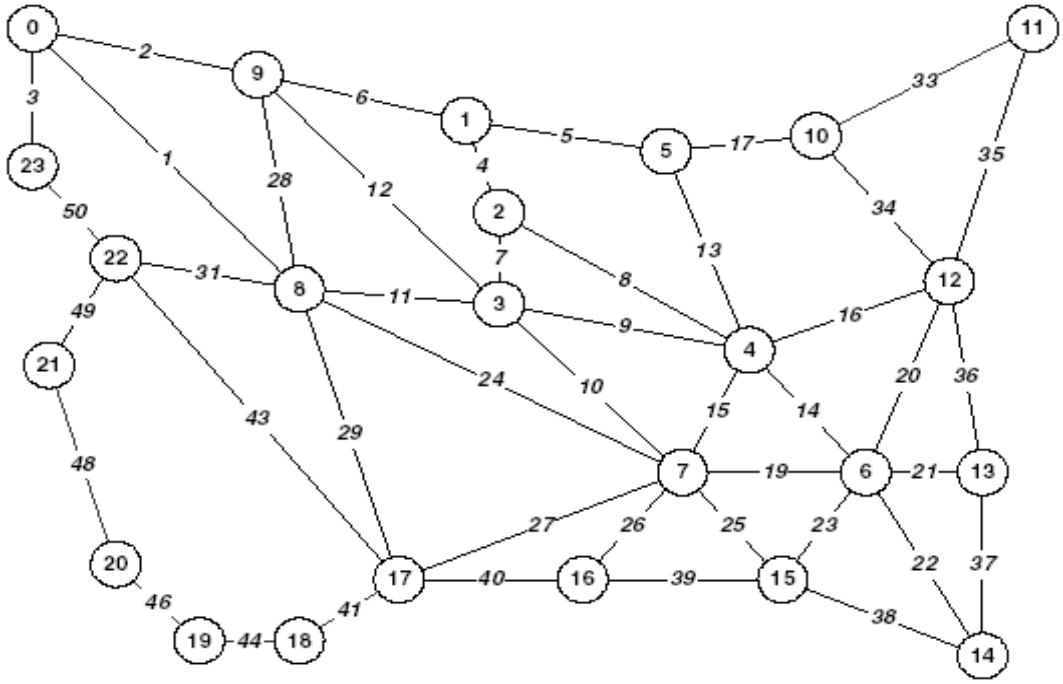
Use Floyd–Warshall algorithm (also known as Floyd's algorithm) to compute all pair **shortest path** for all these following standard network



ARPA Network



NSF Network



NATIONAL NETWORK

Mail your report [Source Codes + Simulation results +Observations] as a single file with name as Roll No DSAD # *to* compscnit@gmail.com..

 # is the part of the assignment you are submitting.

Suppose the following student submit the Laboratory assignment in 4 parts (once in a week).

220CS1032	SHEETAL AGARWA

The files names are:

220CS1031 DSAD I, 220CS1031 DSAD II, 220CS1031 DSAD III, 220CS1031 DSAD IV