Course Code	**:**	**BCS-042**
Course Title	**:**	**Introduction to Algorithms Design**
Assignment Number	**:**	**BCA_NEW/BCAOL(V)042/Assign/2026-27**
Maximum Marks	**:**	**100**
Last Dates for Submission	**:**	**31st October, 2026 (For July 2026 Session)**
	:	**30th April, 2027 (For January 2027 Session)**

This assignment has 8 questions of 10 Marks each; answer all questions. The remaining 20 marks are for viva voce. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.

Q1: Consider the following sorted array DATA with 13 elements. Illustrate the working of the binary search technique, while searching for elements: (i) 42 (ii) 82

11	22	27	35	42	45	55	60	65	70	75	82	85

Q2: Use Strassen's matrix multiplication algorithm to multiply the following two matrices:

$$A=\begin{bmatrix}5 & 9\\ 8 & 7\end{bmatrix} \qquad B=\begin{bmatrix}3 & 6\\ 4 & 2\end{bmatrix}$$

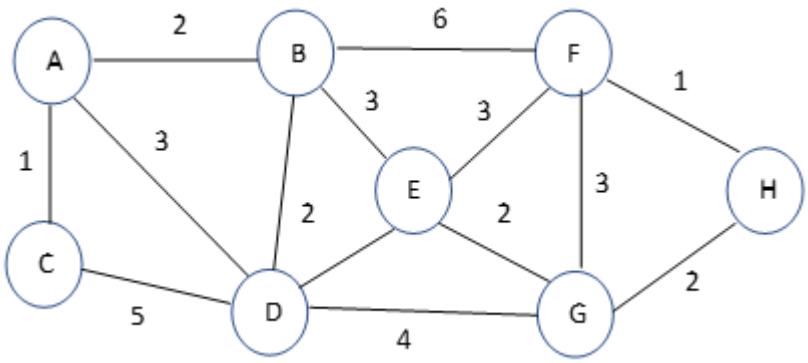
Q3: Find GCD(595,252) by Euclid"s algorithm. Also, do the best-case, worst-case, and average-case time complexity analysis of it.

Q4: Compute a^{283} using left-to-right and right-to-left binary exponentiation.

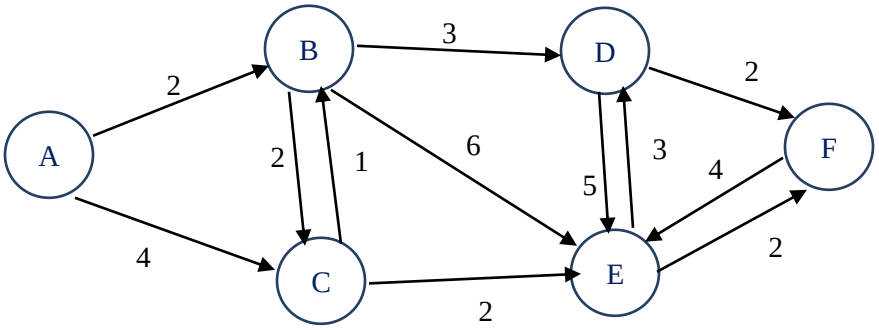
Q5: Analyze the best-case, average-case, and worst-case time complexities of the following algorithms with the help of suitable examples.

(i) Insertion sort
(ii) Selection sort
(iii) Quick sort
(iv) Merge sort

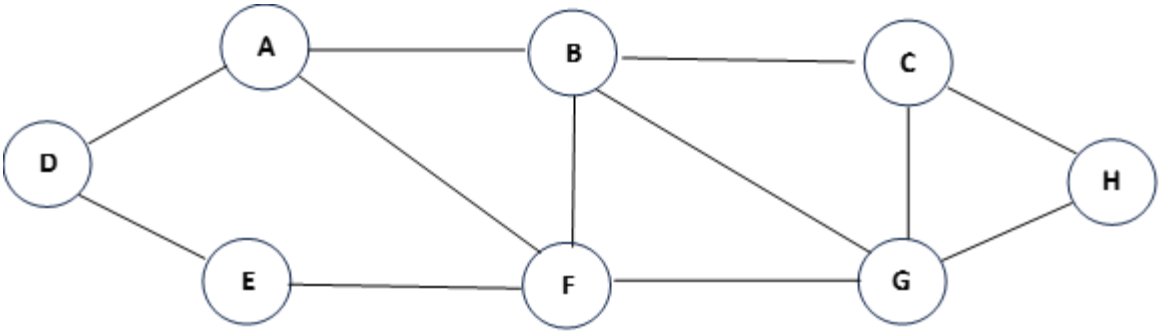
Q6: Apply Prim's Algorithm on the following graph to find the minimum cost spanning tree



Q7: Apply the Bellman-Ford algorithm to find the shortest path from source vertex 'A' to all other vertices for the following graph.



Q8: Apply DFS and BDS Graph traversal algorithms on the following graph and find the traversal order of nodes. Select vertex A as the starting point.



Q1: Consider the following sorted array DATA with 13 elements. Illustrate the working of the binary search technique, while searching for elements: (i) 42 (ii) 82

11	22	27	35	42	45	55	60	65	70	75	82	85

Answer

Binary search is applicable because DATA is already sorted. I use one-based indexing for all thirteen positions. At each step, the middle element of the current search interval is compared with the required value. If the middle value is larger, the right half is rejected; if it is smaller, the left half is rejected.

(i) Search for 42

Step 1. low = 1, high = 13, so mid = floor((1 + 13)/2) = 7. DATA[7] = 55. Since 42 < 55, continue in positions 1 to 6.

Step 2. low = 1, high = 6, so mid = floor((1 + 6)/2) = 3. DATA[3] = 27. Since 42 > 27, continue in positions 4 to 6.

Step 3. low = 4, high = 6, so mid = 5. DATA[5] = 42. The element is found at position 5 after three comparisons.

(ii) Search for 82

Step 1. low = 1, high = 13 gives mid = 7 and DATA[7] = 55. Since 82 > 55, continue in positions 8 to 13.

Step 2. low = 8, high = 13 gives mid = floor((8 + 13)/2) = 10 and DATA[10] = 70. Since 82 > 70, continue in positions 11 to 13.

Step 3. low = 11, high = 13 gives mid = 12 and DATA[12] = 82. The element is found at position 12 after three comparisons.