| (Affiliated to University of Mumbai) | ***End Semester Examination (R-24) FH 2026*** |
|---|---|
| Branch: CE-AIDS-AIML | Course: Analysis of Algorithm |
| Year/ Semester: SE/IV | Course code: CEC403 |
| Time: 03 hours | Marks: 80 |

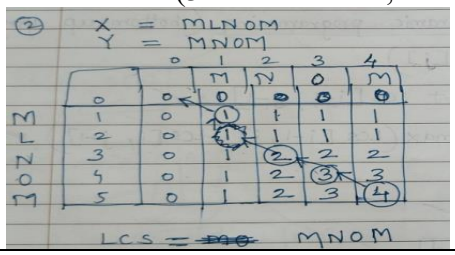
**Note: 1. All questions are compulsory.**
**2. Figures to right indicate full marks.**
**3. Assume suitable data wherever necessary.**

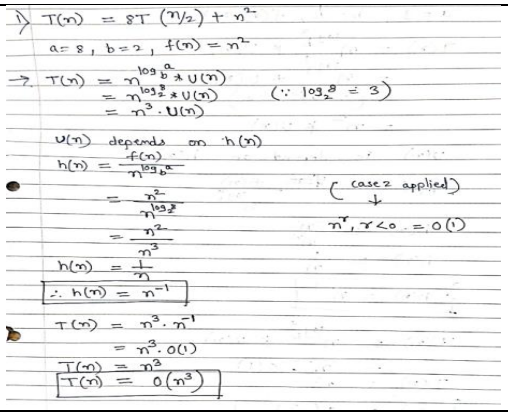
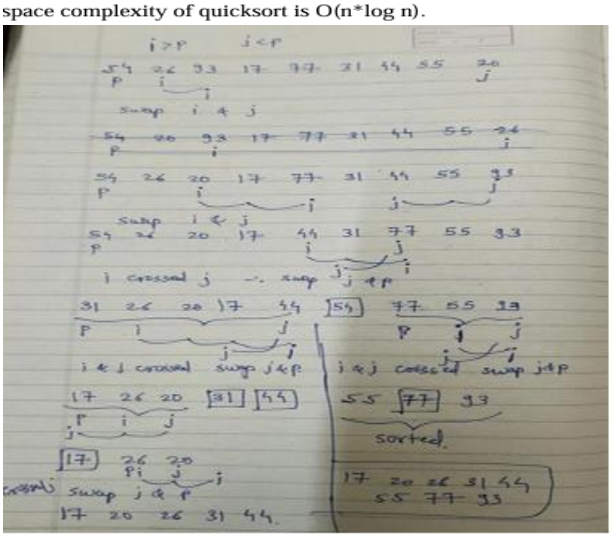
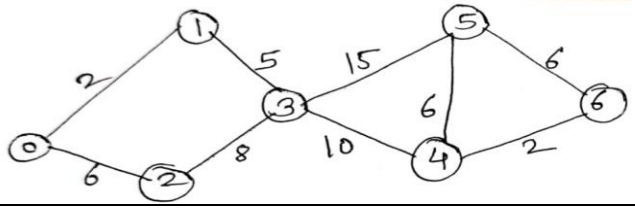
| | | Marks | CO | BL |
|---|---|---|---|---|
| Q. 1 | Attempt any FOUR. (All questions carry equal marks) | **20** | | |
| A. | Differentiate between selection sort and insertion sort. (1M each)) | 05 | CO1 | L2 |
| B. | Describe binary search and write algorithm for binary search. (2m description,3m algorithm.) | 05 | CO2 | L2 |
| C. | Solve using fractional knapsack problem. (N=5, W=10(knapsack weight)) (1 m each step,1m profit value) | 05 | CO3 | L3 |

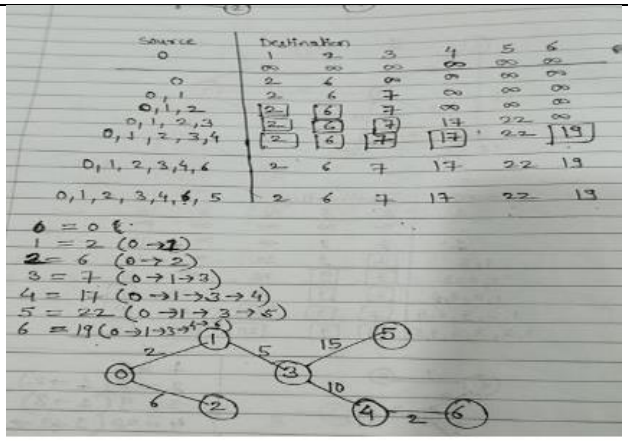
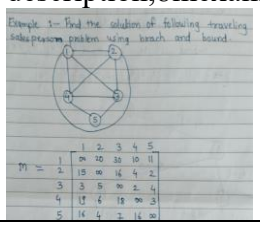
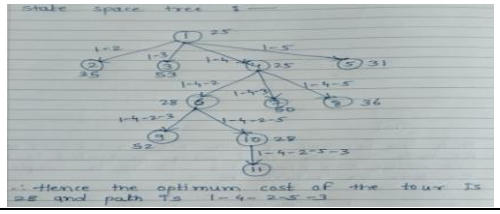
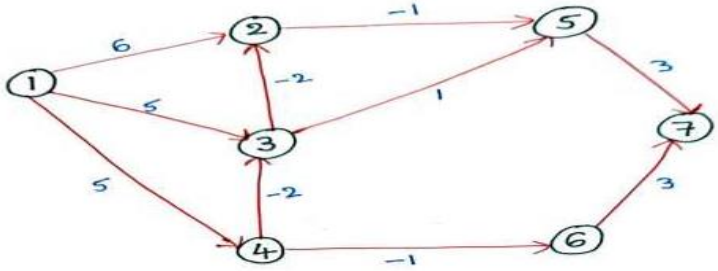
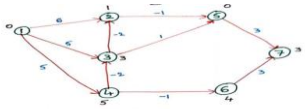
| | Insertion Sort | Selection Sort |
|---|---|---|
| 1. | Inserts the value in the presorted array to sort the set of values in the array. | Finds the minimum / maximum number from the list and sort it in ascending / descending order. |
| 2. | It is a stable sorting algorithm. | It is an unstable sorting algorithm. |
| 3. | The best-case time complexity is Ω(N) when the array is already in ascending order. It have $\Theta(N^2)$ in worst case and average case. | For best case, worst case and average selection sort have complexity $\Theta(N^2)$. |
| 4. | The number of comparison operations performed in this sorting algorithm is less than the swapping performed. | The number of comparison operations performed in this sorting algorithm is more than the swapping performed. |
| 5. | It is more efficient than the Selection sort. | It is less efficient than the Insertion sort. |
| 6. | Here the element is known beforehand, and we search for the correct position to place them. | The location where to put the element is previously known we search for the element to insert at that position. |
| 7. | The insertion sort is used when:<br>• The array is has a small number of elements<br>• There are only a few elements left to be sorted | The selection sort is used when<br>• A small list is to be sorted<br>• The cost of swapping does not matter<br>• Checking of all the elements is compulsory<br>• Cost of writing to memory matters like in flash memory (number of Swaps is O(n) as compared to O(n2) of bubble sort) |

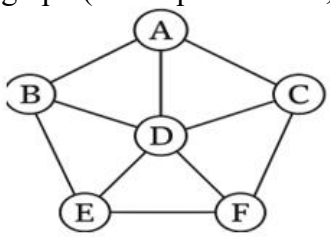
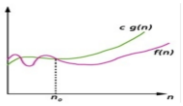
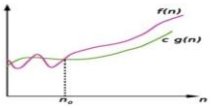
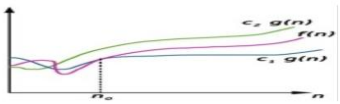
```
Algorithm BINARY_SEARCH(A, Key)
// Description : Perform a binary search on
array A
// Input : Sorted array A of size n and Key
to be searched
// Output : Success / Failure

low ← 1
high ← n
while low < high do
    mid ← (low + high) / 2
    if A[mid] == key then
        return mid
    else if A[mid] < key then
        low ← mid + 1
    else
    high ← mid - 1
    end
end
return 0
```

| Object | Profit Value | Weight |
|---|---|---|
| 1 | 10 | 3 |
| 2 | 15 | 3 |
| 3 | 10 | 2 |
| 4 | 12 | 5 |
| 5 | 8 | 1 |

| | | | | |
|---|---|---|---|---|
| | <table><tr><th>Knapsack weight</th><th>Items in knapsack</th><th>profit</th></tr><tr><td>10</td><td>0</td><td>0</td></tr><tr><td>9</td><td>I5</td><td>8</td></tr><tr><td>6</td><td>I5,I2</td><td>8+15=23</td></tr><tr><td>4</td><td>I5,I2,I3</td><td>23+10=33</td></tr><tr><td>1</td><td>I5,I2,I3,I1</td><td>33+10=43</td></tr></table> Knapsack weight left to be filled =1, but item 4 has weight =5<br>In fractional knapsack problem, even the fraction of nay item can be taken.<br>Therefore knapsack will contain: [I5, I2, I3, (1/5)*I4]<br>Total profit=43+ (1/5)*12<br>=43+2.4<br>=45.4 | | | |
| D. | Apply LCS and find longest subsequence: X=MLNOM and Y=MNOM (3m table values,2m LCS)<br>  | 05 | CO4 | L3 |
| E. | Explain N queen problem and write 4*4 queen solution.(3m n queen description with rules,2m 4*4 solution)<br><table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th></tr><tr><td>0</td><td></td><td></td><td>Q1</td><td></td></tr><tr><td>1</td><td>Q2</td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td>Q3</td></tr><tr><td>3</td><td></td><td>Q4</td><td></td><td></td></tr></table><table><tr><td>x</td><td>Q1</td><td>x</td><td>x</td></tr><tr><td>x</td><td>x</td><td>x</td><td>Q2</td></tr><tr><td>Q3</td><td>x</td><td>x</td><td>x</td></tr><tr><td>x</td><td>x</td><td>Q4</td><td>x</td></tr></table> | 05 | CO5 | L2 |
| F. | Describe P, NP complexity class. (2.5 m each class description)<br>• **P Class**<br>• The P in the P class stands for **Polynomial Time.** It is the collection of decision problems(problems with a "yes" or "no" answer) that can be solved by a deterministic machine in polynomial time.<br>• **Features:**<br>– The solution to **P problem**s is easy to find.<br>– **P** is often a class of computational problems that are solvable and tractable. Tractable means that the problems can be solved in theory as well as in practice. But the problems that can be solved in theory but not in practice are known as intractable.<br>• This class contains many problems:<br>– Calculating the greatest common divisor.<br>– Finding a maximum matching.<br>– Prime number<br>– insertion, and Merge Sort, Binary search<br>• **NP Class**<br>• The NP in NP class stands for **Non-deterministic Polynomial Time(Exponential Time eg 2^k).**<br>• A problem that can not be solved in polynomial time but can be verifiable in polynomial time using non-deterministic algorithm is known NP Problem.<br>• It is the collection of decision problems that can be solved by a non-deterministic machine in polynomial time.<br>• **Features:**<br>– The solutions of the NP class are hard to find since they are being solved by a non-deterministic machine but the solutions are easy to verify.<br>– Problems of NP can be verified by a Turing machine in polynomial time.<br>• This class contains many problems that one would like to be able to solve effectively:<br>– Scheduling problems<br>– Sudoku Problem<br>– 0/1 Knapsack problem.<br>– Graph coloring.<br>– Traveling Salesman problem | 05 | CO6 | L2 |
| | | | | |
| Q.2 | Attempt any FOUR. (All questions carry equal marks) | **40** | | |
| A. | Explain recurrence and solve using master method. $T(n)=8T(n/2)+n^2$ (2m recurrence,2m master method formula,6m example.) | 10 | CO1 | L3 |

| | | | | |
|---|---|---|---|---|
| |  | | | |
| B. | Sort the given array elements using quick sort and write quick sort algorithm and mention time and space complexity. 54,26,93,17,77,31,44,55,20. (5m example,3m algorithm,2m time and space complexity)<br><br>**Case** / **Time Complexity**<br>**Best Case** — O(n*logn)<br>**Average Case** — O(n*logn)<br>**Worst Case** — $O(n^2)$<br><br>space complexity of quicksort is O(n*log n).<br>  | 10 | CO2 | L3 |
| C. | Describe greedy approach. Apply Dijkstra's algorithm and find the shortest path from source vertex 0. (2m greedy approach description, 6m table,2m path and graph)<br>  | 10 | CO3 | L3 |

| | | | | |
|---|---|---|---|---|
| |  | | | |
| D. | Explain Travelling Salesperson Problem with example using branch and bound. (2m Travelling Salesperson Problem description,8mexample)<br>   | 10 | CO5 | L3 |
| E. | Apply Bellman ford Algorithm and find the shortest path from source vertex 1 for the given graph.(2m each iteration)<br>    | 10 | CO4 | L3 |
| F. | Explain Knuth-Morris-Pratt algorithm with example .(4m Knuth-Morris-Pratt algorithm,6m example)<br>• **Algorithm Steps**<br>**Step 1: Build LPS Array**<br>• Initialize LPS[0] = 0<br>• Compare characters and update prefix-suffix matches<br>**Step 2: Pattern Matching**<br>• Compare text and pattern<br>• If match → move both pointers<br>• If mismatch:<br>  • Use LPS to shift pattern (no rechecking) | 10 | CO6 | L2 |
| | | | | |
| Q.3 | Attempt any FOUR | **20** | | |
| A. | Apply Merge sort algorithm to sort an array and mention time and space complexity. Elements: 78,21,35,44,12,55,66,88,90 (4m | 05 | CO2 | L3 |

| | | | | |
|---|---|---|---|---|
| | algorithm,2m time and space complexity,4m example)<br>**Final Merge:**<br>`[21,35,44,78] + [12,55,66,88,90]`<br>`→ [12,21,35,44,55,66,78,88,90]`<br>**Final Sorted Array**<br>`[12, 21, 35, 44, 55, 66, 78, 88, 90]`<br>**Time and Space Complexity**<br>Case / Time Complexity / Space Complexity<br>Best / O(n log n) / O(n)<br>Average / O(n log n) / O(n)<br>Worst / O(n log n) / O(n) | | | |
| B. | Apply graph coloring technique and find chromatic number of a graph.(3m stepwise color,2m chromatic number)<br>  <br>**A valid 4-coloring**<br>One valid assignment with 4 colors is:<br>• Color 1: D<br>• Color 2: A, E<br>• Color 3: B, C<br>• Color 4: F | 05 | CO5 | L3 |
| C. | Explain multistage graph with example. (2m multistage graph,3m example) | 05 | CO4 | L2 |
| D. | Explain Big-Oh, Omega and Theta notation in detail. (1.7m each notation)<br>**1. Big-oh(O) notation:** Big-oh is the formal method of expressing the upper bound of an algorithm's running time. It is the measure of the longest amount of time.<br>Big-Oh (O) notation gives an upper bound for a function f(n) to within a constant factor.<br>  <br>**2. Omega (Ω) Notation:**<br>• Big-Omega (Ω) notation gives a lower bound for a function f(n) to within a constant factor.<br>  <br>**3.Big Theta Notation**<br>• Big-Theta(Θ) notation gives bound for a function f(n) to within a constant factor.<br>  | 05 | CO1 | L2 |
| E. | Explain 0/1 knapsack problem with example. (2m 0/1 knapsack problem,3m example) | 05 | CO4 | L3 |
| F. | Explain Job sequencing with deadlines algorithm with example. (2m Job sequencing with deadlines algorithm,3m example)<br>**Step-01:**<br>• Sort all the given jobs in decreasing order of their profit.<br>**Step-02:**<br>• Check the value of maximum deadline.<br>• Draw a Gantt chart where maximum time on Gantt chart is the value of maximum deadline.<br>**Step-03:**<br>• Pick up the jobs one by one.<br>• Put the job on Gantt chart as far as possible from 0 ensuring that the job gets completed before its deadline. | 05 | CO3 | L3 |
| | *********** All the Best*********** | | | |