Q1

W21

(a) Define algorithm. Discuss key characteristics of algorithms. 03

- A step-by-step procedure, to solve the different kinds of problems.
- An algorithm is any well-defined computational procedure that takes some value, or a set of values as input and produces some value, or a set of values as output.
- Characteristics: finiteness, input, output, optimization

(b) Explain why analysis of algorithms is important? Explain: Worst Case, Best Case and Average Case Complexity with suitable example. 04

- By analyzing some of the candidate algorithms for a problem, the most efficient one can be easily identified.
- Eg. linear search:
  - Worst Case (O(n)) ,
  - Best Case(O(1)) ,
  - Average Case (O(n))

(c) Write and analyze an insertion sort algorithm to arrange n items into ascending order. 07

- Greedy approach

```
insertion_sort(arr,num)

for i←0 ,i < length(arr) than i++
    minindex←i
    for j←i ,j< length(arr) than j++
        if arr[minindex] < arr[j] than
            minindex←j
    (now, swap elements indexed at minindex and ... than repeat the swapping for i=i+1 till it
    gets to minindex) or
    (
    for k←i, k<minindex than k++
            swap(k,minindex,arr)
    )
```

Best Case: $O(N^2)$, Average Case: $O(N^2)$, Worst Case: $O(N^2)$

W22

(a) Sort the best case running times of all these algorithms in a non-decreasing order. LCS, Quick-Sort, Merge-Sort, Counting-Sort, Heap-Sort, Selection-Sort, Insertion-Sort, Bucket-Sort, Strassen's Algorithm. **03**

- **Counting Sort**: Θ(n+k) , **Bucket Sort**: Θ(n) , **Insertion Sort**: Θ(n) , **Selection Sort**: Θ(n2) , **Quick Sort**: Θ(nlogn) , **Merge Sort**: Θ(nlogn) , **Heap Sort**: Θ(nlogn) , **Strassen's Algorithm**: Θ(nlog27)≈Θ(n2.81) , **Longest Common Subsequence (LCS)**: Θ(mn)

(b) State whether the statements are correct or incorrect with reasons.

1. O(f(n)) + O(f(n)) = O (2f(n))

2. If 3n + 5 = O(n2) , then 3n + 5 = o(n2) **04**

- 1. False: To make notation we remove any constant
- 2. False: Since f(n) = 3n+5 is close to n2 hence condition o(n2) fails.

(c) Explain asymptotic analysis with all the notations and its mathematical inequalities. **07**

- Asymptotic notation is a way to describe how the running time (or complexity) of an algorithm changes as the size of the input grows
    - O-Notation (Big O notation) (Upper Bound) (f(n) <= g(n))
    - o-Notation (Small o notation) (Strict-Upper Bound) (f(n) < g(n))
    - Ω-Notation (Omega notation) (Lower Bound) (f(n) >= g(n))
    - Ω-Notation (Small Omega notation) (Strict-Lower Bound) (f(n) > g(n))
    - θ-Notation (Theta notation) (Average order)

W23

(a) What is an algorithm? Explain various properties of an algorithm.RE **03**

(b) Solve the following using Master's theorem:

a. T(n) = 2T(n/4) + 1

b. T(n)=3T(n/4) + nlgn **04**

- **a. T(n) = 2T(n/4) + 1**

  Step 1 : a = 2 , b = 4 , f(n)=1

  Step 2 : now, $n^{\log_b a} = n^{\log_4 2} = n^{1/2}$

  Step 3 : f(n) = $n^{\log_b a - E}$

  Step 4 : T(n) = $\theta(n^{\log_b a}) = \theta(n^{1/2})$

- **b. T(n)=3T(n/4) + nlgn**

  Step 1 : a = 3 , b = 4 , f(n)=n(logn)

  Step 2 : now, $n^{\log_b a} = n^{\log_4 3} = n^{0.80}$

  Step 3 : f(n) = $n^{\log_b a+E}$

  Step 4 : T(n) = θ(f(n)) = θ(n(logn))

(c) Write selection sort algorithm and compute running time of algorithm. 07

- Greedy approach

```
Selection_sort(arr,num)

for i ← 0 ,i < length(arr) than i++
    minindex ← i
    for j ← i ,j< length(arr) than j++
        if arr[minindex] < arr[j] than
            minindex ← j
    (now, swap elements indexed at minindex and ... ) or
    (
            swap(i,minindex,arr)
    )
```

Best Case: $O(N^2)$, Average Case: $O(N^2)$, Worst Case: $O(N^2)$

S22

(a) Define Algorithm, Time Complexity and Space Complexity RE 03

(b) Explain: Worst Case, Best Case and Average Case Complexity with suitable example. RE 04

(c) Sort the following list using quick sort algorithm:< 5, 3 ,8 ,1 ,4 ,6 ,2 ,7 > Also write Worst and Best case and Average case of quick sort algorithm. 07

- Initial p = 7, i = 5 , j = 2.

5	3	8	1	4	6	2	7
i						j	p

- For i = 8 the condition fails for element at i<element at p hence for an element that doesn’t meet the condition element at j>element at p will be replaced.

5	3	2	1	4	6	8	7
		i				j	p

- Now i will keep getting incrimented till the element at i < element at p hence for j it will keep getting decrimented till element at j > element at p. but since i>j p will be swapped with i and its elements too.

5	3	2	1	4	6	8	7
					j	i	p

- Now element at the pivot p will be swapped with i+1 th element
- doing this we will get all element of left side which are smaller than p and on the other hand all elements which are on right side will be greater than p

5	3	2	1	4	6	7	8
					j	ip	

- Since for the right side of the pivot p there is only an element therefor it doent meet the condition i < j hence its complete

5	3	2	1	4	6	7
i					j	p

- Since i+1 = p therefor p = i.

5	3	2	1	4	6	7
					ijp	

- Since for the right side of the pivot p there is only an element therefor it doent meet the condition i < j hence its complete

5	3	2	1	4	6
i				j	p

- Since i+1 = p therefor p = i.

5	3	2	1	4	6
				ijp	

- Since for the right side of the pivot p there is only an element therefor it doent meet the condition i < j hence its complete

5	3	2	1	4
i			j	p

- For i = 5 the condition fails for element at i<element at p hence for an element that doesn’t meet the condition element at j>element at p will be replaced

1	3	2	5	4
i		j		p

- Now i will keep getting incrimented till the element at i < element at p hence for j it will keep getting decrimented till element at j > element at p. but since i>j p will be swapped with i and its elements too.

1	3	2	4	5
		j	ip	

- Since for the right side of the pivot p there is only an element therefor it doent meet the condition i < j hence its complete

1	3	2	4
i		j	p

- Since i+1 = p therefor p = i.

1	3	2	4

ijp

- Since for the right side of the pivot p there is only an element therefor it doent meet the condition i < j hence its complete

1	3	2

i j p

- The first condition which is about
- Now since i>j hence element at p index and I index will get swapped and p will become i

1	3	2

j ip

Now for both sides of p, left and right i<j doesn't meet the condition hence its sorted.

S23

(a) Define following terms:
(i) Big O Notation
(ii) Big Theta Notation
(iii) Big Omega Notation. RE **03**

(b) Perform Bucket sort for following sequence: 30, 12, 22, 66, 48, 27, 35, 43, 47, 41. **04**

- make array buckets of size 10.
- than store elemenets based on 0<x<10,10<x<20,20<x<30,30<x<40,40<x<5.
- after than sort elements in buckets than merge them orderly.

(c) Explain the bubble sort algorithm and derive its best case, worst case, and average case time complexity. **07**

- Greedy approach

```
Bubble_sort(arr,num)

for i ← 0 ,i < length(arr) than i++
    for j ← 0 ,j< length(arr)-1 than j++
        if arr[j+1] < arr[j] than
            swap(j,j+1,arr)
```

- Best Case: $O(N^2)$ (Already sorted)
- Average Case: $O(N^2)$
- Worst Case: $O(N^2)$ (Sorted in reverse order)

Q2

W21

(a) Write an algorithm of Selection Sort Method. RE 03
(b) Sort the following numbers using heap sort.20, 10, 50, 40, 30 RE 04
(c) Sort the following list using quick sort algorithm: <50, 40, 20, 60, 80, 100,45, 70, 105, 30, 90, 75> Also discuss worst and best case of quick sort algorithm. RE 07
OR (c) Apply merge sort algorithm on array A = {2,7,3,5,1,9,4,8}. What is time complexity of merge sort in worst case? 07

2	7	3	5
i			mid

1	9	4	8
mid+1			j

- Than it will get devided

2	7
i	mid

3	5
mid+1	j

1	9
i	mid

4	8
mid+1	j

- Than they will get devide too

2	7	3	5	1	9	4	8
i	j	i	j	i	j	i	j

- Since it has been devide now we can merge by sorting them

2	7
i	mid

3	5
mid+1	j

1	9
i	mid

4	8
mid+1	j

- Now we will merge two array by sorting them while merging them eg.

itteration1:	2,7	3,5	2
itteration2:	7	5	2,3
itteration3:	7		2,3,5
itteration4:			2,3,5,7

2	3	5	7
i			mid

1	4	8	9
mid+1			j

Again we will merge two array by sorting them while merging them.

1	2	3	4	5	7	8	9
i							j

- Time complexity=nlogn

W22

(a) What is the use of Loop Invariant? What should be shown to prove that an algorithm is correct? 03

- A loop invariant is a property of a program that remains unchanged throughout the execution of a loop. In other words, it is a condition that holds true before and after each iteration of the loop.
- To prove that algorithm is correct one can use Flags , one can trace the algo for few itterations , one can check whether outcome us equvalent to prediction or not etc...

(b) Apply LCS on sequence <A,B,A,C,B,C> for pattern <A,B,C> **04**

-	-	A	B	A	C	B	C
-	**0**	0	0	0	0	0	0
A	0	**1**	1	1	1	1	1
B	0	1	**2**	**2**	2	2	2
C	0	1	2	2	**3**	**3**	**3**

Answer : ABC is the lcs

(c) Write and explain the recurrence relation of Merge Sort. **07**

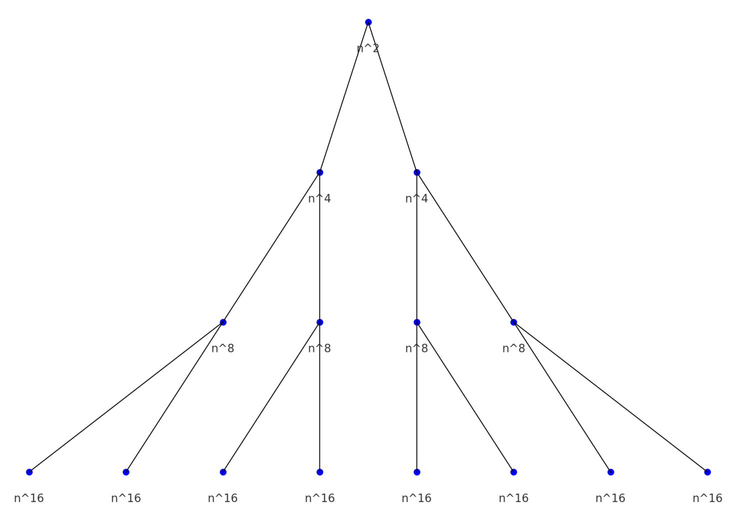
- In merge sort the array is being split into two parts than those two becomes branches and same operation is being perform on it. But after splitting it, it takes n itteration to merge it by sorting it.
- Hence a = 2 , b = 2 and f(n) = n
  f(n) = $n^{\log_b a + E}$
  therefor, n = $n^{\log_2 2 + E}$
  therefor, 0 = E
- *Hence* $T(n) = \theta\,(n^{\log_b a} * \log n) = \theta(n \log n)$

OR (c) Perform the analysis of a recurrence relation T(n)= 2T (n2) + θ(n2) by drawing its recurrence tree. **07**

- This recurrence relation is expanding with itteration there is no no stoping it.
- From this we can see that the function is being split into 2 parts.
- $T(n^2)$ that each splits has length of n^2 hence its exponentialy growing
- The n^2 at the end of the relation represent that each branch is processing n^2 itteration
- So its may look like this with different logic

```
def fun(n):
    for i in range(n*n): #function which is n^2
        print(i)
    fun(n*n) #split 1 with n**2 length of the original size
    fun(n*n) #split 2 with n**2 length of the original size
```

Recurrence Tree for T(n) = 2T(n^2) + Θ(n^2)



W23

(a) Explain general characteristics of greedy algorithms. **03**

- Immediate Decisions
- Assumes Optimality with Immediate Choices
- Applicability in Specific Scenarios
- Efficiency and Speed

(b) What is asymptotic notation? Find out big-oh notation of the f(n) = 3n2+5n+10 **04**

- $3n^2+5n+10 <= 3n^2+5n+10$
  $3n^2+5n+10 <= 3n^2+5n+n$
  $3n^2+5n+10 <= 3n^2+6n$
  $3n^2+5n+10 <= 3n^2+n^2$
  $3n^2+5n+10 <= 4n^2$ (for every n>= 7)

- Hence $O(3n^2+5n+10) = n^2$ (In order to make the form simpler $O(c*n^2)$ can be writtern as $O(n^2)$)

(c) Illustrate the working of the quick sort on input instance: 25, 29, 30, 35, 42, 47, 50, 52, 60. Comment on the nature of input i.e. best case, average case or worst case. Also discuss worst and best case of quick sort algorithm. RE **07**

OR (c) Give the properties of Heap Tree. Sort the following data using Heap Sort Method: 20, 50, 30, 75, 90, 60, 80, 25, 10, 40. **07**

S22

(a) Write an algorithm of Selection Sort Method. RE **03**

(b) Demonstrate Binary Search method to search Key = 14, form the array A=<2,4,7,8,10,13,14,60> **04**

- for key 14 in binaru search it will ½ the original length and check whether the soreted array's n/2 th element is 14 or not if it is than it will return that n/2 else if that element is smaller than key than it will search on right side by (n+n/2)/2 th element and so on else if its larger than it will search in left side by (0+n/2)/2 and so on…
- In this n/2 the element is 8 which is smaller than 14 hence it will do (n/2 + n)/2 th element and compare it.
- (4+8) / 2 th term is 13 which is still smaller than 14 hence again it will do (6+8)/2 th element and key's equality check.
- 7 th element is 14 which is equal to key hence it will return 7 index as answer.

(c) Write the Master theorem. Solve following recurrence using it.
(i)T(n)= T(n/2) + 1
(ii) T(n)=2T(n/2) + n log n **07**

- (i)T(n)= T(n/2) + 1
  - step1: a=1 , b=2 , f(n) = 1
  - step2: $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$
  - step3: f(n) = $n^{\log_b a+E}$ hence E=0
  - step4: T(n) = $\theta(n^{\log_b a} * \log n) = \theta(1 * \log n) = \theta(\log n)$

- (ii) T(n)=2T(n/2) + n log n
  - step1: a=2 , b=2 , f(n) = n log n
  - step2: $n^{\log_b a} = n^{\log_2 2} = n^1 = n$
  - step3: f(n) = $n^{\log_b a+E}$ hence E>0
  - step4: T(n) = $\theta(f(n)) = \theta(n\log n)$

OR (c) Solve following recurrence relation using iterative method T(n) = T(n - 1) + 1 with T(0) = 0 as initial condition. Also find big oh notation **07**

- Substitution method
- T(n) = T(n - 1) + 1
- T(n-1) = T((n-1) - 1) + 1 = T(n - 2) + 1 by putting this equation in T(n)
- T(n) = T(n - 2) + 1 + 1= T(n - 2) + 2
- T(n-2) = T((n-2) - 1) + 1 = T(n - 3) + 1 by putting this equation in T(n)
- T(n) = T(n - 3) + 1 + 2 = T(n - 3) + 3
- T(n-k) = T((n-k) - 1) + 1 = T(n – k - 1) + 1 by putting this equation in T(n)
- T(n) = T(n – k - 1) + k+1
- By assuming k is such a value that k+1 is the n th term hence,
- T(n) = T(n – (k + 1)) + (k+1)
- T(n) = T(n – (n)) + (n)
- T(n) = 0+ (n)
- T(n) = n

- Iterative method
- T(n) = T(n-1)+1
- T(0) = 0
- T(1) = T(0)+1 = 1
- T(2) = T(1)+1 = 2
- T(3) = T(2)+1 = 3
- T(4) = T(3)+1 = 4
- T(n) = T(n-1)+1 = n-1+1 = n
- hence for this the big Oh is O(n)

(a) Define Algorithms and characteristics of algorithms. RE 03
(b) What is a recurrence? Solve recurrence equation for T (n) =T (n-1) + 1 using substitution method. RE 04
(c) Discuss Binary search algorithm, also write and solve its recurrence relation. RE 07
OR (c) Explain Merge Sort algorithm with suitable example. RE 07

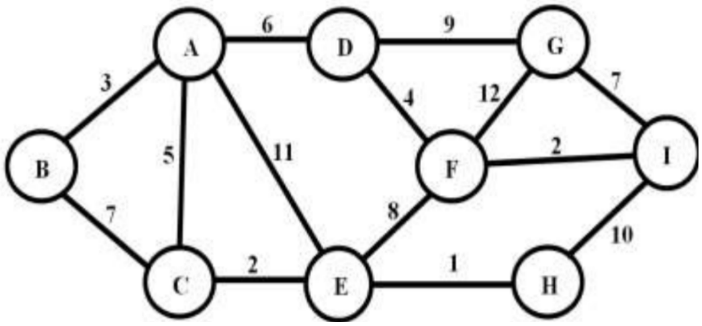
Q3

W21

(a) What is Principle of Optimality? Explain its use in Dynamic Programming Method 03
(b) Explain Binomial Coefficient algorithm using dynamic programming. 04
(c) Solve the following 0/1 Knapsack Problem using Dynamic Programming. There are five items whose weights and values are given in following arrays. Weight w [] = {1,2,5,6,7} Value v [] = {1, 6, 18, 22, 28} Show your equation and find out the optimal knapsack items for weight capacity of 11 units. 07
OR (a) Compare Dynamic Programming Technique with Greedy Algorithms 03
OR (b) Give the characteristics of Greedy Algorithms. 04
OR (c) Obtain longest common subsequence using dynamic programming. Given A = “acabaca” and B = “bacac”. 07

W22

(a) Consider the array 2,4,6,7,8,9,10,12,14,15,17,19,20. Show (without actually sorting), how the quick sort performance will be affected with such input. RE 03
(b) "A greedy strategy will work for fractional Knapsack problem but not for 0/1", is this true or false? Explain. ANS TRUE 04
(c) Apply Kruskal’s algorithm on the given graph and step by step generate the MST. 07



OR (a) Consider an array of size 2048 elements sorted in non-decreasing order. Show how the Binary Search will perform on this size by analysis of its recurrence relation. Derive the running time. 03
OR (b) Explain the steps of greedy strategy for solving a problem. 04
OR (c) Apply Prim's algorithm on the given graph in Q.3 (C) FIG:1 Graph G(V,E) and step by step generate the MST. 07

W23

(a) Sort the List "G,U,J,A,R,A,T,S,A,R,K,A,R" in alphabetical order using merge sort.RE 03
(b) Following are the details of various jobs to be scheduled on multiple processors such that no two processes execute on the same processor. Show schedule of these jobs on minimum number of processors using greedy approach.

Jobs	J1	J2	J3	J4	J5	J6	J7
Start time	0	3	4	9	7	1	6
Finish time	2	7	7	11	10	5	8

04
(c) Using algorithm find an optimal parenthesization of a matrix chain product whose sequence of dimension is (5,10,3,12,5,50,6) (use dynamic programming). 07
OR (a) Apply counting sort for the following numbers to sort in ascending order.3, 1, 2, 3, 3, 1 03
(b) Find the Optimal Huffman code for each symbol in following text ABCCDEBABFFBACBEBDFAAAABCDEEDCCBFEBFCAE 04
(c) Solve following knapsack problem using dynamic programming algorithm with given capacity W=5, Weight and Value are as follows (2,12),(1,10),(3,20),(2,15) RE 07

S22

(a) What is Principle of Optimality? Explain its use in Dynamic Programming Method RE 03
(b) Find out LCS of A={K,A,N,D,L,A,P} and B = {A,N,D,L} RE 04
(c) Discuss Assembly Line Scheduling problem using dynamic programming with example.07

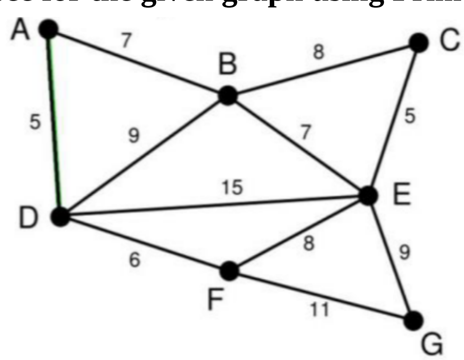
OR (a) Give the characteristics of Greedy Algorithms RE 03
OR (b) Give difference between greedy approach and dynamic programming. RE 04
(c) Consider Knapsack capacity W=15, w = (4, 5, 6, 3) and v=(10, 15, 12, 8) find the maximum profit using greedy method. 07

S23

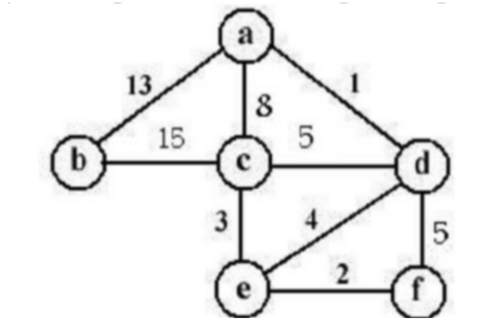
(a) Explain principle of optimality with suitable example. RE 03
(b) Explain advantages and disadvantages of dynamic programming. 04
(c) Given the denominations: d1=1, d2=4, d3=6. Calculate for making change of Rs. 8 using dynamic programming. 07
OR (a) Explain Weighted Graph, Undirected Graph, Directed Graph. RE 03
OR (b) Discuss advantages and disadvantages of greedy algorithm. 04
OR (c) Consider weights w=(3,4,6,5) and profit v=(2,3,1,4) and Knapsack capacity W=8. Find the maximum profit using dynamic approach. RE 07

Q4

W21
(a) Using greedy algorithm find an optimal schedule for following jobs with n=7 profits: (P1, P2, P3, P4, P5, P6, P7) = (3, 5, 18, 20, 6, 1, 38) and deadline (d1, d2, d3, d4, d5, d6, d7) = (1, 3, 3, 4, 1, 2, 1) 03
(b) Find Minimum Spanning Tree for the given graph using Prim's Algo. RE 04



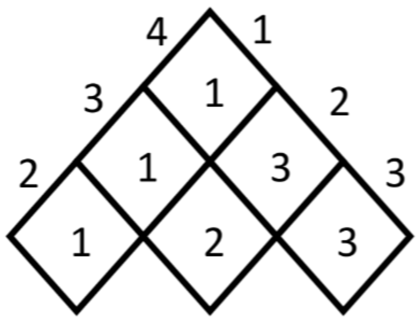
(c) Explain in brief Breadth First Search and Depth First Search Traversal techniques of a Graph with Example. 07
OR (a) Find an optimal Huffman code for the following set of frequency.
A : 50, b: 20, c: 15, d: 30 03
OR (b) Find Minimum Spanning Tree for the given graph using Kruskal Algo. RE 04



OR (c) Explain Backtracking Method. What is N-Queens Problem? Give solution of 4- Queens Problem using Backtracking Method **07**

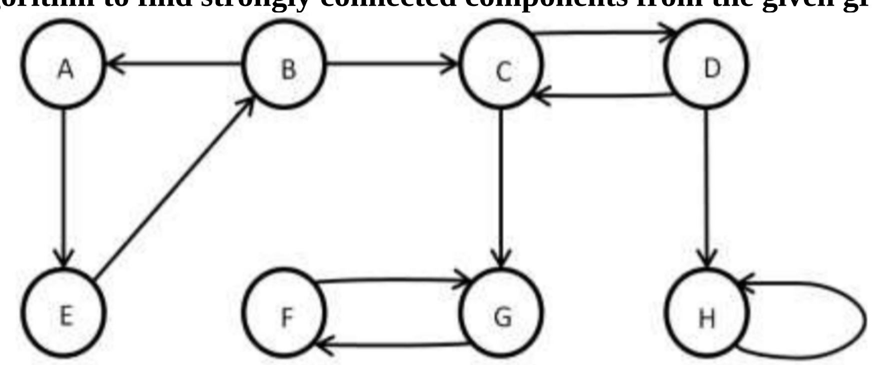
W22

(a) Given is the S-table after running Chain Matrix Multiplication algorithm. Calculate the parenthesized output based on PRINT_OPTIMAL_PARENTHESIS algorithm. Assume the matrix are names from A1, A2, ….,An **03**



(b) Explain states, constraints types of nodes and bounding function used by backtracking and branch and bound methods. **04**

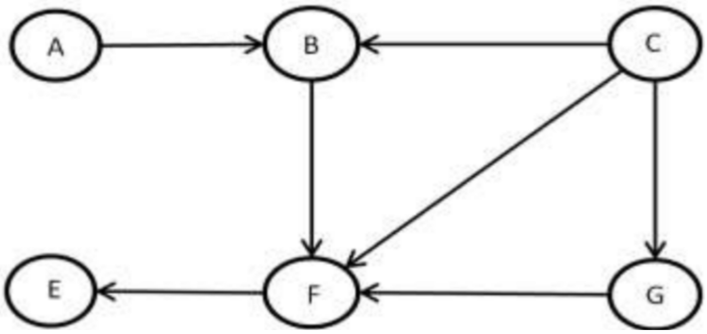
(c) Apply the algorithm to find strongly connected components from the given graph. **07**



OR (a) Consider a Knapsack with maximum weight capacity M is 7, for the three objects with value <3, 4, 5> with weights <2, 3, 4> solve using dynamic programming the maximum value the knapsack can have. RE **03**

OR (b) Explain the Minimax principle and show its working for simple tic-tac-toe game playing. **04**

OR (c)Given is the DAG, apply the algorithm to perform topological sort and show the sorted graph. **07**



W23

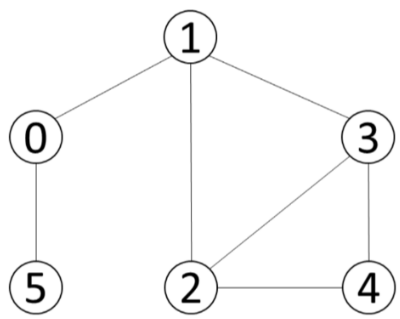
(a) Solve the following Task Assignment problem for minimization using following cost matrix. (Cost matrix represents cost of Task T performed by Person P).

	T1	T2	T3
P1	10	20	25
P2	20	23	26
P3	12	16	25

03

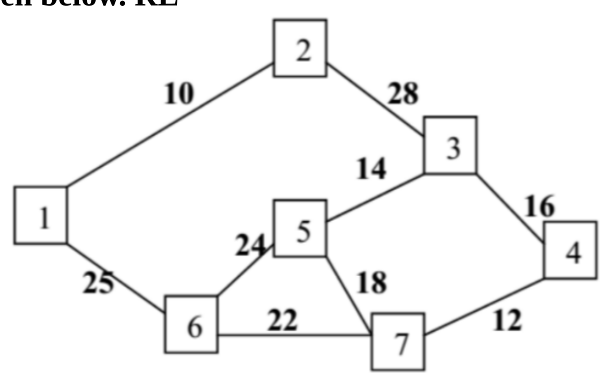
(b) Given coins of denominations 2, 3 and 4 with amount to be pay is 5. Find optimal no. of coins and sequence of coins used to pay given amount using dynamic method. 04

(c) Write an algorithm to find out the articulation points of an undirected graph. Find out articulation points for the following graph. Consider vertex 0 as the starting point. 07



OR (a) Find out the NCR ($^5C_3$) Using Dynamic Method. 03

OR (b) Write the Kruskal's Algorithm to find out Minimum Spanning Tree. Apply the same and find MST for the graph given below. RE 04

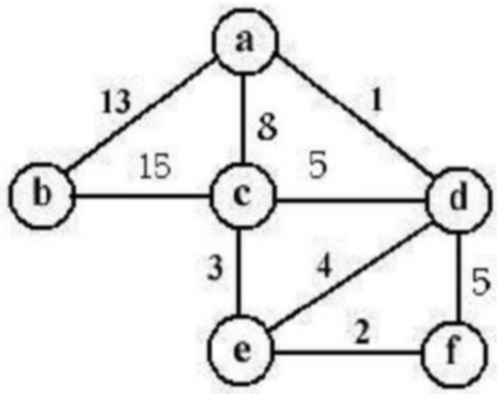


OR (c) Explain Backtracking Method. What is N-Queens Problem? Give solution of 4-Queens Problem using Backtracking Method. 07

S22

(a) Explain: Articulation Point, Graph, Tree RE 03

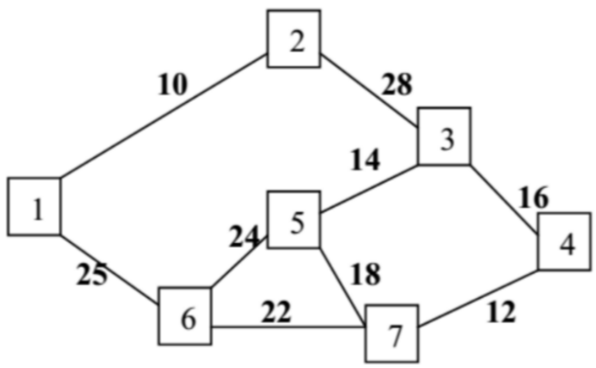
(b) Find Minimum Spanning Tree for the given graph using Prim's Algorithm. RE **04**



(c) Explain Breath First Traversal Method for Graph with algorithm with example.RE **07**

OR (a) Explain Huffman code with Example. **03**

OR (b) Write the Kruskal's Algorithm to find out Minimum Spanning Tree. Apply the same and find MST for the graph given below RE **04**



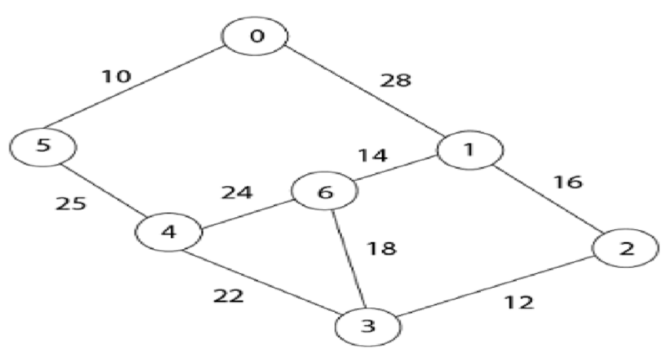
OR (c) Explain fractional knapsack problem with example. **07**

S23

(a) Find an optimal Huffman code for the following set of frequency. a: 40, b: 20, c: 15, d: 30, e: 10. RE **03**

(b) Explain depth first traversal using suitable example. RE **04**

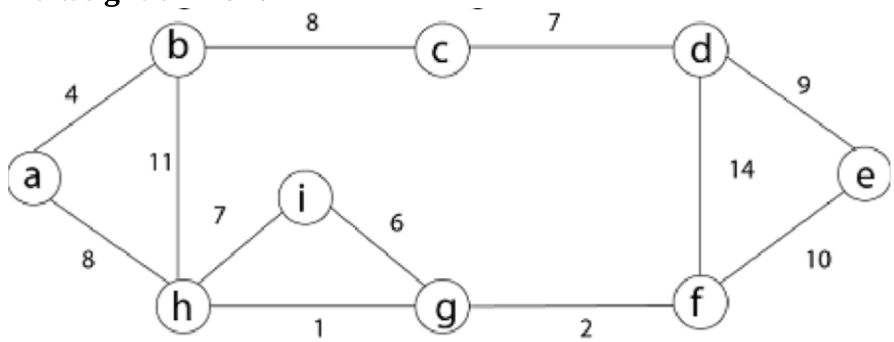
(c) Draw the minimum spanning tree correspond to following graph using Prim's algorithm and find the MST weight: RE **07**



OR (a) Differentiate between Kruskal's algorithm and Prim's algorithm for finding MST. RE **03**

OR (b) Explain the need of topological Sort with example. **04**

OR (c) Draw the minimum spanning tree correspond to following graph using Kruskal's algorithm and find weight of MST: RE **07**



Q5

W21

(a) Define Articulation point, Acyclic Directed Graph, Back Edge **03**

(b) Show the comparisons that naïve string matcher makes for the pattern p=0001 in the text T=000010001010001 **04**

(c) Explain spurious hits in Rabin-Karp string matching algorithm with example. Working modulo q=13, how many spurious hits does the Rabin- Karp matcher encounter in the text T = 2359023141526739921 when looking for the pattern P = 31415? **07**

OR (a) Explain polynomial reduction. **03**

OR (b) Differentiate branch and bound and back tracking algorithm. **04**

OR (c) Explain P, NP, NP complete and NP-Hard problems. Give examples of each **07**

W22

(a) When can we say that a problem exhibits the property of Optimal Sub-structure? **03**

(b) Create an example of string P of length 7 such that, the prefix function of KMP string matcher returns π[5] = 3, π[3] = 1 and π[1] = 0 **04**

(c) Explain the 3SAT problem and show that it is NP Complete. **07**

OR (a) Explain Over-lapping Sub-problem with respect to dynamic programming. **03**

OR (b) Show that if all the characters of pattern P of size m are different, the naïve string matching algorithm can perform better with modification. Write the modified algorithm that performs better than O(n.m). **04**

OR (c) Explain with example, how the Hamiltonian Cycle problem can be used to solve the Travelling Salesman problem. **07**

W23

(a) Demonstrate Binary Search method to search Key = 14, form the array A = <2,4,7,8,10,13,14,60>. RE **03**

(b) Solve the following knapsack problem using greedy method. Number of items = 5, knapsack capacity W = 100, weight vector = {50,40,30,20,10} and profit vector = {1,2,3,4,5}. RE **04**

(c) Define P, NP, NP-complete, NP-Hard problems. Give examples of each **07**

OR (a) Explain in Brief: Polynomial reduction. **03**

OR (b) Traverse the following graph using Breadth First Search Technique. Also draw BFS Tree for a given graph. RE 04
OR (c) Explain spurious hits in Rabin-Karp string matching algorithm with example. Working modulo q=13, how many spurious hits does the Rabin-Karp matcher encounter in the text T = 2359023141526739921 when looking for the pattern P = 26739? 07

S22

(a) What is string-matching problem? Define valid shift and invalid shift. 03
(b) Define P, NP, NP-Hard and NP-Complete Problem RE 04
(c) Explain Backtracking Method. What is N-Queens Problem? Give solution of 4- Queens Problem using Backtracking Method. RE 07

OR (a) Explain "P = NP ?" problem. 03
OR (b) Explain Minimax principal. 04
OR (c) What is Finite Automata? Explain use of finite automata for string matching with suitable example. 07

S23

(a) Explain Spurious hits with an example. 03
(b) Write the pseudocode for Naïve String-Matching Algorithm. RE 04
(c) What is state space tree. How do you solve the Eight queens problem using backtracking with the help of state space tree. 07
OR (a) Explain polynomial time reduction. RE 03
OR (b) Differentiate between Backtracking and Branch-and-Bound algorithms. RE 04
OR (c) Define P, NP, NP complete and NP-Hard problems. Give examples of each. RE 07