UNIT - III

SYLLABUS

Divide and conquer basic strategy, binary search, quick sort, merge sort, matrix operations, Multiplication Algorithm Greedy method - basic strategy, Knapsack problem, application to job sequencing with deadlines problem, minimum cost spanning trees, single source shortest path, optimal search patterns.

UNIVERSITY PAPER SOLUTIONS
SINCE SUMMER - 2006

Note to the students :

- The questions given below are from previous Nagpur University examination papers.
- Though these questions are from old university papers, they have been included in the new syllabus.
- Questions which are out of new syllabus are not included here.

SUMMER - 06 (CT)

Q.1. Show that quicksort's best-case running time is Ω (n/gn). **7M**

Ans. **P.3-18, Q.12.**

Q.2. Illustrate the stepwise operation of Heap sort on the input array,

A = <5, 13, 2, 25, 7, 17, 20, 8, 4 >

Also find the recurrence relation for the algorithm and find its complexity. **9M**

Ans. **P.3-26, Q.24.**

Q.3. Explain the Divide and Conquer strategy. How does binary search fit into the strategy? **4M**

Ans. **P.3-10, 15, Q.1 & Q.9.**

Q.4. Discuss the job scheduling problem in brief. **6M**

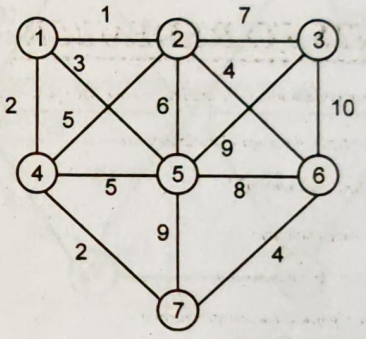
Ans. **P.3-43, Q.50.**

Q.5. What is minimum cost spanning trees? Write an algorithm for any one method for finding the minimum cost spanning tree. Also discuss its complexity. **7M**

Ans. **P.3-45, 46, Q.55 & Q.56.**

WINTER - 06 (CT)

Q.1. What is minimum cost spanning tree? Using Kruskal's algorithm find the minimum cost spanning tree for the following graph. Show how you have selected the edges. **4M**



Ans. **P.3-45, 48, Q.55 & Q.61.**

Q.2. Modify the binary search such the input is divided into three equal parts in the search procedure. Write down the algorithm and find its time complexity. **6M**

Ans. **P.3-19, 20, Q.15 & Q.16.**

Q.3. Explain Divide and Conquer method. Justify how Merge-sort fits into this algorithm design method. **7M**

Ans. **P.3-10, 19, Q. 1 & Q. 15.**

Q.4. Write the algorithm for quick sort to sort the data in descending order. Find out the time complexity function T(n). Also give the best case, average case and worst case of the algorithm. **8M**

Ans. **P.3-16, 18, Q.10, Q.11, Q.12 & Q.13.**

Q.5. Discuss Prim's minimum cost spanning tree with an example. **5M**

Ans. **P.3-50, Q.65.**

Q.6. Give the greedy algorithm for job sequencing with deadlines. Also give its time complexity. **8M**

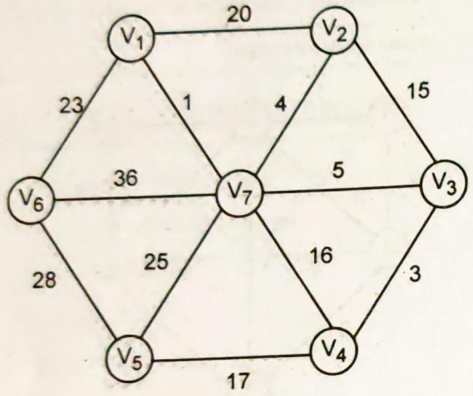
Ans. **P.3-44, Q.52.**

SUMMER - 07 (CT)

Q.1. Write an recursive algorithm to implement merge sort technique. Derive its recurrence relation and solve to obtain the time complexity. **13M**

Ans. **P.3-20, Q.16.**

Q.2. Explain how the concept of greedy method is used to find out minimum cost spanning tree. Give an algorithm to find minimum cost spanning tree using Kruskal's algorithm. Discuss the time complexity. Obtain the minimum cost spanning tree for the given graph. **13M**



Ans. **P.3-46, 49, Q.56 & Q.62.**

WINTER - 07 (CT)

Q.1. Give an algorithm to sort an array using m quicksort method. Obtain its recurrence relation. Analyse the algorithm for best case, average case and worst case. **6M**

Ans. **P.3-16, 18, Q.10, Q.11, Q.12 & Q.13.**

Q.2. Give general characteristics of greedy algorithm and skeleton algorithm. Give greedy algorithm for knapsack problem. Work out the algorithm step by step for the given problem and find out the optimal solution.

Problem : Assume the maximum capacity of a knapsack m = 20, n = 3, (p1, p2, p3) = (25, 24, 15) and (w1, w2, w3) = (18, 15, 10). **13M**

Ans. **P.3-37, 39, 40, Q.36, Q.43 & Q.45.**

SUMMER - 08 (CT)

Q.1. Show that the best case running time complexity of quick sort is Ω (n lg n). **7M**

Ans. **P.3-18, Q.12.**

Q.2. Illustrate the stepwise operation of Heapsort for the input < 8, 20, 17, 7, 25, 2, 13, 5 >. **7M**

Ans. **P.3-25, Q.21.**

Q.3. Give Kruskal's algorithm for the same. Also obtain its time complexity. **4M**

Ans. **P.3-46, Q.56.**

WINTER - 08 (CT)

Q.1. Write program segment which has the time complexity :

(i) O(n log n).

(ii) O(log n). **4M**

Ans. **P.3-12, 18, Q.3 & Q.13.**

Q.2. Explain the quicksort algorithm with suitable example. Find out the time complexity of the algorithm. **6M**

Ans. **P.3-16, Q.10.**

Q.3. Write an algorithm for Merge Sort. Give stepwise run of Merge Sort algorithm when supplied with the input.

(30, 28, 17, 65, 35, 42, 20, 45, 52). **6M**

Ans. **P.3-19, Q.15.**

Q.4. Explain Greedy algorithm. Write down the characteristics of Greedy algorithm. **4M**

Ans. **P.3-37, Q.36.**

Q.5. Discuss the job sequencing problem in brief. **4M**

Ans. **P.3-43, Q.50.**

Q.6. What is minimum cost spanning trees ? Write an algorithm for Kruskal's method for finding the minimum cost spanning tree. Also discuss its complexity. **7M**

Ans. **P.3-45, 46, Q.55 & Q.56.**

SUMMER - 09 (CT)

Q.1. State the recurrence relation for the best case and worst case of quicksort algorithm. Obtain the worst case time complexity using recurrence tree. **7M**

Ans. **P.3-16, 18, Q.10 & Q.11.**

Q.2. Write an algorithm to merge the two arrays, which are in sorted order. Discuss the time complexity. **6M**

Ans. **P.3-19, 20, Q.15 & Q.16.**

Q.3. Give an algorithm to find the minimum cost spanning tree using Kruskal's method. Discuss the time complexity. **7M**

Ans. **P.3-46, Q.56.**

Q.4. Given the denominations of Rs. 100, Rs. 50, Rs. 20, Rs. 10, Rs. 05, Rs. 02 and Rs. 01. Design an algorithm using Greedy approach to find the number of notes of each denomination for a sum of 'n' rupee. **6M**

Ans. **P.3-38, Q.40.**

Q.5. Write an algorithm to search an element using binary search method. Give the recurrence relation and solve the same. **7M**

Ans. **P.3-12, Q.3.**

Q.6. Define heap tree. Construct a heap tree for each of the following numbers, if added in the given order :

1, 15, 2, 14, 20, 11, 7, 9, 5, 6. **6M**

Ans. **P.3-25, Q.22.**

WINTER - 09 (CT)

Q.1. Explain how an array of 'n' elements can be sorted using divide and conquer technique. Give the algorithm and obtain its recurrence relation for the best case and worst case. **7M**

Ans. **P.3-11, Q.2.**

Q.2. Discuss the general characteristics of a greedy algorithm and give the general format. **6M**

Ans. **P.3-37, Q.36.**

Q.3. Design an algorithm using the Greedy approach for a job scheduling problem with deadline. Work out the algorithm for the given below six jobs to obtain the maximum profit and respecting the deadlines : **7M**

Job No.	gi (Profit)	di (Dead line)
1	20	3
2	15	1
3	10	1
4	7	3
5	5	1
6	3	3

Ans. **P.3-44, Q.52.**

Q.4. Give an algorithm to find the shortest distance using greedy approach. **6M**

Ans. **P.3-56, Q.73.**

Q.5. Write an algorithm to add a node in a heap tree such that the tree remains a heap tree after insertion. **6M**

Ans. **P.3-26, Q.23.**

Q.6. Write an algorithm ternary-search which searches an element in an array by dividing the array in three equal parts instead of two halves as in case of binary search. Give the recurrence relation. **7M**

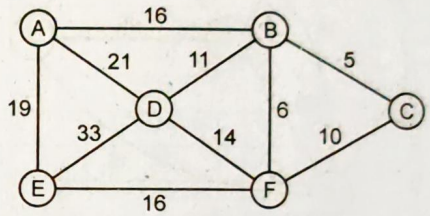
Ans. **P.3-19, Q.15.**

SUMMER - 10 (CT)

Q.1. Explain quicksort algorithm with suitable example. Comment on the time complexity of the algorithm. **7M**

Ans. **P.3-16, Q.10.**

Q.2. Compute a minimum cost spanning tree for the following graph using Kruskal algorithm and explain its complexity. **7M**



Ans. **P.3-47, Q.58.**

Q.3. Find an optimal solution to the Knapsack instance

$n = 7,\ m = 15,$

$(p_1, p_2, p_3 \ldots\ldots p_7) = (10, 5, 15, 7, 6, 18, 3)$ and

$(w_1, w_2, \ldots\ldots w_7) = (2, 3, 5, 7, 1, 4, 1).$ **8M**

Ans. **P.3-41, Q.47.**

Q.4. Explain the job scheduling problem with example. **8M**

Ans. **P.3-43, Q.50.**

SUMMER - 10 (CS)

Q.1. Explain analysis of binary search and discuss its complexity for an array of size 10. Write recursive algorithm for binary search. **8M**

Ans. **P.3-13, Q.4.**

Q.2. Write insert and adjust algorithm of heap-sort. **5M**

Ans. **P.3-26, Q.23.**

Q.3. Find optimal solution to the Knapsack instance $n = 7, m = 15$:

$(P_1, P_2, \ldots\ldots, P_7) = (15, 20, 10, 7, 6, 18, 3)$ and

$(w_1, w_2, \ldots, w_7) = (2, 3, 5, 7, 1, 4, 1)$. **8M**

Ans. **P.3-42, Q.48.**

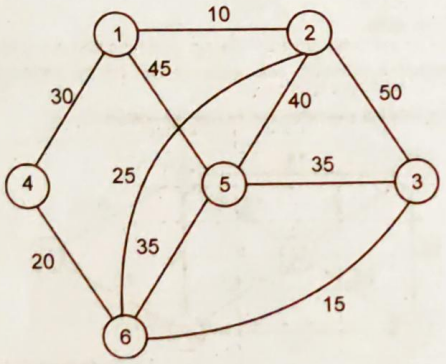
Q.4. Explain Greedy method based algorithm for job sequencing with deadline problem. **5M**

Ans. **P.3-44, Q.52.**

Q.5. Write algorithm for finding minimum maximum element from array using DAC. **6M**

Ans. **P.3-36, Q.33.**

Q.6. Find minimum cost spanning tree for following graph using prim's algorithm. **7M**



Ans. **P.3-51, Q.66.**

Q.7. Draw merge and split tree for the following sequence. Write recursive merge sort and also explain its recurrence equation. How merge sort is different from quick sort? **13M**

210 495 325 125 280 861 652

730 423 148 351 176 481 530

Ans. **P.3-20, 21, Q.16 & Q.17.**

WINTER - 10 (CT)

Q.1. What is the best possible sequence generated by the job scheduling approach when n = 7,

$(P_1, P_2, \ldots, P_7) = (3, 5, 20, 18, 1, 6, 30)$

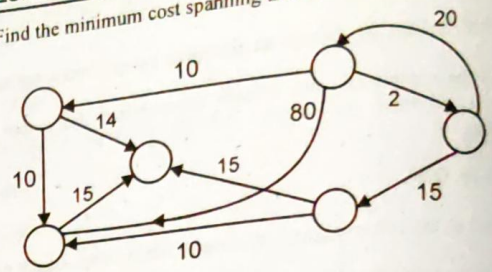
and $(d_1, d_2, \ldots d_7) = (1, 3, 4, 3, 2, 1, 2)$. **7M**

Ans. **P.3-43, Q.51.**

Q.2. Explain job scheduling problem with example and comment on complexity of algorithm. **7M**

Ans. **P.3-43, Q.50.**

Q.3. Find the minimum cost spanning tree for the following graph. **6M**



Ans. **P.3-49, Q.63.**

WINTER - 10 (CS)

Q.1. Write insert, delete and adjust algorithms used in Heap Sort. Also explain the complexity of Heap Sort. **7M**

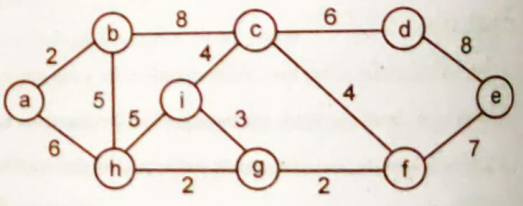
Ans. **P.3-26, Q.23 & Q.24.**

Q.2. Write an algorithm based on DAC approach to find minimum and maximum element from given array. Explain the complexity of algorithm. Implement the algorithm on following array and draw min-max tree.

123, 82, 25, – 20, – 62, 43, 172, 95, 57, – 45. **5M**

Ans. **P.3-36, Q.33.**

Q.3. Write PRIM's algorithm to generate spanning tree. Implement the algorithm on following graph. Explain the complexity. **8M**



Explain the steps in execution.

Ans. **P.3-50, Q.65.**

Q.4. What is the significance of Knapsack problem? Implement three approaches on following objects and find out the profit value. Also write algorithm for best approach. Total capacity = 16. Number of objects = 7. **9M**

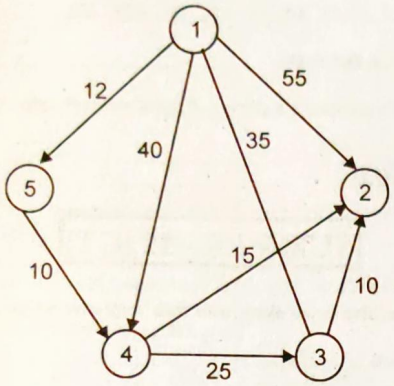
i	1	2	3	4	5	6	7
Pi	10	15	12	4	6	16	8
Wi	2	4	5	4	2	3	3

Ans. **P.3-39, 40, Q.43 & Q.44.**

Q.5. Write an algorithm for 'ternary' search. Explain how it is different from binary search. **5M**

Ans. **P.3-19, Q.15.**

Q.6. Write Greedy based single source shortest path algorithm. Implement the algorithm on following graph and explain execution. **6M**



Ans. P.3-56, 57, Q.73 & Q.75.

SUMMER - 11 (CT)

Q.1. Explain the divide and conquer strategy. How does quick sort fit into the strategy? **5M**

Ans. P.3-10, 16, Q.1 & Q.10.

Q.2. Explain what is Knapsack problem. Give algorithm for it and time complexity for the algorithm. **6M**

Ans. P.3-39, Q.43.

Q.3. Solve following Knapsack problem.

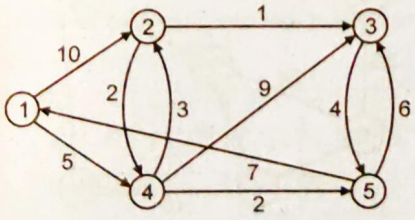
$n = 3, m = 20$

$(P_1, P_2, P_3) = (25, 24, 15)$

$(W_1, W_2, W_3) = (18, 15, 10)$. **3M**

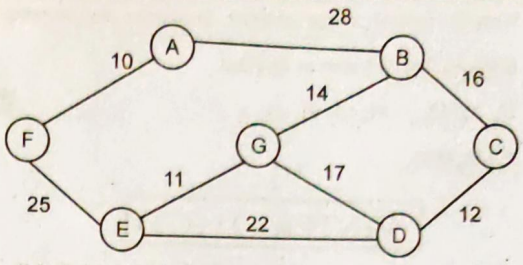
Ans. P.3-40, Q.45.

Q.4. Find single source shortest path for the following diagram. Write the algorithm for same. **8M**



Ans. P.3-56, Q.73 & Q.74.

Q.5. What is minimum cost spanning tree? Using prim's algorithm find minimum-cost spanning tree for following graph. Show how edges are selected. **8M**



Ans. P.3-45, 52, Q.55 & Q.67.

SUMMER - 11 (CS)

Q.1. Write an algorithm for Heap Sort and Heaps Implement a heap for the set (1, 6, 9, 2, 7, 5, 2, 7, 4, 10) **6M**

Ans. P.3-23, Q.20.

Q.2. Explain job scheduling approaches. Find the best possible sequence for the following deadlines :

Job	Gain	Deadline
1	35	3
2	20	1
3	18	3
4	16	4
5	12	2
6	10	2
7	8	1

6M

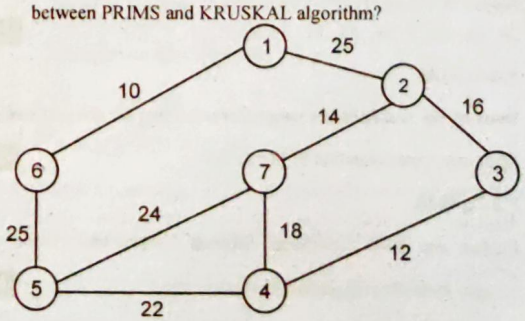
Ans. P.3-43, 44, Q.50, Q.53.

Q.3. Show the Strassen's matrix multiplication process on the matrix A and B given below :

$$A = \begin{bmatrix} 4 & 2 & 0 & 1 \\ 3 & 1 & 2 & 5 \\ 3 & 2 & 1 & 4 \\ 5 & 2 & 6 & 7 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 5 & 4 & 2 & 3 \\ 1 & 4 & 0 & 2 \\ 3 & 2 & 4 & 1 \end{bmatrix}.$$ **7M**

Ans. P.3-34, Q.31.

Q.4. Write algorithm to find out minimum cost spanning tree for the following graph, using PRIMS algorithm. What are the difference between PRIMS and KRUSKAL algorithm? **8M**



Ans. P.3-46, Q.57.

Q.5. What is optimal merge pattern? Implement the merging on following files with sizes as specified.

28, 32, 12, 5, 84, 53, 91, 35, 3, 11. **4M**

Ans. **P.3-59, Q.80.**

WINTER - 11 (CT)

Q.1. What are the principles of greedy algorithm techniques? Discuss one example with its algorithm based on technique. **7M**

Ans. **P.3-37, 38, Q.36 & Q.37.**

Q.2. What is divide and conquer strategy? Using this strategy write an algorithm to search a given element in a sorted array. Also find its time complexity. **6M**

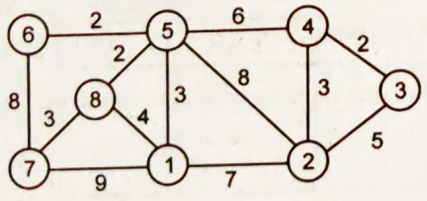
Ans. **P.3-10, 12, Q.1 & Q.3.**

Q.3. Write down the merge sort algorithm and find its time complexity. Can you improve the time complexity? What will be the pros and cons if the sort file is very large? **8M**

Ans. **P.3-19, 20, Q.15 & Q.16.**

WINTER - 11 (CS)

Q.1. What is the use of spanning tree? Draw the spanning tree for following graph using PRIM's method. Write an algorithm for PRIM's method and explain the execution steps with complexity. **9M**



Ans. **P.3-50, Q.64 & Q.65.**

Q.2. What is optimal merge pattern? Implement the merging on following files with sizes as specified.

28, 32, 12, 5, 84, 53, 91, 35, 3, 11. **4M**

Ans. **P.3-59, Q.80.**

Q.3. Draw Merge and Split tree using Merge Sorting for array of size 15. Write recurrence equation of Merge Sort. **5M**

Ans. **P.3-22, Q.18.**

Q.4. Explain any three differences between Greedy and Divide and conquer methods of algorithm design. **3M**

Ans. **P.3-38, Q.38.**

Q.5. Explain the complexity of Binary Search. Implement Binary Search on following array and find average no. of comparisons required for successful and unsuccessful search.

– 12, – 4, 9, 32, 50, 79, 109, 135, 203, 230. **5M**

Ans. **P.3-12, 14, Q.3 & Q.5.**

Q.6. Explain worst case complexity of quick sort with suitable example. **4M**

Ans. **P.3-18, Q.11.**

SUMMER - 12 (CT)

Q.1. Consider five items along with their respective weights and values as follows :

Item	I_1	I_2	I_3	I_4	I_5
Weight	05	10	20	30	40
Value	30	20	100	90	160

The capacity of knapsack is 60.

Find the solution for fractional knapsack problem using greedy approach. **7M**

Ans. **P.3-41, Q.46.**

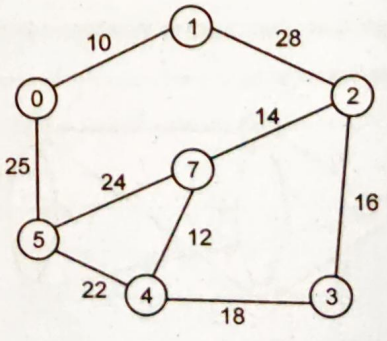
SUMMER - 12 (CS)

Q.1. In the given array, design split tree and merge tree. Assume 14 elements is given in an array as follows : **5M**

7, 8, 14, 10, 2, 6, 18, 11, 21, 19, 12, 14, 28, 23.

Ans. **P.3-23, Q.19.**

Q.2. Implement Prim's minimum cost spanning tree algorithm in the following graph. **8M**



Ans. **P.3-46, Q.57.**

Q.3. Generate a Huffman code for the following 6 characters given below: **6M**

Symbol	Frequency
a	45
b	13
c	12
d	16
e	09
f	05

Ans. **P.3-59, Q.79.**

Q.4. For the following sequence of jobs, give the snapshot of execution which will achieve maximum profit : **5M**

Jobs	Gain	Deadline
1	3	1
2	5	3
3	20	4
4	18	3
5	1	2
6	6	1
7	30	2

Ans. **P.3-43, Q.51.**

Q.5. Sort the given array using heap sort algorithm. Write algorithm and also explain the complexity of heap sort algorithm.

4, 1, 3, 2, 16, 9, 10, 14, 8, 7 **9M**

Ans. **P.3-23, 30, Q.20 & Q.26.**

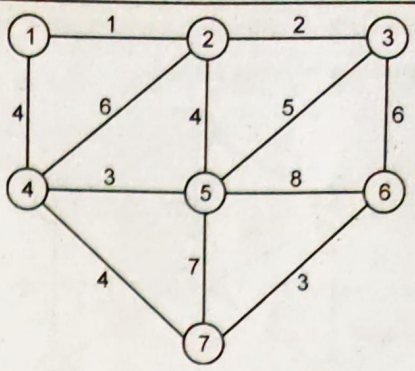
WINTER - 12 (CT)

Q.1. Explain greedy algorithm for sequencing unit time jobs with deadlines and profit. Also find the solution generated by this algorithm for following 7 jobs with their respective profit and deadlines mentioned as follows : **6M**

Job	1	2	3	4	5	6	7
Profit	3	5	20	18	1	6	30
Deadline	1	3	4	3	2	1	2

Ans. **P.3-43, 44, Q.51 & Q.52.**

Q.2. What is minimum cost spanning tree? Show the snapshots of Prim's algorithm to find minimum cost of spanning tree for the given graph: **8M**



Ans. **P.3-45, 53, Q.55 & Q.68.**

Q.3. What are the Optimal Huffman codes for following set of frequencies and discuss its complexity :

a : 50, b : 25, c : 15, d : 40, e : 75. **6M**

Ans. **P.3-58, Q.78.**

Q.4. Illustrate the stepwise operation of Heap sort on the input array,

A = <5, 13, 2, 25, 7, 17, 20, 8, 4>

Also find the recurrence relation for the algorithm and find its complexity. **9M**

Ans. **P.3-26, Q.24.**

Q.5. Write an algorithm to search an element using binary search method. Find the location of 45 in the given array using binary search method. A = < 9, 12, 15, 24, 30, 36, 45, 70 > **7M**

Ans. **P.3-12, 14, Q.3 & Q.6.**

WINTER - 12 (CS)

Q.1. Design Heap tree for the following sequence : **9M**

31 28 7 65 23 79 45 95

Write algorithm for INSERT, DELMAX and ADJUST operations.

Ans. **P.3-26, 30, Q.23 & Q.25.**

Q.2. Explain the principle of DAC method. For the following sequence of integers generate MIN-MAX TREE write an algorithm and comment on its complexity. **8M**

14, – 5, 30, 120, 55, 19, 10, – 4, 125, 120, 11

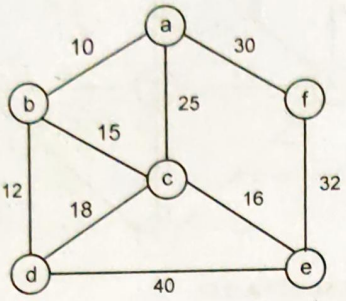
Ans. **P.3-10, 37, Q.1 & Q.34.**

Q.3. Find out average no. of comparisons required for successful and unsuccessful binary search on following array. **5M**

– 12 22 34 45 56 78 91 103 114 125 156

Ans. **P.3-14, Q.7.**

Q.4. Implement Prim's algorithm. Design cost matrix, near array and output matrix. Write Prim's algorithm. **8M**



Ans. P.3-50, 55, Q.65 & Q.71.

SUMMER - 13 (CT)

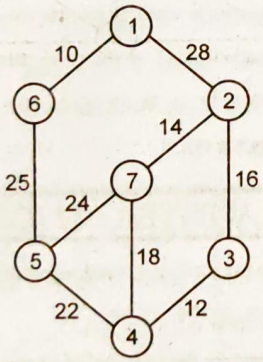
Q.1. Give stepwise operation of heap sort on following input array :

< 4, 8, 20, 17, 7, 25, 2, 13, 5> **5M**

Ans. P.3-25, Q.21.

Q.2. What is minimum-cost spanning tree? Write an algorithm for Kruskal's method for finding the minimum-cost spanning tree. Also discuss its complexity.

Solve the following example using Kruskal's method. **13M**



Ans. P.3-45, 46, 47, Q.55, Q.56 & Q.59.

Q.3. Write an algorithm to search an element using binary search method. Give recurrence relation and solve the same. **7M**

Ans. P.3-12, Q.3.

SUMMER - 13 (CS)

Q.1. Perform analysis of binary search on the following set of data and find out average number of comparisons required for successful and unsuccessful search operation. Explain how binary search will perform if the actual data is not available at the time of analysis :

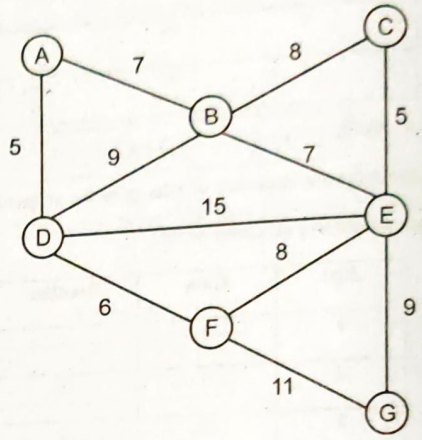
– 12, – 19, 87, 43, – 11, 65, 171, 123, 89, 105, – 9, 98. **6M**

Ans. P.3-15, Q.8.

Q.2. Write an algorithm for job schedule deadline approach. **8M**

Ans. P.3-44, Q.52.

Q.3. Write the PRIMS algorithm and implement PRIMS algorithm on the following graph. Find out the contents of near array at each step of processing. Comment on the complexity of the algorithm. **7M**



Ans. P.3-50, 55, Q. 65 & Q.72.

WINTER - 13 (CT)

Q.1. Show the snapshot of quick sort (1, 2, 4, 7, 6, 10, 9). Also find recurrence and time complexity in best and worst case. **7M**

Ans. P.3-16, 19, Q.10 & Q.14.

Q.2. Use strassen's algorithm to compute the matrix product.

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\begin{pmatrix} 6 & 7 \\ 3 & 8 \end{pmatrix}.$$ **5M**

Also find its recurrence relation and time complexity.

Ans. P.3-33, 34, Q.29 & Q. 30.

Q.3. Write binary search algorithm and calculate its time complexity. **5M**

Ans. P.3-12, Q.3.

Q.4. Prove that n-element heap has height h = [log n]. **5M**

Ans. P.3-33, Q. 27.

Q.5. Write Huffman code algorithm. Find optimal code for following set of frequencies.

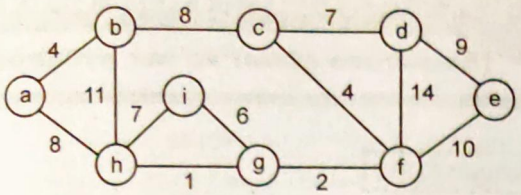
a : 20, b : 15, c : 5, d : 25, e : 35. **5M**

Ans. P.3-58, Q.77.

Q.6. What are the applications of greedy algorithm.? **3M**

Ans. P.3-38, Q.39.

Q.7. Obtain MCST and cost using Prim's algorithm. **7M**



Ans. **P.3-53, Q.69.**

WINTER - 13 (CS)

Q.1. Write insert, delete and adjust algorithm used in heap sort. Also explain complexity. **6M**

Ans. **P.3-26, Q.23 & Q.24.**

Q.2. Explain DAC. How does quick sort and merge sort fit into the strategy? **6M**

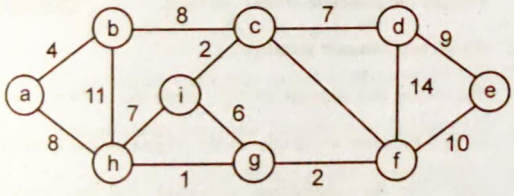
Ans. **P.3-10, 16, 19, Q.1, Q.10 & Q.15.**

Q.3. Write DAC based min-max algorithm and find the value from given array. **7M**

128, 82, 25, – 20, – 62, 43, 172, 95, 57, – 45

Ans. **P.3-36, Q.33.**

Q.4. Write Prim's algorithm. Implement the algorithm on the following graph. Explain its complexity. Explain the steps in execution. **7M**



Ans. **P.3-50, 53, Q.65 & Q.69.**

Q.5. Explain greedy based algorithm for job sequencing with deadline. **6M**

i	1	2	3	4	5	6
gi	20	15	10	7	5	3
di	3	1	1	3	1	3

Ans. **P.3-44, Q.52.**

Q.6. Write the algorithm for job scheduling with deadline. **6M**

Ans. **P.3-44, Q.52.**

SUMMER - 14 (CT)

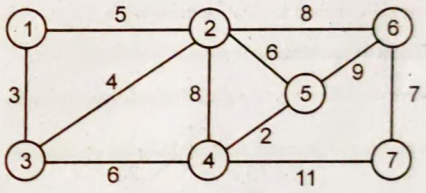
Q.1. For the following set of objects implement partial knapsack problem, with maximum capacity of 18:

Objects	Profit	Weight
1	9	2
2	15	3
3	12	5
4	4	4
5	6	3
6	16	6
7	8	3

Explain all three methods and comment on their profit value. **8M**

Ans. **P.3-42, Q.49.**

Q.2. Draw spanning tree for following graph using Kruskal's algorithm : **5M**



Ans. **P.3-48, Q.60.**

Q.3. Compare complexity of Quick Sort algorithm in best case, worst case and average case with example. **4M**

Ans. **P.3-18, Q.11, Q.12 & Q.13.**

Q.4. For the following sequence of job, give the snapshot of execution which will achieve maximum profit : **6M**

Gain	15	10	20	7	5	3
Deadline	1	1	3	3	1	3

Ans. **P.3-45, Q.54.**

SUMMER - 14 (CS)

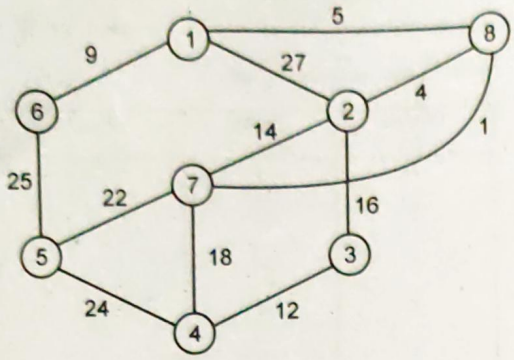
Q.1. Given stepwise operation of heap sort on following input array : **5M**

A = < 4, 8, 20, 17, 7, 25, 2, 13, 5>

Ans. **P.3-25, Q.21.**

Q.2. What is minimum cost spanning tree? Write an algorithm for Prim's method for finding the minimum-cost spanning tree. Also discuss its complexity.

Solve the following example using Prim's algorithm. **13M**



Ans. **P.3-45, 50, 54, Q.55, Q.65 & Q.70.**

Q.3. Write an algorithm for evaluation of partial knapsack. **4M**

Ans. **P.3-39, Q.43.**

Q.4. Write an algorithm for job scheduling with deadline. **5M**

Ans. **P.3-44, Q.52.**

SOLVED QUESTION BANK

[Sequence given as per syllabus]

INTRODUCTION

Divide and conquer is a technique which solves a problem by partitioning it into the smaller parts, finds the solution for the parts and then combine the solutions for the parts into a solution for the whole. In this unit, we will study binary search, sorting techniques and matrix operations.

Greedy algorithm are usually the most straight forward. They are shortsighted in their approach, taking decisions on the basis of information immediately at hand without worrying about the effect these decisions may have in the future. It studies various approaches as minimum cost spanning tree, knapsack problem, Huffman code etc.

DIVIDE AND CONQUER BASIC STRATEGY

Q.1. **Explain the divide and conquer strategy.**

CT : S-06,11, W-06(2M), CS: W-13(2M)

OR **What is divide and conquer strategy?** ***CT: W-11(2M)***

OR **Explain the principle of DAC method.** ***CS : W-12(2M)***

Ans. **Divide and conquer strategy :**

- The divide and conquer strategy break the problems into several subproblems that are similar to the original problem but smaller in size, solve the subproblems recursively and then combine these solutions to create a solution to the original problem.
- The divide-and-conquer paradigm involves three steps at each level of the recursion :

(1) **Divide :** In this step whole problem is divided into number of several sub-problems.

(2) **Conquer :** The sub-problems are conquered by solving them recursively. If the sub-problem sizes are small enough, however, just solve the sub-problems in a straight forward manner.

(3) **Combine :** Finally, the solutions obtained by the sub-problems are combined to create solution to the original problem.

Q.2. Explain how an array of n elements can be sorted using divide and conquer technique. Give algorithm and obtain its recurrence relation for best and worst case. CT: W-09(7M)

Ans.

- In cases where main problem and subproblems are of same type, then divide-and-conquer principle is naturally expressed by a recursive algorithm.
- Three examples of the divide and conquer algorithms are :
  (i) Quick sort algorithm.
  (ii) Binary search algorithm.
  (iii) Merge sort algorithm.
- Let A be the list of 9 items such as
  A = [(10), 3, 25, 13, 1, 28, 11, 14, (24)]
- Beginning with the last number 24, scan the list from right to left, comparing each number with 10 and stopping at the first number less than 10.
- The number is 1. Interchange 10 and 1 to obtain the list
  (1), 3, 25, 13, (10), 11, 14, 24
- Notice that all numbers to the right of 10 are greater than 10. Beginning with 1, next scan the list in the opposite direction from left to right, comparing each number with 10 and stopping at the first number greater than 10. The number is 25. Interchange 10 and 25 to obtain the list.
  1, 3, (10), 13, (25), 28, 11, 14, 24
- Notice that all numbers to the left of 10 are smaller than 10.
- Beginning this time with 25, now scan the list in the right to left direction until meeting the first number less than 10.
- We do not get such a number. This means all numbers have been scanned and compared with 10.
- Furthermore, all numbers less than 10 now form the sublist of numbers to the left of 10 and all numbers greater than 10 now form the sublist of numbers to the right of 10, as shown below

$$\underbrace{1, 3, 10}_{\text{first sublist}}, \underbrace{13, 25, 18, 11, 14, 24}_{\text{second sublist}}$$

- Thus, 10 is placed in its final position and the task of sorting the original list A has now been reduced to the tasks of sorting each of the above sublists.
- We can sort these sublists again as proceeding above to obtain a complete sorted list, i.e.
  1, 3, 10, 11, 13, 14, 18, 24, 25

Algorithm : Divide-and-conquer algorithm

```
Algorithm DAndC(P)
{
if small(P) then return S(P);
else
{
        divide P into smaller instances
        P1, P2, ....., Pk, k ≥ 1 ;
        Apply DAndC to each of these subproblems;
        return Combine (DAndC( P1 ),
        DAndC( P2 ) ..., DAndC (Pk ) );
        }
}
```

- DAndC (algorithm) is initially invoked as DAndC(P), where P is the problem to be solved.
- Small(P) is a Boolean-valued function that determines whether the input size is small enough the answer can be computed without splitting.
- If its value is true means problem is small enough and can be easily solved.
- If its value is false means problem is not small enough and it should be partitioned into smaller problems.
- These subproblems $P_1, P_2, \ldots, P_k$ are solved by recursive applications of DAndC.
- At the end, combine is a function that determines the solution to P using the solutions to the k subproblems.
- If the size of problem P is l, and the sizes of k subproblems are $l_1, l_2, \ldots, l_k$, respectively.

  Then the computing time of DAndC is described by the recurrence relation :

$$T(l) = \begin{cases} g(l) & l \text{ small} \\ T(l_1) + T(l_2) + \ldots + T(l_k) + f(l) & \text{otherwise} \end{cases}$$

Here, g(I) is the time to compute answer directly.

F(I) is the time for dividing P and combining the solutions to subproblems.

T(I) is the time for DAndC on any input of size I.

- The complexity of many divide-and-conquer algorithms is given by recurrences of the form

$$T(n) = \begin{cases} T(1) & n = 1 \\ aT\left(\frac{n}{b}\right) + f(n) & n > 1 \end{cases}$$

Here, a and b are constants.

- Here, we consider that value of T(1) is known and n is defined as power of b (i.e., $n = b^k$).

This complexity can be found by solving above recurrence relation. One of the method is substitution.

Example: Consider the case in which a = 2 and b = 2. Let T(1) = 2 and F(n) = n. Then we have

$$T(n) = 2T(n/2) + n$$
$$= 2[2T(n/4) + n/2] + n$$
$$= 4T(n/4) + 2n$$
$$= 4[2T(n/8) + n/4] + 2n$$
$$= 8T(n/8) + 3n$$

- Hence generally,

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + \text{ in any } \log_2 n \geq i \geq 1.$$

- In particular, then

$$T(n) = 2^{\log_2 n} T\left(\frac{n}{2^{\log_2 n}}\right) + n \log_2 n$$

corresponding to the choice of $i = \log_2 n$.

- Hence, $T(n) = nT(1) + n \log_2 n = \log_2 n + 2n$.

BINARY SEARCH

Q.3. Write binary search algorithm and calculate its time complexity. **CT : W-13(5M)**

OR Write an algorithm to search an element using binary search method. Give the recurrence relation and solve the same. **CT: S-09,13(7M),W-12(3M)**

OR Using divide and conquer strategy, write an algorithm to search a given element in a sorted array. Also find its time complexity. **CT: W-11(4M)**

OR Write program segment which has complexity O (log n). **CT: W-08(2M)**

OR Explain complexity of binary search. **CS: W-11(2M)**

Ans. Binary search algorithm :

Algorithm BINARY_SEARCH(KEYS, N, X)

Input : Given a vector KEYS of size N can input parameter whose elements are in ascending order, this function searches the vector for a given element whose value is given by the input parameter X. The variables LOW, HIGH and MIDDLE are local integer variables and denote the lower and upper limits of search interval and the midpoint respectively. All variables are of type integer.

Output : If the search is successful, the position of the found element is returned; otherwise the value 0 is returned.

Step 1: [Initialize]
LOW ← 1
HIGH ← N

Step 2 : [Perform search]
Repeat through step 4 while LOW ≤ HIGH

Step 3 : [Obtain index of midpoint of interval]
MIDDLE ← (LOW + HIGH) div2

Step 4 : [Compare]
If X > KEYS [MIDDLE]
then HIGH ← MIDDLE –1
else if X > KEYS [MIDDLE]
then LOW ← MIDDLE + 1
else return (MIDDLE)
end if

Step 5 : [Unsuccessful search]
Return (0)

Time complexity of binary search :

- To analyse the algorithm binary search, first we have to compute the number of comparisons expected for the successful binary search.
- So consider 'P' such that

$$2^P \geq CN + 1$$

- The maximum number of comparison required for successful search.

$$C_{ms} = i$$

Or $C_{ms} = \log_2(N+1)$

- For unsuccessful search the maximum number of comparisons : $C_{ms} = C_{mv}$
- For computing the average no of comparisons 'a' indicates successful search.

$$C_s = \frac{C}{N}$$

$$C = i.\ \alpha^i - (2^0 + 2^1 + 2^2 + ... + 2(i-1))$$

$$= 1 + 2^i(i-1)$$

- Consider that the probability for searching a request data is 1/n then

$$C_s = (1 + 2^i(1 - i/n))$$

$$C_s = [1 + (N+1)\log_2(N+1) - 1]/n$$

$$\boxed{C_s = O(\log_2 N)}$$

Recurrence equation :

$$T(n) = T\left(\frac{n}{2}\right) + (n)$$

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

Let us assume solution is 0 (log n)

∴ We need to prove

$$T(n) \leq c\log(n)$$

$$T(n) \leq c\log\left(\frac{n}{2}\right) + 1$$

$$= c(\log n - \log_2) + 1$$

$$= c(\log n - 1) + 1$$

$$= c\log n - c + n$$

$T(n) \leq c \log n$ for $c \geq n$

$T(n) = O(\log n)$ hence proved.

Q.4. Explain analysis of binary search and discuss its complexity for an array of size 10. Write recursive algorithm for binary search.

CS: S-10(8M)

Ans. Analysis of binary search :

- This search will find the number of comparison required for successful search or unsuccessful search.
- It is the method of finding out an element from a sorted array by dividing array into two equal parts and division is continued till the element is found or array is reduced to size 1.

The analysis of binary search are as following :

(1) Worst-case (successful or fail) :

$$C_w(n) = 1 + C_w([n/n]),$$

$$C_w(1) = 1$$

Solution : $C_w(n) = [(\log_2 n] + 1[\log_2(n+1)]$

This is very fast.

$$C_w(10^6) = 20$$

(2) Best case :

successful $C_b(n) = 1$

fail $C_b(n) = [\log_2 n] + 1$

(3) Average case :

successful $C_{avg}(n) = \log_2 n - 1$

fail $C_{avg}(n) = \log_2(n+1)$

- The recursive algorithm for binary search are as follows :

```
Binary_Search_Recur (A [0 ..... n – 1], l, r, k)
if l > r
return – 1
else
        m = [(l + r)/2]
if k = A(m)
return m
else
        if k < A[m]
            return Binary_Search_Recur (A [0 ..... n – 1], l, r, k)
        else
        return Binary_Search_Recur (A [0 ..... n – 1], m + 1, r, k)
```

Q.5. Implement binary search on the following array and find average number of comparison required for successful and unsuccessful search?

– 12, – 4 , 9, 32, 50, 79, 109, 135, 203, 230 *CS: W-11(3M)*

Ans. Given array = a [10]

search – 4

set,

low = 1

high = 10(n)

$$\therefore \text{ mid} = \left(\frac{\text{low} + \text{high}}{2}\right) = \frac{1+10}{2} = 5$$

Now, low = 1 and high = 4

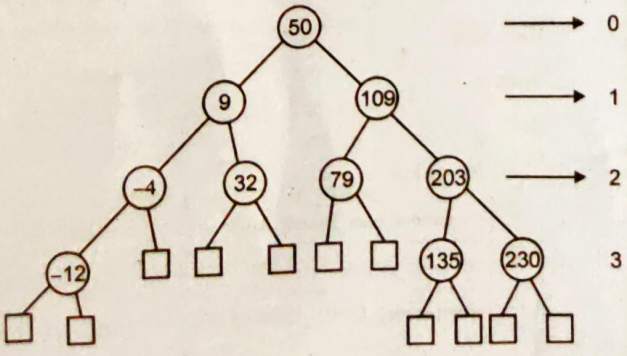
$$\text{mid} = \frac{1+4}{2} = 2$$

Now, a [mid] = – 4

– 4 = – 4

∴ – 4 is searched.

Array	Elements	Comparison
1	– 12	4
2	– 4	3
3	9	2
4	32	3
5	50	1
6	79	3
7	109	2
8	135	4
9	203	3
10	230	4



∴ Total number of comparison = 29

Average $= \frac{29}{10} = 2.9$ → for successful search.

Average $= \frac{29}{11} = 2.63$ → for unsuccessful search.

Q.6. Find the location of 45 in the given array using binary search method.

A = < 9, 12, 15, 24, 30, 36, 45, 70 > *CT: W-12(4M)*

Ans. Given array :

9	12	15	24	30	36	45	70
a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]

Lower bound = LB = 1

Upper bound = UB = 8

Middle element = Mid

Mid = (LB + UB)/2

= (1 + 8)/2

= 9/2

= 4

∴ Mid = a[4] = 24

Search element = 45 = K

K > a[4]

∴ LB = Mid + 1

LB = 5, UB = 8

Mid = (5 + 8)/2 = B/2 = 6

Mid = a[6] = K

∴ Low = Mid + 1 = 6 + 1 = 7

Mid = (7 + 8)/2 = 15/2 = 7

Now, Mid = K, hence search is unsuccessful.

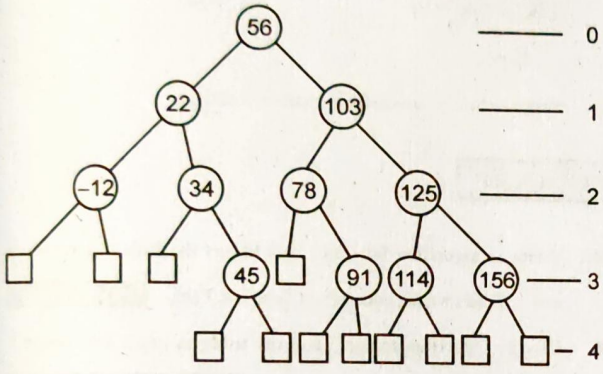
Q.7. Find out the average number of comparisons required for successful and unsuccessful search on following array.

– 12, 22, 34, 45, 56, 78, 91, 103, 114, 125, 156 *CS: W-12(5M)*

Ans.

Array	Elements	Comparison
1	– 12	3
2	22	1

3	34	3
4	45	4
5	56	1
6	78	3
7	91	4
8	103	1
9	114	3
10	125	3
11	156	4



Total no of comparison

$$= \frac{30}{11} = 2.7 \text{ for successful search}$$

for unsuccessful search

$$= \frac{EPL}{n+1} = \frac{44}{12} = 3.6$$

EPL = (3 * 4) + (4 * 8) = 12 + 32 = 44

Q.8. Perform analysis of binary search on the following set of data and find out average number of comparisons required for successful and unsuccessful search operation. Explain how binary search will perform if the actual data is not available at the time of analysis :

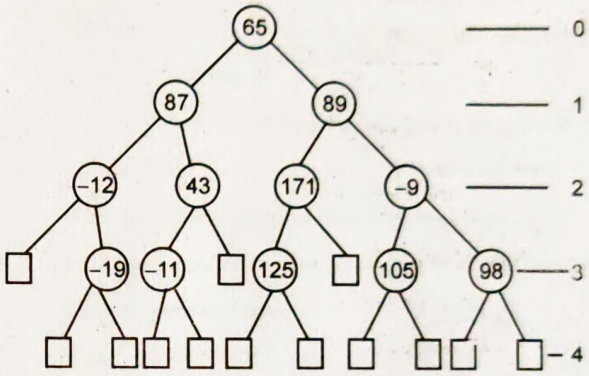
– 12, – 19, 87, 43, – 11, 65, 171, 123, 89, 105, – 9, 98.

CS: S-13(6M)

Ans.

Array	Elements	Comparison
1	– 12	3
2	– 19	4
3	87	2

4	43	3
5	– 11	4
6	65	1
7	171	3
8	123	4
9	89	2
10	105	4
11	– 9	3
12	98	4



Average number of comparison for successful search

= (0 * 1) + (1 * 2) + (2 * 4) + (3 * 5)

= 0 + 2 + 8 + 15 = 25

$$\frac{25}{12} = 2.08$$

- If actual data is not available at the time of analysis then from the array size we can calculate the average number of comparison for successful search as ($\log_2 n$).

  Where, n is the size of array.

Q.9. How does binary search fit into the divide and conquer strategy?

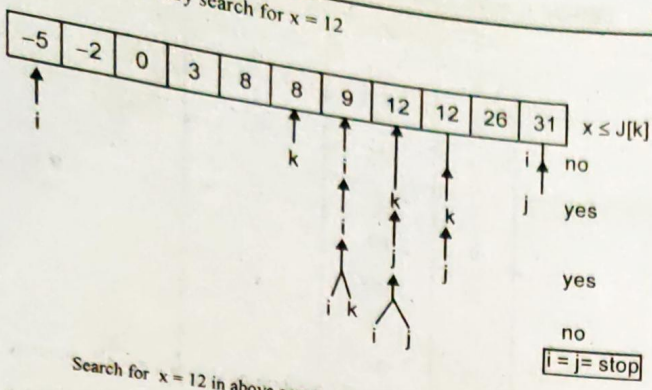
CT: S-06(2M)

Ans. Binary search : It is a method of finding out an element from a sorted array by dividing array into two equal parts and division is continued till the element is found or the array is reduced to size 1.

Example : Suppose A is given array shown below :

	1	2	3	4	5	6	7	8	9	10	11
A	– 5	– 2	0	3	8	8	9	12	12	26	31

Apply binary search for x = 12



Search for x = 12 in above array

Step 1 : Find the value of i and j

$i = 1$

$j = 11$

Now, find value of k as

$$K = \text{int}\left(\frac{i+j}{2}\right)$$

$$= \text{int}\left(\frac{1+11}{2}\right)$$

$$= \text{int}\left(\frac{12}{2}\right) - 6$$

$K = 6$

Step 2 : $T[K] = T[6] = 8$

$8 < 12$ then

$i = K + 1$

$i = 6 + 1 = 7$

Step 3 : Again

$$K = \text{int}\left(\frac{i+j}{2}\right)$$

$$= \text{int}\left(\frac{7+11}{2}\right)$$

$$= \text{int}\left(\frac{18}{2}\right)$$

$K = 9$

7	8	9	10	11
9	12	12	26	31

$T[K] = T[9] = 12$

i.e. $12 = 12$

Hence search is successful at location 9 again,

$$K = \text{int}\left(\frac{i+j}{2}\right)$$

$$= \text{int}\left(\frac{7+9}{2}\right)$$

$$= \text{int}\left(\frac{16}{2}\right)$$

$K = 8$

7	8	9
9	12	12

$T[K] = T[8] = 12$

i.e. $12 = 12$

Hence search is successful at location 8 also.

QUICK SORT

Q.10. Write an algorithm for quick sort to sort the data in descending order. Find its time complexity function T(n). CT: W-06(5M)

OR Give an algorithm to sort an array using m quick sort method. Obtain its recurrence relation. CT: W-07(3M), W-13(2M)

OR Explain the quick sort algorithm with suitable example. Find out the time complexity of algorithm. CT: W-08(6M), S-10(7M)

OR State the recurrence relation for the best case and worst case of quick sort algorithm. CT: S-09(5M)

OR How does quick sort fit into the divide and conquer strategy? CT: S-11(3M), CS: W-13(2M)

Ans. Quick sort :

- The basic version of quick sort algorithm was invented by C.A.R. Hoare in 1960.
- It is used on the principle of divide and conquer.
- In quick sort, we divide the list of elements into two list by choosing a partitioning element called pivot element.
- One list contains all elements less than or equal to the partitioning element and the other list contains all elements greater than the pivot element.
- These two lists are recursively partitioned in the same manner till the resulting list becomes trivially small to sort by comparison .We then go on combining the sorted lists to produce the sorted list of all input elements.

- A general algorithm based on recursive approach is as follows :

(1) Partition the current table into two subtables.

(2) Invoke quick sort to sort the left subtables.

(3) Invoke quick sort to sort the right subtables.

- A detailed algorithm based on recursive formulation can now be given as :

```
Algorithm partition (a, i, j) || i = 1, j = n,
{
    Key = a [i], p = i, q = j ;
        do
        {
        do
            p = p + 1
            while (a [p] > key);
              do
              q = q – 1
              while (a [q] < key) ;
        if (p < q)
        interchange a [p] and a [q]
    }
    while (p < q) ;
    interchange key and a [q]
}
Algorithm Quicksort (a, m, n)
{
    if (m < n)
    j = partition (a, m, n) → n
    Quicksort (a, m, j – 1)→ T(n | 2)
    Quicksort (a, j + 1, n) → T(n | 2)
}
Algorithm Interchange (a, p, q)
{
    m = a [p];
    a [p] = a [q];
    a [q] = m;
}
```

- Let us consider the following unsorted array and try to sort it using quick sort method.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
24	56	47	35	10	90	82	31

- We choose the first number in the array 24 as the pivot and is list partitioned into sublists.

Pivot = 24

(10) and (56 47 35 90 82 31)

- At this point 24 is in its proper position at x[1] where each element in the first subarray is less than 24 and each element in the second subarray is greater than 24.
- Now our problem is broken into two subproblems that is to sort the two subarrays.
- Since a first subarray contains a single element it is already sorted.
- To sort the second subarray we choose its first element 56 as the pivot and yields two subarrays.

(47 35 31) and (90 82)

- The entire array may be represented as

10 24 (47 35 31) 56 (90 82)

Where, the two subarrays to be sorted are closed in parentheses. The further steps may be shown as:

10 24 (35 31) 47 56 (90 82)

10 24 (31) 35 47 56 (90 82)

10 24 31 35 47 56 (90 82)

10 24 31 35 47 56 (82) 90

10 24 31 35 47 56 82 90

And the final array is sorted.

Time complexity:

Recursive equation = $t(n) = 2T(n|2) + n$

$T(n) \le 2(C\lfloor n|2 \rfloor) \lg(\lfloor n|2 \rfloor) + n$

$\le C_n \lg(n|2) + n$

$= C_n \lg n - C_n \lg 2 + n$

$= C_n \lg n - C_n + n$

$\le C_n \lg n$.

∴ Time complexity of quick sort = n log n.

(1) Worst -case complexity :

$$T(n) = T(n-1) + O(n)$$

(2) Best-case complexity :

$$T(n) = 2T(n/2) + O(n)$$

(3) Average-case complexity :

$$T(n) = O(n \log n)$$

Q.11. Explain worst case complexity of quick sort with suitable example.

CT: S-14, W-06,07(1M).

CT: S-09(2M), CS: W-11(4M)

Ans. In this case, on every function call, the given array is partitioned into two subarrays. One of them is an array. One of them is an empty array, now the timing analysis takes the form,

$$T(n) = C_n + T(0) + T(n-1)$$

$$= C_n + T(n-1)$$

$$= C_n + C(n-1) + T(n-2)$$

$$= C_n + C(n-1) + C(n-2) + T(n-3)$$

$$\vdots \quad \vdots$$

$$= C_n + C(n-1) + C(n-2) + \ldots + C(1) + T(0)$$

$$= C[n + (n-1) + (n-2) + \ldots + 2 + 1]$$

$$= C\left[\frac{n(n+1)}{2}\right]$$

$$= \frac{C_n^2}{2} + \frac{C_n}{2}$$

$$= O(n^2).$$

Q.12. Show that quick sort's best-case running time is Ω (n/g n).

CT: S-06, 08(2M)

OR Give the best case of the quick sort algorithm.

CT: S-14, W-06,07(1M)

Ans. Best case :

$$T(n) = \min_{1 \le q \le n-1} (T(q) + T(n-q)) + \theta(n)$$

Guess that $T(n) \ge Cn \lg n$ for some constant C

$$T(n) \ge \min_{1 \le q \le n-1} (C_q \lg q + C(n-q) \lg (n-q)) + \theta(n)$$

$$= C \min_{1 \le q \le n-1} (q \lg q + (n-q) \lg (n-q)) + \theta(n)$$

Now, to find the minimum value of the function

$$f(q) = q \lg q + q + (n-q) \lg (n-q)$$

for $q = 1$

$$f(q) = (n-1) \lg (n-1)$$

for $q = n-1$

$$f(q) = (n-1) \lg (n-1)$$

for $q = \frac{n}{2}$

$$f(q) = n \lg \left(\frac{n}{2}\right) < n \lg n$$

Thus, the minimum appears somewhere between

$q = 1$ and $q = n-1$

Now, $f'(q) = \frac{1}{q} q + \lg q + (n-q) \frac{1}{(n-q)} (-1) + (-1) \lg (n-q)$

$$= 1 + \lg (q-1) - \lg (n-q)$$

$$= \lg q - \lg (n-q)$$

for minimum

$$f'(q) = 0$$

i.e. $\lg q - \lg (n-q) = 0$

which gives $q = \frac{n}{2}$

Put $q = \frac{n}{2}$ into the original equation

$$T(n) \ge C\left(n \lg \left(\frac{n}{2}\right)\right) + \theta(n)$$

$$= Cn \lg n - Cn \lg 2q + \theta(n)$$

$$\ge Cn \lg n$$

If we choose the constant C, so that the constant is $\theta(n) > C \lg 2$.

Thus, quick sort's best case execution time is Ω (n lg n).

Q.13. Find average case of the quick sort algorithm.

CT: S-14(2M), W-06, 07(1M)

OR Write a program segment which has complexity O(n log n).

CT: W-08(2M)

OR Prove that Quick-Sort takes a time in O(n log (n)) to sort n elements on the average.

Ans.

- In analyzing Quick Sort, we count only the number of comparison C(n).

- Assume that the n elements to be sorted are distinct and the input distribution is such that the position element v = a[m] in the call to partition (a, m, p) has an equal probability of being the smallest element, $1 \le i \le p - m$, in a[m : p - 1].
- Hence the two subarrays remaining to be sorted are a[m : i] and a[j + 1 : p - 1] with probability $1/(p - m)$, $m \le j < p$.
- For this we obtain the recurrence.

Average time,

$$C_A(n) = n + 1 + \frac{1}{n}\sum_{1 \le k \le n} [C_A(k-1) + C_A(n-k)] \quad \ldots. (i)$$

- The number of element comparison required by partition on its first call is n + 1. Note that $C_A(0) = C_A(1) = 0$.
- Multiplying both sides of equation (ii) by n, we obtained.

$$nC_A(n) = n(n+1) + 2[C_A(0) + C_A(1) + \ldots. + C_A(n-1)] \quad \ldots\ldots (ii)$$

- Replacing n by n – 1 in equation (ii) gives

$$(n-1)C_A(n-1) = n(n-1) + 2[C_A(0) + \ldots. + C_A(n-2)]$$

- Subtracting this from equation (ii), we get

$$nC_A(n) - (n-1)C_A(n-1) = 2n + 2C_A(n-1)$$

OR $C_A(n)/(n+1) = C_A(n-1)/n + 2/(n+1)$

- Repeatedly using this equation to substitute for $C_A(n-1).C_A(n-2)$, we get

$$\frac{C_A(n)}{n+1} = \frac{C_A(n-2)}{n-1} + \frac{2}{n} + \frac{2}{n+1}$$

$$= \frac{C_A(n-3)}{n-2} + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1} \quad \ldots. (iii)$$

$$= \frac{C_A(1)}{2} + 2\sum_{3 \le k \le n+1} \frac{1}{k} = 2\sum_{3 \le k \le n+1} \frac{1}{k}$$

Since, $\sum_{3 \le k \le n+1} \frac{1}{k} \le \int_{2}^{n+1} \frac{1}{x} dx = \log_e(n+1) - \log_e 2$

Then, equation (iii) yields

$$C_A(n) \le 2(n+1)[\log_e(n+2) - \log_e 2]$$

$$= O(n \log n)$$

Thus, the average time complexity is O (n log n).

Q.14. Show the snapshot of quick sort when supplied with input 1, 2, 4, 7, 6, 10, 9. *CT : W-13(3M)*

Ans. Snapshot :

(1) 2 4 7 6 10 (9)

(1) Compare with all element from right to left direction.

∴ The array will be :

1 2 4 7 6 10 9

(2) Compare 6 with elements from left to right 7 > 6. Interchange 6 and 7

∴ The array will be :

$\underbrace{1\ 2\ 4}_{\text{First sublist}}$ 6 $\underbrace{7\ 10\ 9}_{\text{Second sublist}}$

(3) Now, sort the second sublist as :

1 2 4 6 7 10 9

∴ The array will be :

1 2 4 6 7 9 10

MERGE SORT

Q.15. Write an algorithm for Merge Sort. Give stepwise run of Merge Sort algorithm when supplied with the input. (30, 28, 17, 65, 35, 42, 20, 45, 52). *CT: W-08(7M)*

OR Write an algorithm to merge the two arrays, which are in sorted order. *CT : S-09(4M)*

OR Write an algorithm ternary search which searches an element in an array by dividing the array in three equal parts instead of two halves as in case of binary search. Give the recurrence relation. *CT: W-09(7M), CS : W-10(5M)*

OR Modify the binary search such the input is divided into three equal parts in the search procedure. *CT : W-06(3M)*

OR Justify how merge sort fits into divide and conquer algorithm design method. *CT : W-06(5M), CS : W-13(2M)*

OR Write down the merge sort algorithm. *CT: W-11(4M)*

Ans. Algorithm for merge sort :

- An auxiliary array of size n is required to hold the result of merging two subarray of x.
- The variable size contains the size of subarrays being merged.

- Since at any time the two files being merged are both subarrays of x, lower and upper bounds are required to indicate the subfiles of x being merged. l_1 and u_1 represent the lower and upper bounds of the first file, and l_2 and u_2 represent the lower and upper bounds of the second files, respectively.

Step 1 : Start

Step 2 : Read size of array, say 'n' and an array, say 'x'.

Step 3 : To merge files of size 1, set size = 1.

Step 4 : While (size < n) do

(1) Initialize lower bound of first file, put $l_1 = 0$

(2) While (l_1 + size < n) do (i.e. check if there are two files to merge).

(3) Identify lower bound (l_2) of second file and upper bounds (u_1 , u_2) of first and second file respectively.

(4) Sort two subfiles and store into auxiliary array (i.e. aux)

(5) Advance l_1 to the start of the next pair of files, $l_1 = u_2 + 1$

(6) Copy any remaining single file into auxiliary array. (If n is odd, then 1 subfile always remain)

(7) Copy auxiliary array (i.e. aux) into original array (i.e. x)

(8) Size = size *2

Step 5 : Display sorted array.

Step 6 : Stop.

- Consider the array with elements

  30, 28, 17, 65, 35, 42, 20, 45, 52
- The array of 9 elements. a [1:9] = (30, 28, 17, 65, 35, 42, 20, 45, 52)
- Algorithm merge sort splits a[ ] into two subarrrays a [1 : 5] and a [6 : 9].
- The elements in a [1 : 5] are split into two subarray of size three i.e. a[1 : 3] and a[4 : 5]. Then the items in a[1 : 3] are split into subarays of size two (a[1 : 2]) and one a[3 : 3].
- The two values in a [1 : 2] are split a final time into one element subarrays and now the merging begins.
- Pictorially the file can now be viewed as :

  (30 | 28 | 17 | 65, 35 | 42, 20, 45, 52)

  Elements a[1] and a [2] are mergd to yield

  (30, 28 | 17 | 65, 35| 42, 20, 45, 52)
- a[3] is merged with a[1 : 2] then

  (17, 28, 30 | 65, 35 | 42, 20, 45, 52) is produced.
- Next elements a[4] and a[5] are merged and sorted

  (17, 28, 30 | 35, 65 | 42, 20, 45, 52)

  and then a[1 : 3] and a [4 : 5]

  (17, 28, 30, 35, 65 | 42, 20, 45, 52)
- Now, the merge sort processes the second recursive call. Repeated recursive calls produces the following subarrays.

  (17, 28, 30, 35, 65 | 42, 20 | 45, 52)
- Next a[8] and a[9] are merged and then a[6 : 7] and a[8 : 9]

  (17, 28, 30, 35, 65| 20, 42, 45, 52)
- At this point there are two sorted subarrays and the final merge produces the fully sorted result :

  (17, 20, 28, 30, 35, 42, 45, 52, 65).

Q.16. Write an recursive algorithm to implement merge sort technique. Derive its recurrence relation and solve to obtain the time complexity. **CT: S-07(13M)**

OR Write recursive merge sort and also explain its recurrence equation. **CT : W-06(3M), CS : S-10(7M)**

OR Discuss time complexity of merge sort. **CT : S-09(2M)**

OR Find time complexity of merge sort algorithm. Can you improve the time complexity? What will be the pros and cons if the sort file is very large? **CT: W-11(4M)**

Ans. Algorithm:

```
Algorithm mergesort (a, low, high)
{
    if (low < high)
    {
        mid = ⌊ low + high | 2 ⌋ ;
        mergesort (a, low, mid);
        mergesort (a, mid+1, high);
        merge (a, low, high);
    }
}
```

Recurrence Relation for merge sort :

$t(n) = 2T(n/2) + n$

Let $n = 2^i$

$t(i) = 2t(2^i/2) + 2^i$

$ti = 2t(2^{i-1}) + 2^i$ [non homogeneous + Exponent]

$ti - 2t(2^{i-1}) = 2^i$

Roots are

$(x-2)$ and $(x-2) \Rightarrow (x-2)^2$

$r_1 = 2, \quad r_2 = 2 \Rightarrow$ Roots are real and same.

$ti = c_1 r_1^i + c_2 \cdot i \cdot r_2^i$,

$ti = c_1 2^i + c_2\ i \cdot 2^i$

Replace $n = 2^i$ or $i = \log_2 n$

$t(n) = c_1 . n + c_2 (n . \log_2 n)$

↓

Largest factor

$\therefore \quad n . \log_2 n \Rightarrow$ Time complexity.

```
Algorithm merge (low, mid, high)
{
h = low ; i = low; j = mid + 1;
while ((h ≤ mid) and (j ≤ high)) do
{
    if (a [h] ≤ a [j]) then
    {
        b [i] = a [h];
        h = h + 1;
    }
    else
    {
    b[i] = a [j];
    j = j + 1;
    }
    i = i + 1;
    if (h > mid) then
    for K = i to high do
    {
    b[i] = a [K];
    i = i + 1;
    }
    else
    for K = h to mid do
    {
    b [i] = a [K]; i = i + 1;
    }
```

- Each recursive call has O(n) runtime and a total of O(log n) recursions are required, thus the runtime of this algorithm is O(n*logn)
- A merge sort can also be modified for performance on lists that are nearly sorted to begin with.
- After sorting each half of the data, if the highest element in one list is less than the lowest element in the other half, then the merge step is unnecessary.
- Apart from being fairly efficient, a merge sort has the advantage that it can be used to solve other problems, such as determining how "unsorted" a given list is.

Advantages :

(1) Guaranteed to be O(n log n).

(2) Simple to understand.

Disadvantages :

(1) Not as fast as Quick sort on average.

(2) Requires as much memory as the original array.

Q.17. Draw merge and split tree for the following sequence. Write recursive merge sort and explain its recurrence equation. How merge sort is different from quick sort?

210	**495**	**325**	**125**	**280**	**861**	**652**
730	**423**	**148**	**351**	**176**	**481**	**530**

CS: S-10(6M)

Ans. Merge sort is different from quick sort as it requires $\theta(1)$ extra space and slow random access performance of linked list.

Merge and split tree :

a [1 . . 14] = [210, 495, 325, 125, 280, 861, 652, 730, 423, 148, 351, 176, 481, 530]

VBD

Split tree :

$$mid = \frac{1+14}{2} = \frac{15}{2} = 7$$

1 = [210, 495, 325, 125, 280, 861, 682] [730, 423, 148, 351, 176, 481, 530]

2 = [210, 495, 325, 125] [280, 861, 652] [730, 423, 148, 351] [176, 481, 530]

3 = [210, 495] [325, 125] [280, 861] [652] [730, 423] [148, 351] [176, 481] [530]

4 = [210] [495] [325] [125] [280] [861] [652] [730] [423] [148] [351] [176] [481] [530]

Merge Tree :

[125, 210, 325, 495] [280, 652, 651, 730] [148, 176, 351, 423] [481, 530]

[125, 210, 280, 325, 495, 652, 681, 730] [148, 176, 351, 423, 481, 530]

Sorted array :

[125, 148, 176, 210, 280, 325, 351, 423, 481, 495, 530, 652, 681, 730]

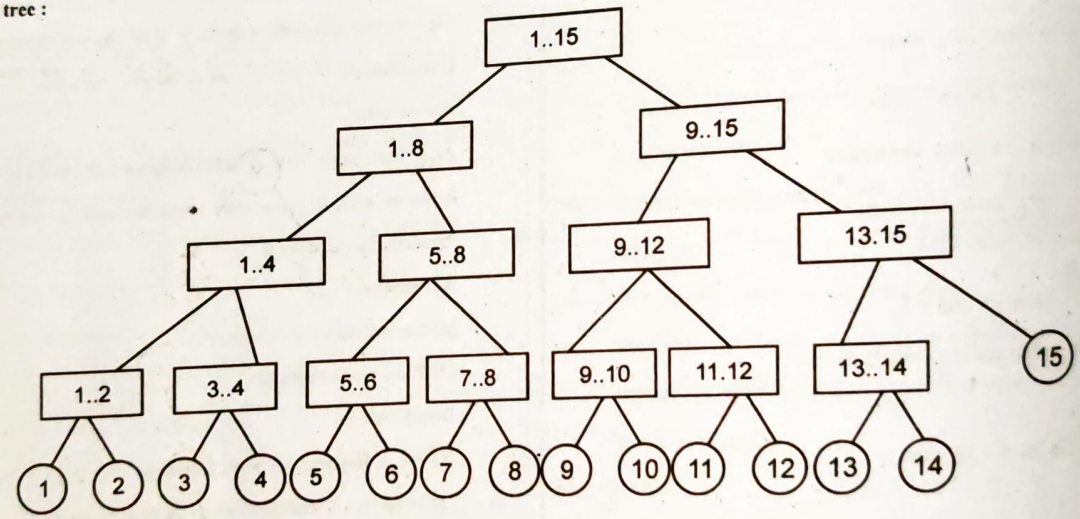
CS: W-116

Q.18. Draw merge and split tree using merge sort for array of size 15. Write recurrence equation of merge sort.

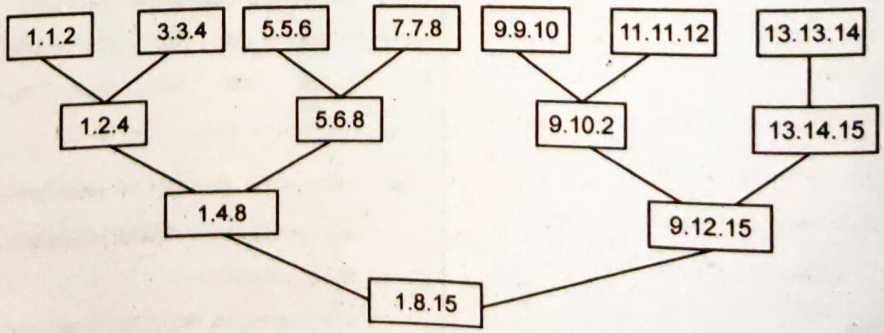
Ans. Recurrence equation of merge sort :

$$T(n) = \begin{cases} \theta(1) & \text{if } n = 1 \\ 2T(n/2) + \theta(n) & \text{if } n > 1 \end{cases}$$

Split tree :



Merge tree :



VBD

19. In the given array design split tree and merge tree. Assume 14 elements is given in array as follows : 7, 8, 14, 10, 2, 6, 18, 11, 21, 19, 12,14, 28, 23. CS: S-12(5M)

ns. Given : Size of array = 14

Split tree :

[7, 8, 14, 10, 2, 6, 18] [11, 21, 19, 12, 14, 28, 23]

[7, 8,14,10] [2, 6, 18] [11, 21, 19, 12] [14, 28, 23]

[7, 8] [14, 10] [2, 6], [18] [11, 21] [19, 12] [14, 28] [23]

[7] [8] [10] [14] [2] [6] [18] [11] [21] [19] [12] [14] [28] [23]

Merge tree :

[7, 8] [10,14] [2,6][11,18] [19, 21][12,14][23, 28]

[7,8, 10,14] [2, 6, 11, 18] [12, 14, 19, 21] [23, 28]

[2, 6, 7, 8, 10, 11, 14, 18] [12, 14, 19, 21, 23, 28]

Sorted array :

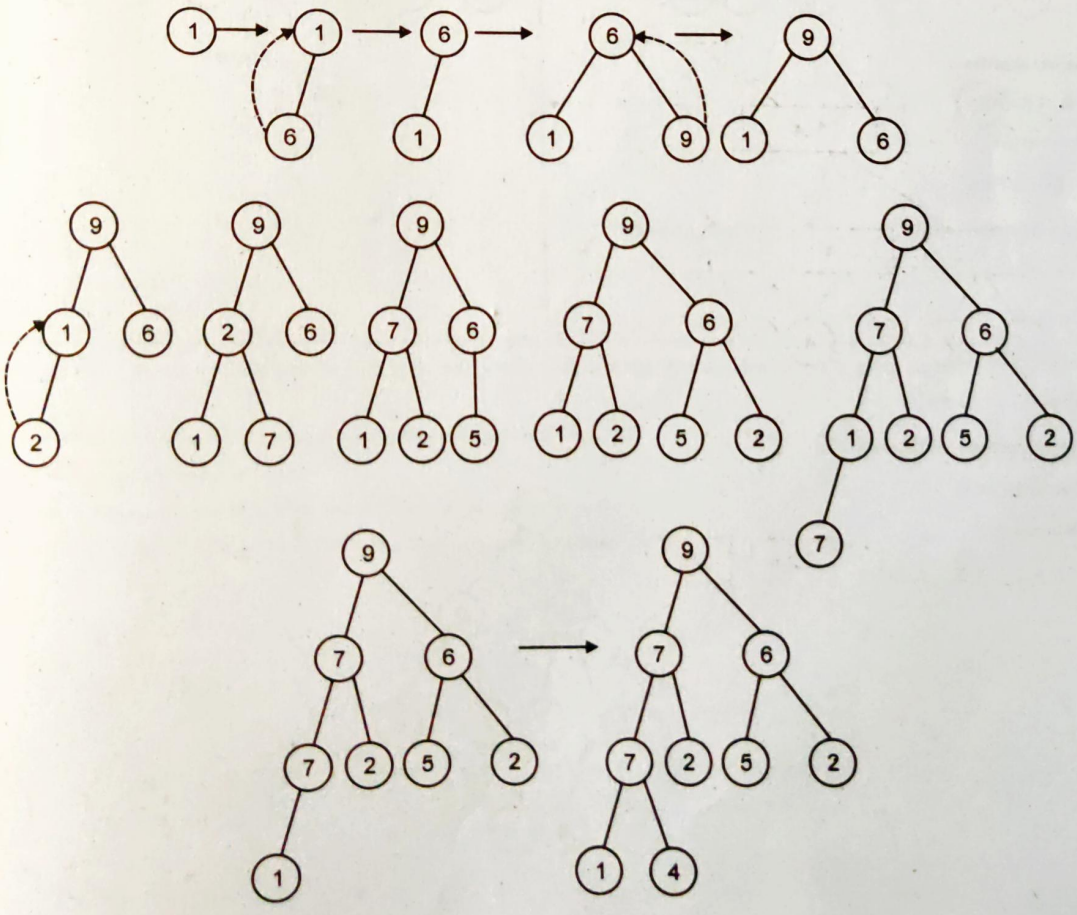
[2, 6, 7, 8, 10, 11, 12, 14, 18, 19, 21, 23, 28]

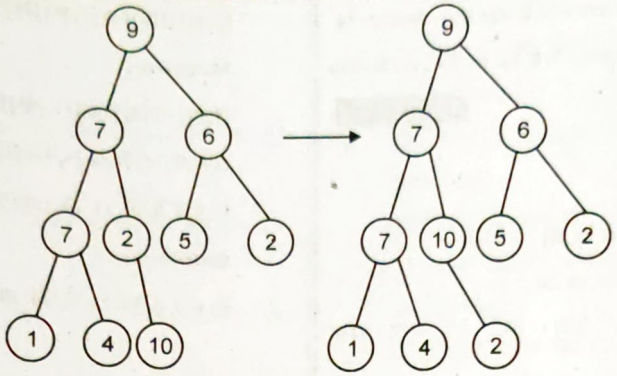
HEAP SORT

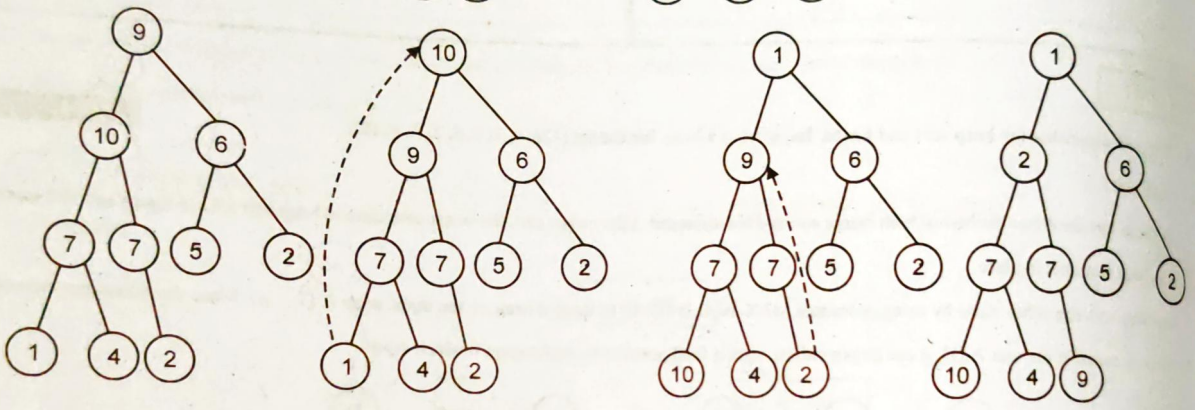
Q.20. Write an algorithm for heap sort and heaps. Implement a heap for the set (1, 6, 9, 2, 7, 5, 2, 7, 4, 10). CS : S-11, 12(6M)

Ans. Heap sort :

- The heap sort combines the best of both merge sort and insertion sort. Like merge sort, the worst case time of heap sort is O (n log n) and like insertion sort, heap sort sorts in place.
- The heap sort algorithm starts by using procedure MAX-BUILD-HEAP to build a heap on the input array A [1 . . n]. Since the maximum element of the array stored at the root A [1], it can be put into its correct final position by exchanging it which A[n].







Heap sort algorithm :

HEAP_SORT(A)

(1) Build - heap (A)

(2) For i ← n to 1 do

(3) Swap (A [1], A [i])

(4) n ← n – 1

(5) Heapify (A, 1)

Build MAX-HEAP (A)

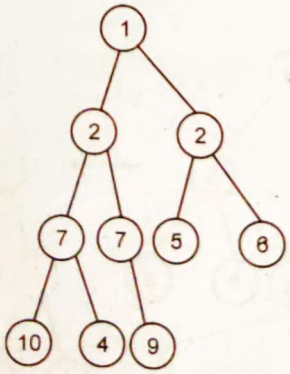
(1) heap - size (A) ← length [A]

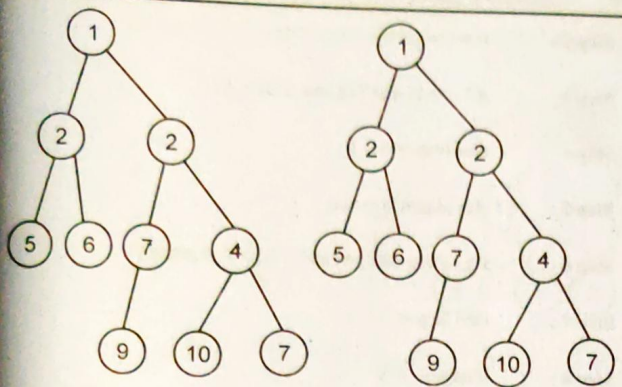
(2) for i ← floor (length [A]/2) down to 1 do

(3) MAX-HEAPIFY (A, i)

Given array :

A < 1, 6, 9, 2, 7, 5, 2, 7, 4, 10 >





Sorted array :

1, 2, 4, 5, 6, 7, 8, 10

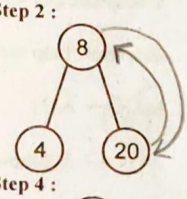
Q.21. Illustrate the stepwise operation of heap sort on the input array <4, 8,20, 17, 7, 25, 2,13, 5>

CT : S-08(7M), S-13(4M), CS : S-14(5M)

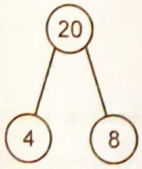
Ans.

Step 1:

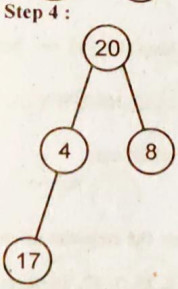
Step 2 :



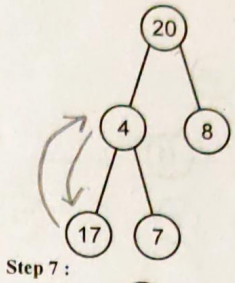
Step 3 :



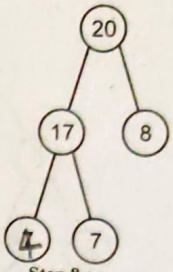
Step 4 :



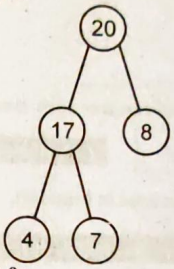
Step 5 :



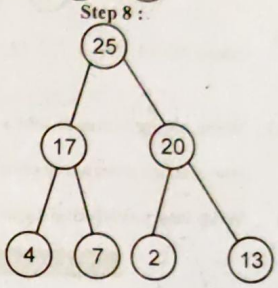
Step 6 :



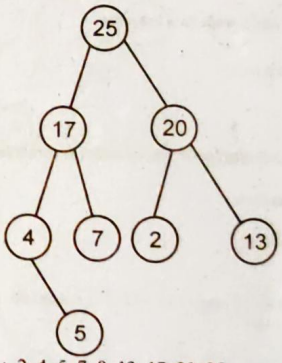
Step 7 :



Step 8 :



Step 9 :



Sorted array : 2, 4, 5, 7, 8, 13, 17, 20, 25

Q.22. Define heap tree. Construct a heap tree for each of the following numbers if added in the given order 1, 15, 2, 14, 20, 11, 7, 9, 5, 6.

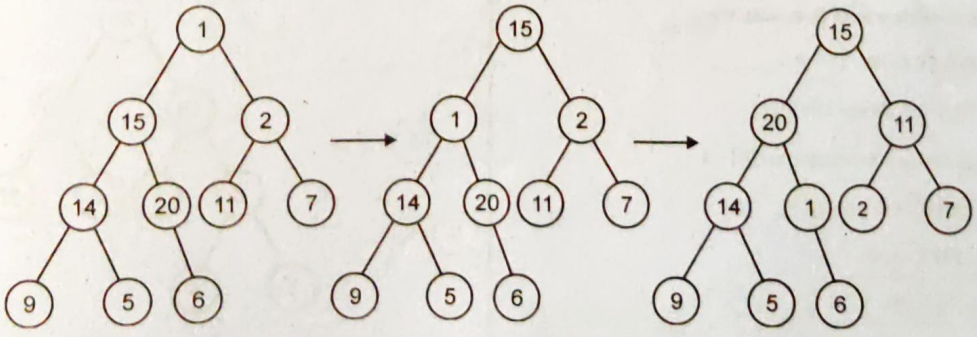
CT : S-09(6M)

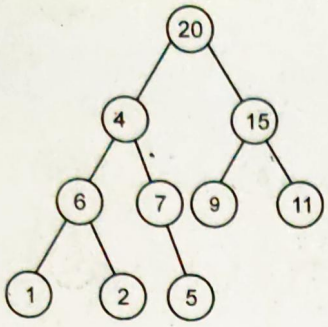
Ans. **Heap tree :** A heap tree is defined to be a binary tree with a key in each node such that :

(1) All leaves are on, at most, two adjacent levels.

(2) All leaves on the lowest level occur to the left and all level except the lowest completely filled.

(3) The key in root is ≥ to all its children and the left and right sub tree are again binary heaps.





Sorted list = 1, 2, 5, 6, 7, 9, 11, 14, 15, 20.

Q.23. Write an algorithm to add a node in a heap tree such that the tree remains a heap tree after insertion. ***CT : W-09(6M)***

OR Write insert, delete and adjust algorithm used in heap sort.

CS :S-10(5M), W-10(7M), W-12(5M), W-13(4M)

Ans. Algorithm to add a node in a heap tree :

```
Algorithm insert (a, n)
{
    //Inserts a[n] into the heap which is sorted in a[1 : n – 1].
    i = n;
    item = a[n];
    while ((i > 1) and (a [⌊ i | 2 ⌋ ] < item) do
    {
        a[i] = a [⌊ i | 2 ⌋ ];
        i = ⌊ i | 2 ⌋ ;
    }
    a [i] = item;
    return true;
}
```

Algorithm to delete a node from heap tree :

HEAP-DELETE (A, i)

Step 1 : A [i] ← A [heap - size [A]]

Step 2 : heap-size [A] ← heap-size [A] – 1

Step 3 : MAX-HEAPIFY (A, i)

MAX-HEAPIFY (A, i)

Step 1 : *l* ← left [i]

Step 2 : r ← right [i]

Step 3 : if *l* ≤ heap-size [A] and A [*l*] > A [i]

Step 4 : then largest ← *l*

Step 5 : else largest ← i

Step 6 : if r ≤ heap-size [A] and A [r] > A [Largest]

Step 7 : then largest ← r

Step 8 : if largest ≠ i

Step 9 : then exchange A[i] ↔ A [Largest]

Step 10 : MAX-HEAPIFY (A, largest)

Algorithm to adjust a node in a heap tree :

HEAP-ADJUST

Step 1 : If heap-size [A] < 1

Step 2 : then error "heap underflow"

Step 3 : MAX ← A [1]

Step 4 : A [1] ← A [heap-size [A]]

Step 5 : heap size [A] ← heap-size [A] – 1

Step 6 : MAX-HEAPIFY (A, 1)

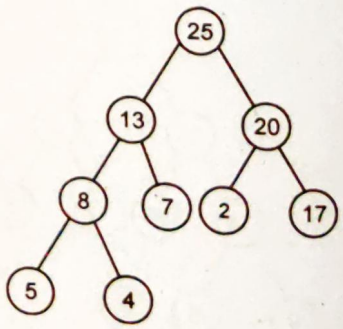
Step 7 : return max.

Q.24. Illustrate the stepwise operation of heap sort on input array A = <5, 13, 2, 25, 7, 17, 20, 8, 4> Also find recurrence relation for the algorithm and find its complexity.

CT: S-06, W-12(9M)

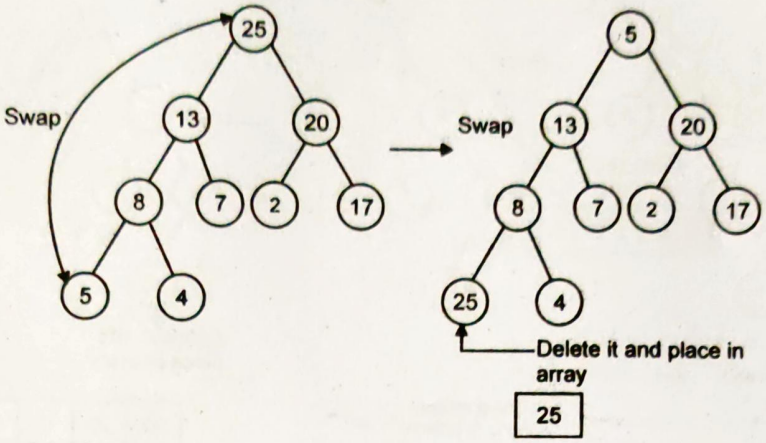
CS : W-10(1M), W-13(2M)

Ans. Heap tree :

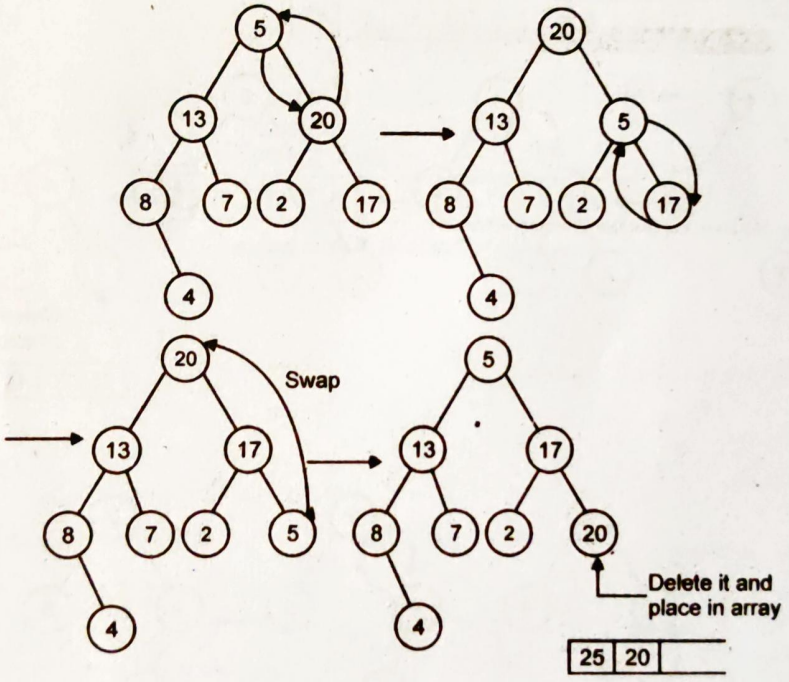


Sorting the list using heap sort and place last deleted element in an array :

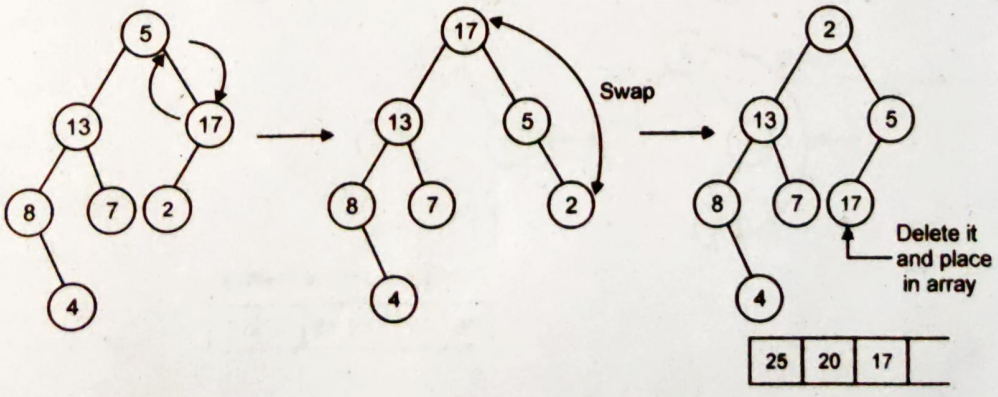
Step 1 :



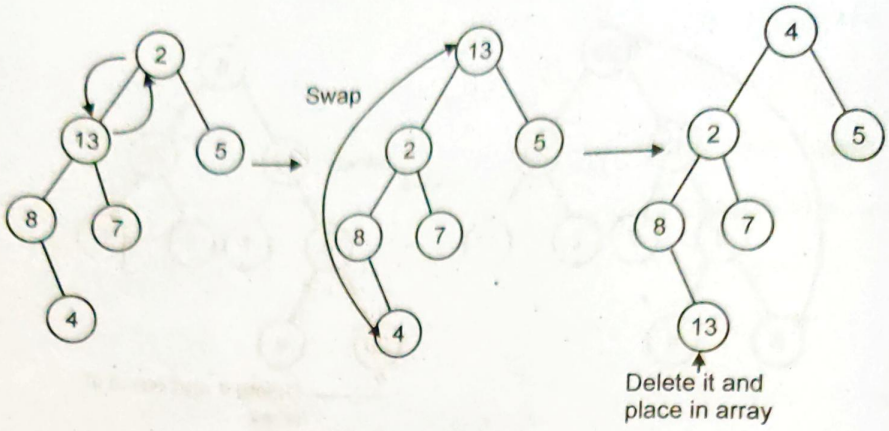
Step 2 :



Step 3 :

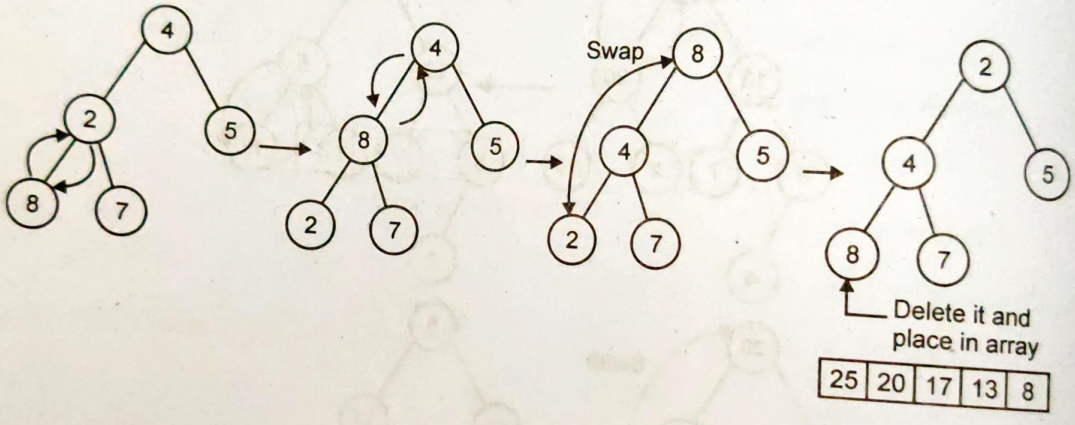


Step 4 :



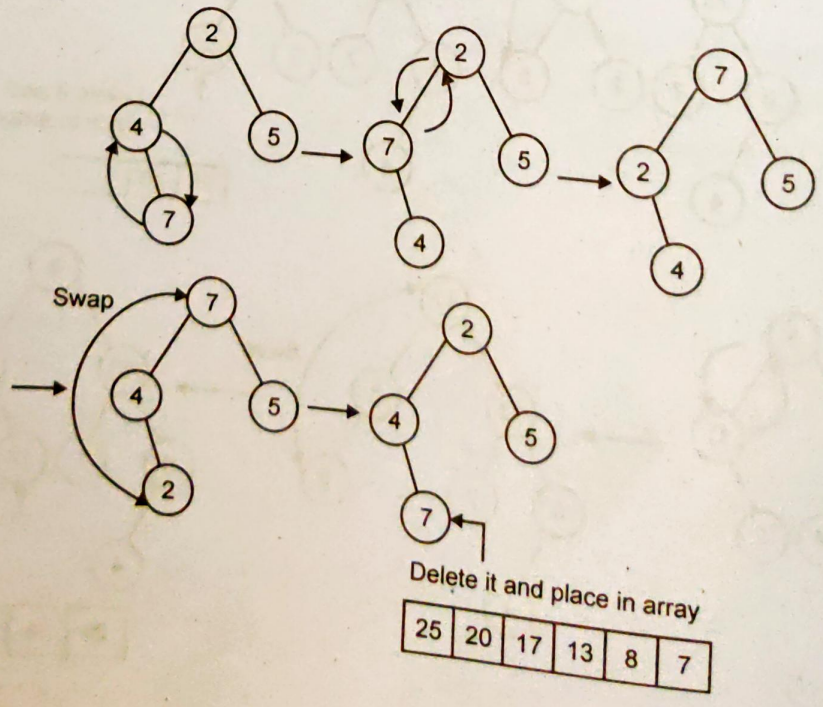
25	20	17	13	

Step 5 :



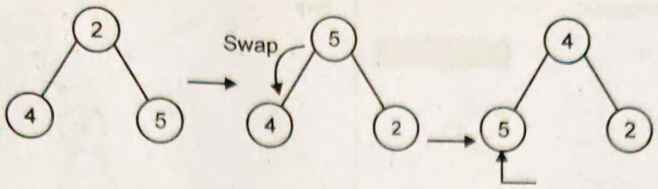
25	20	17	13	8

Step 6 :



25	20	17	13	8	7

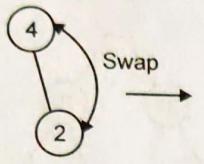
Step 7 :



Delete it & place in array

25	20	17	13	8	7	5	

Step 8 :



Delete it and place in array

25	20	17	13	8	7	5	4	

Step 9 :

Delete it & place in array

25	20	17	13	8	7	5	4	2	

sorted array in descending order

Recurrence Relation :

$t(n) = 2T(n/2) + n$

$\Downarrow$

$c_1.n + c_2.n.\log_2 n$

Let $n = 2^i$

$t(i) = 2t(2^{i/2}) + 2^i$

$t(i) = 2t(2^{i-1}) + 2^i$

$t(i) - 2t(2^{i-1}) + 2^i$

Roots are $(x-2)(x-2)$

$r_1 = 2, \quad r_2 = 2$

$t_i = c_1 r_1^i = c_2.i.r_2^i$

$t_i = c_1 2^i + c_2.i.2^i$

Replace $n = 2^i$ or $i = \log_2 n$

$t(n) = c_1.n + c_2\,(n.\log_2 n)$

$\downarrow$

Largest factor

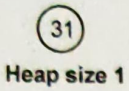
$n.\log_2 n \Rightarrow$ Complexity

Q.25. Design heap tree for following sequence

31, 28, 7, 65, 23, 79, 45, 95. *CS: W-12(4M)*

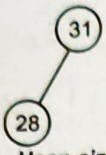
Ans. Heap tree :

Step 1 :



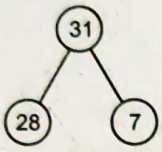
Heap size 1

Step 2 :



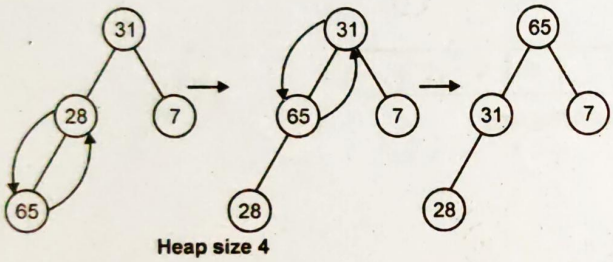
Heap size 2

Step 3 :



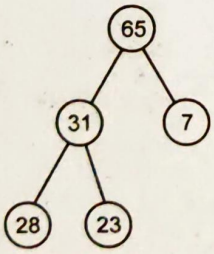
Heap size 3

Step 4 :



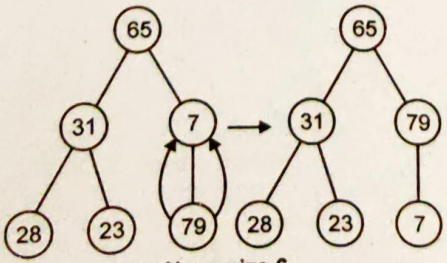
Heap size 4

Step 5 :



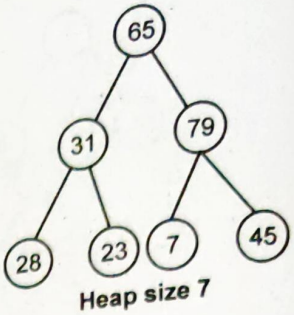
Heap size 5

Step 6 :



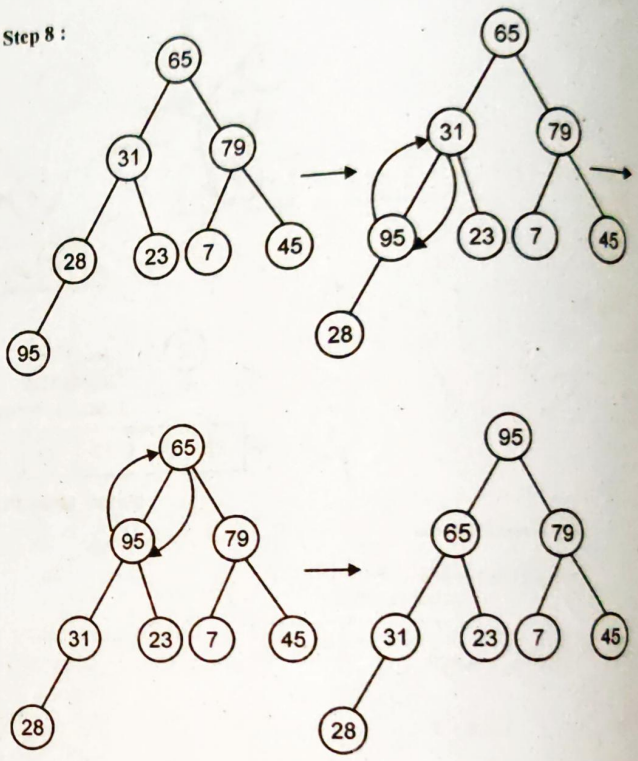
Heap size 6

Step 7 :



Heap size 7

Step 8 :



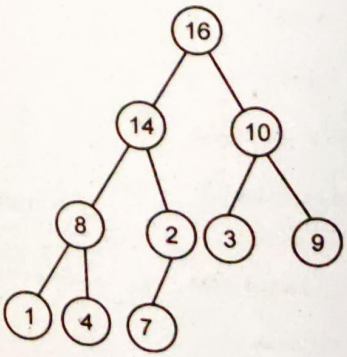
Heap size 8

∴ Heap size 8 is a required max heap.

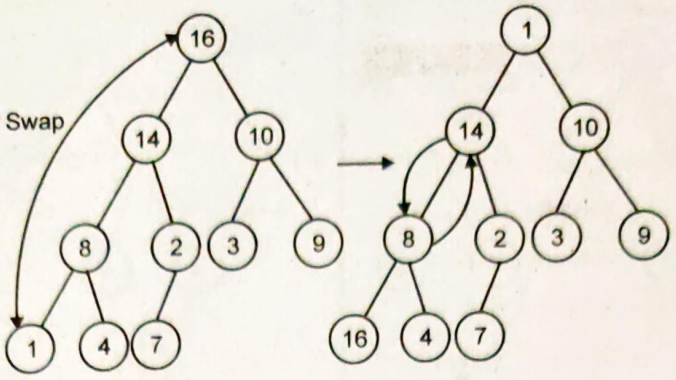
Q.26. Sort the given array using Heap sort algorithm.

4, 1, 3, 2, 16, 9, 10, 14, 8, 7 *CS : S-12(4M)*

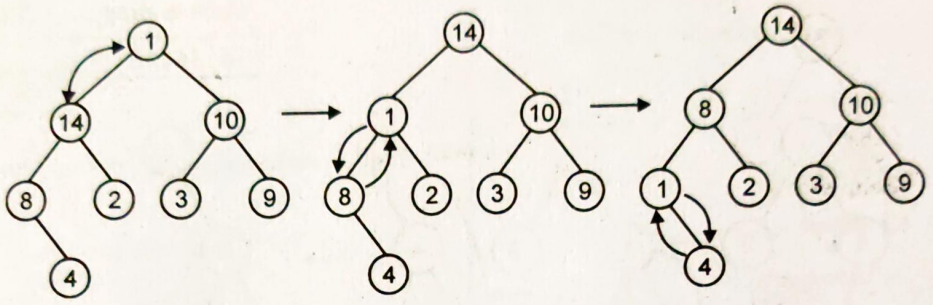
Ans. Heap tree :

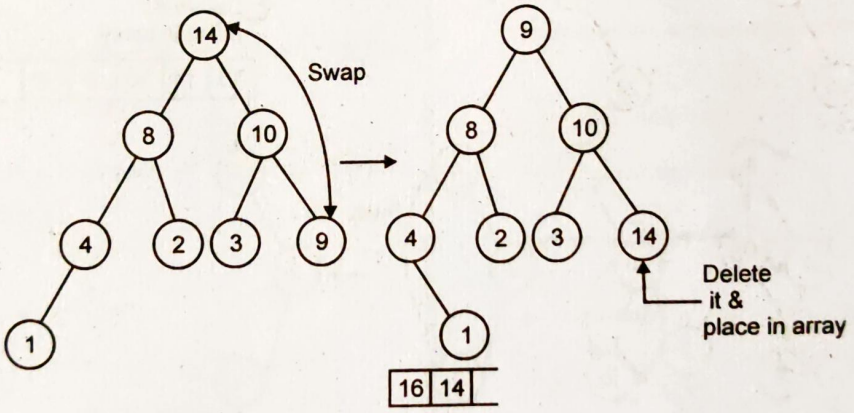


Step 1 :

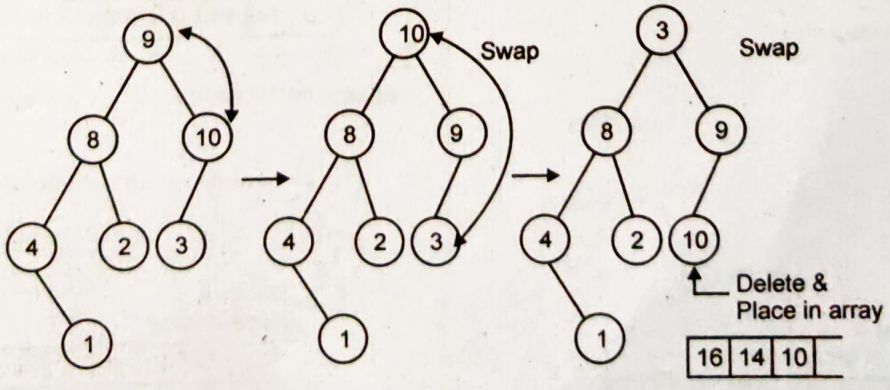


Step 2 :

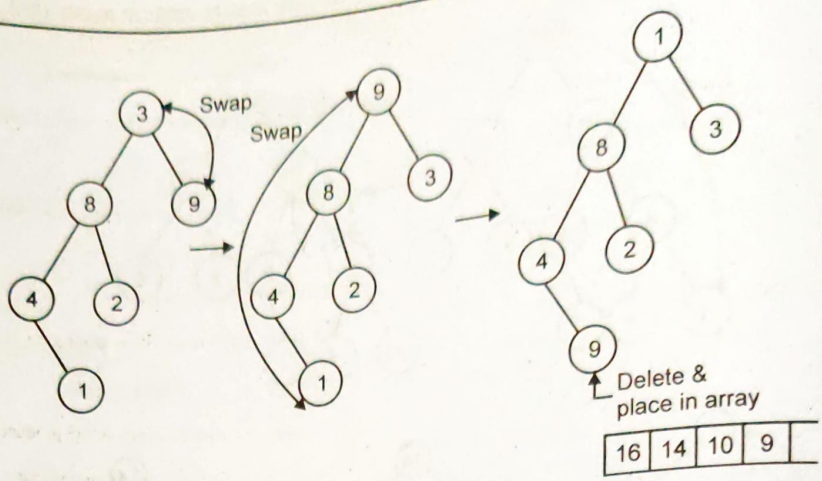




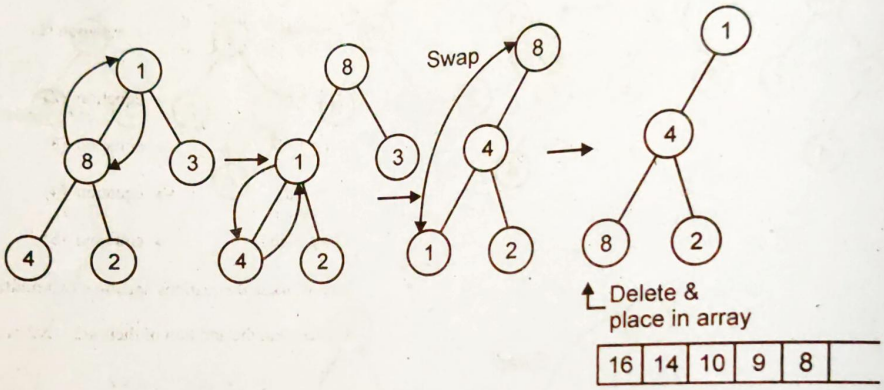
Step 3 :



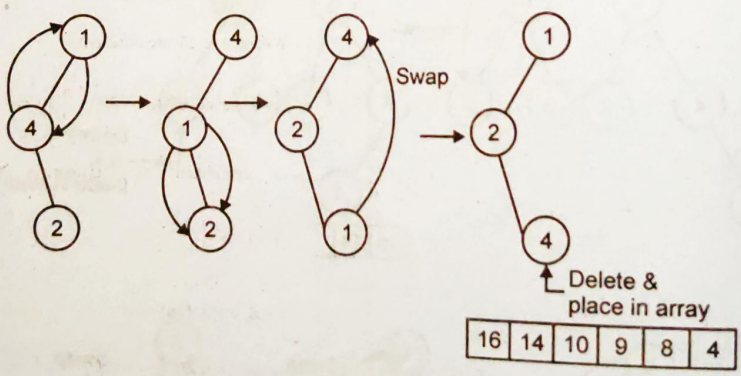
Step 4:



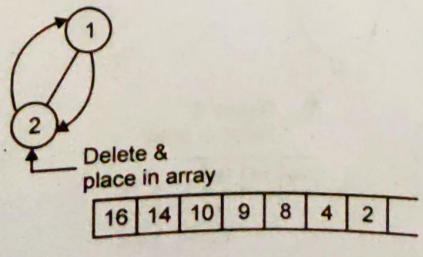
Step 5 :



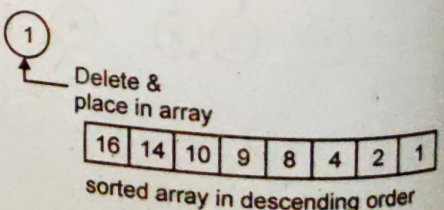
Step 6 :



Step 7 :



Step 8:



Q.27. Prove that n-element heap has height $h = \lfloor \log(n) \rfloor$.

CT: W-13(5M)

Ans. Minimum number of elements in a heap of height h is $h \le n \le 2^{h+1} - 1$

From above we can derive the following constraints by taking the logarithm on both sides.

$$h \le \log n \le \log(2^{h+1} - 1)$$

Since $\log(2^{h+1} - 1) < \log(2^{h+1} - 1) = h + 1$ we can write

$$h \le \log n \le h + 1$$

which is same as saying $h = \lfloor \log \rfloor$

MATRIX OPERATIONS

Q.28. Explain the matrix operation. Also explain its related terms.

Ans. Matrix operations :

- A matrix is a rectangular array of numbers.

Example :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

It is a 3×3 matrix $A = (a_{ij})$, Where for i = 1, 2, 3 and j = 1,2, 3 the element of the matrix in row i and j is a_{ij}.

- The transpose of matrix A is the matrix A^T obtained by exchanging the rows and columns of A.

$$(A^T)_{ij} = A_{ji}$$

$$\begin{bmatrix} 0 & 4 \\ 7 & 0 \\ 3 & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 7 & 3 \\ 4 & 0 & 1 \end{bmatrix}$$

Vector :

Vector is a one dimensional array of numbers. The standard form of a vector is a column vector equivalent to a $n \times 1$ matrix ; the corresponding row vector is obtained by taking the transpose.

Upper triangular :

- An upper triangular matrix U is one for which $u_{ij} = 0$ if $i > j$. All entries below the diagonal are zero :

$$u = \begin{bmatrix} u_{11} & u_{12} \cdots & u_1 n \\ 0 & u_{22} \cdots & u_2 n \\ 0 & 0 \cdots & u_n n \end{bmatrix}$$

- A lower triangular matrix is unit lower triangular if it has all 1's along the diagonal.

MULTIPLICATION ALGORITHM

Q.29. Explain Strassen's matrix multiplication. Give its recurrence relation.

OR Find recurrence relation for matrix multiplication algorithm and time complexity. **CT: W-13(2M)**

Ans. Matrix multiplication :

- It is an application of a familiar design technique 'divide and conquer'.
- Suppose we wish to compute the product C = AB where each A, B and C are $n \times n$ matrices. Assuming that n is an exact power of 2.

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & g \\ f & h \end{bmatrix} \rightarrow \text{equation (1)}$$

$r = ae + bf \rightarrow$ equation (2)

$s = ag + bh \rightarrow$ equation (3)

$t = ce + df \rightarrow$ equation (4)

$u = cg + dh \rightarrow$ equation (5)

- Each of these 4 equations specifics two multiplications of $n/2 \times n/2$ matrices and the addition of their $n/2 \times n/2$ products.

$$T(n) = \begin{cases} b & n \le 2 \\ 8\,T(n/2) + an^2 & n > 2 \end{cases}$$

Where, a and b are constant.

- For recurrence $T(n) = 8\,T\left(\frac{n}{2}\right) + an^2$ if we find running time then it is calculated as comparing the above equation with

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a = 8, b = 2 \quad f(n) = an^2$

$$n \log_b^a = n \log_2^8 = n \log_2^{23} = n^3 \log_2^2 = n^3$$

$$T(n) = O(n^3)$$

Suppose, $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) B_{11}$

$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{22})$

$T = (A_{11} + A_{22}) B_{22}$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (A_{12} - A_{22})(B_{21} + B_{22})$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

The resulting recurrence relation T(n) is :

$$T(n) = \begin{cases} b & n \le 2 \\ 7\,T(n/2) + an^2 & n > 2 \end{cases}$$

$a = 7,\ b = 2\ \ f(n) = an^2$

$n \log_{\frac{a}{b}} = n \log_{\frac{7}{2}} = n^{2.81}$

$T(n) = O(n^{2.81})$

Q.30. Use Strassen's algorithm to compute the matrix product.

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\begin{pmatrix} 6 & 7 \\ 3 & 8 \end{pmatrix}.$$

Ans. $\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\begin{pmatrix} 6 & 7 \\ 3 & 8 \end{pmatrix}$

Here, $A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix} B = \begin{bmatrix} 6 & 7 \\ 3 & 8 \end{bmatrix}$

$P = (1 + 5)(6 + 8)$

$= 6 \times 14$

$\boxed{P = 84}$

$Q = (7 + 5) \times 6$

$= 12 \times 6$

$\boxed{Q = 72}$

$R = 1(7 - 8) = 7 - 8$

$\boxed{R = -1}$

$S = 5(3 - 6)$

$= 5(-3)$

$\boxed{S = -15}$

$T = (1 + 3) \times 8$

$= 4 \times 8$

$\boxed{T = 32}$

$V = (3 - 5)(3 + 8)$

$= -2(11)$

$\boxed{V = -22}$

$C_{11} = P + S - T + V$

$C_{11} = 84 + (-15) - 32 - 22$

$= 84 - 15 - 32 - 22$

$= 84 - 15 - 54$

$= 84 - 69$

$\boxed{C_{11} = 15}$

$C_{12} = R + T$

$= (-1) + 32$

$\boxed{C_{12} = 31}$

$C_{21} = Q + S$

$= 72 + (-15)$

$\boxed{C_{21} = 57}$

$C_{22} = P + R - Q + U$

$C_{22} = 84 - 1 - 72 + 78$

$= 84 - 1 + 6$

$= 83 + 6$

$\boxed{C_{22} = 89}$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{bmatrix} 15 & 31 \\ 57 & 89 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix}\begin{bmatrix} 6 & 7 \\ 3 & 8 \end{bmatrix}$$

Q.31. Show the Strassen's matrix multiplication process on the matrix A and B given below.

$$A = \begin{bmatrix} 4 & 2 & 0 & 1 \\ 3 & 1 & 2 & 5 \\ 3 & 2 & 1 & 4 \\ 5 & 2 & 6 & 7 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 5 & 4 & 2 & 3 \\ 1 & 4 & 0 & 2 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Ans.

- Finding the product matrix C by the usual way, will require $16 \times 4 = 64$ multiplication.

We partition the input matrices into sub-matrices as given below :

$$A=\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix} B=\begin{bmatrix}B_{11} & B_{12}\\ B_{21} & B_{22}\end{bmatrix} C=\begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}$$

where,

$$A_{11}=\begin{bmatrix}4 & 2\\ 3 & 1\end{bmatrix} B_{11}=\begin{bmatrix}2 & 1\\ 5 & 4\end{bmatrix}$$

$$A_{12}=\begin{bmatrix}0 & 1\\ 2 & 5\end{bmatrix} B_{12}=\begin{bmatrix}3 & 2\\ 2 & 3\end{bmatrix}$$

$$A_{21}=\begin{bmatrix}3 & 2\\ 5 & 2\end{bmatrix} B_{21}=\begin{bmatrix}1 & 4\\ 3 & 2\end{bmatrix}$$

$$A_{22}=\begin{bmatrix}1 & 4\\ 6 & 7\end{bmatrix} B_{22}=\begin{bmatrix}0 & 2\\ 4 & 1\end{bmatrix}$$

$D=(A_{11}+A_{22})\ (B_{11}+B_{22})$

$$D=\begin{bmatrix}5 & 6\\ 9 & 8\end{bmatrix}\begin{bmatrix}2 & 3\\ 9 & 5\end{bmatrix}=\begin{bmatrix}64 & 45\\ 90 & 67\end{bmatrix}$$

$E=(A_{21}+A_{22})\ B_{11}$

$$E=\begin{bmatrix}4 & 6\\ 11 & 9\end{bmatrix}\begin{bmatrix}2 & 1\\ 5 & 4\end{bmatrix}=\begin{bmatrix}38 & 28\\ 67 & 47\end{bmatrix}$$

$F=A_{11}\ (B_{12}-B_{22})$

$$F=\begin{bmatrix}4 & 6\\ 11 & 9\end{bmatrix}\begin{bmatrix}2 & 1\\ 5 & 4\end{bmatrix}=\begin{bmatrix}38 & 28\\ 67 & 47\end{bmatrix}$$

$F=A_{11}\ (B_{12}-B_{22})$

$$F=\begin{bmatrix}4 & 2\\ 3 & 1\end{bmatrix}\begin{bmatrix}3 & 0\\ -2 & 2\end{bmatrix}=\begin{bmatrix}8 & 4\\ 7 & 2\end{bmatrix}$$

$G=A_{22}\ (B_{21}-B_{11})$

$$G=\begin{bmatrix}1 & 4\\ 6 & 7\end{bmatrix}\begin{bmatrix}-1 & 3\\ -2 & -2\end{bmatrix}=\begin{bmatrix}-9 & -5\\ -20 & 4\end{bmatrix}$$

$H=(A_{11}+A_{12})\ B_{22}$

$$=\begin{bmatrix}4 & 3\\ 5 & 6\end{bmatrix}\begin{bmatrix}0 & 2\\ 4 & 1\end{bmatrix}=\begin{bmatrix}12 & 11\\ 24 & 16\end{bmatrix}$$

$M=(A_{21}-A_{11})\ (B_{11}+B_{12})$

$$=\begin{bmatrix}-1 & 0\\ 2 & 1\end{bmatrix}\begin{bmatrix}5 & 3\\ 7 & 7\end{bmatrix}=\begin{bmatrix}-5 & -3\\ 17 & 13\end{bmatrix}$$

$P=(A_{12}-A_{22})\ (B_{21}+B_{22})$

$$=\begin{bmatrix}-1 & -3\\ -4 & -2\end{bmatrix}\begin{bmatrix}1 & 6\\ 7 & 3\end{bmatrix}=\begin{bmatrix}-22 & -15\\ -18 & -30\end{bmatrix}$$

$C_{11}=D+G-H+P$

$$=\begin{bmatrix}64 & 45\\ 90 & 67\end{bmatrix}+\begin{bmatrix}-9 & -5\\ -20 & 4\end{bmatrix}-\begin{bmatrix}12 & 11\\ 24 & 16\end{bmatrix}+\begin{bmatrix}22 & 15\\ 18 & 30\end{bmatrix}$$

$$=\begin{bmatrix}64 & 45\\ 90 & 67\end{bmatrix}+\begin{bmatrix}-9 & -5\\ -20 & 4\end{bmatrix}$$

$$+\begin{bmatrix}-12 & -11\\ -24 & -16\end{bmatrix}+\begin{bmatrix}-22 & -15\\ -18 & -30\end{bmatrix}$$

$$=\begin{bmatrix}21 & 14\\ 28 & 25\end{bmatrix}$$

$C_{12}=F+H$

$$=\begin{bmatrix}8 & 4\\ 7 & 2\end{bmatrix}+\begin{bmatrix}12 & 11\\ 24 & 16\end{bmatrix}$$

$$=\begin{bmatrix}20 & 15\\ 31 & 18\end{bmatrix}$$

$C_{21}=E+G$

$$=\begin{bmatrix}38 & 28\\ 67 & 47\end{bmatrix}+\begin{bmatrix}-9 & -5\\ -20 & 4\end{bmatrix}$$

$$=\begin{bmatrix}29 & 23\\ 47 & 51\end{bmatrix}$$

$C_{22}=D+F-E+M$

$$=\begin{bmatrix}64 & 45\\ 90 & 67\end{bmatrix}+\begin{bmatrix}8 & 4\\ 7 & 2\end{bmatrix}+\begin{bmatrix}-38 & -28\\ -67 & -47\end{bmatrix}$$

$$+\begin{bmatrix}-5 & -3\\ -17 & 13\end{bmatrix}$$

$$=\begin{bmatrix}29 & 18\\ 47 & 35\end{bmatrix}$$

$$C=\begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}=\begin{bmatrix}21 & 14 & 20 & 15\\ 28 & 25 & 31 & 18\\ 29 & 23 & 29 & 18\\ 47 & 51 & 47 & 35\end{bmatrix}$$

Q.32. Illustrate the matrix multiplication using strassen's method on the following matrices.

$$A=\begin{bmatrix}2 & 3\\ 4 & 5\end{bmatrix} B=\begin{bmatrix}3 & 4\\ 2 & 1\end{bmatrix}.$$

Ans. $P_1=(2+5)\times(3+1)=7\times4=28$

$P_2=(4+5)\times3=9\times3=27$

$P_3 = 2(4-1) = 6$

$P_4 = 5(2-3) = -5$

$P_5 = (2+3) \times 1 = 5$

$P_6 = (3-2)(3+4) = 7$

$P_7 = (3-5)(2+1) = -6$

$C_{11} = 28 - 5 - 5 - 6$

$= 12$

$C_{12} = 06 + 05 = 11$

$C_{21} = 27 - 5 = 22$

$C_{22} = 28 + 6 - 27 + 7$

$= 14$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 12 & 11 \\ 22 & 14 \end{bmatrix}$$

Q.33. Write algorithm for finding minimum and maximum element from array using DAC. Explain the complexity of algorithm. Implement the algorithm on the following array and draw min-max tree.

123, 82, 25, –20, –62, 43, 172, 95, 57, – 45

CS : S-10(6M), W-10(5M)

OR Write DAC based min-max algorithm to find the minimum and maximum value from given array

123, 82, 25, – 20, – 62, 43, 172, 95, 57, – 45.

CS : W-13(7M)

Ans. Min-max algorithm :

- This algorithm will find the minimum as well as maximum value.
- It is based on the principle of binary search in which the given values will be divided into two sets to find min and max value.

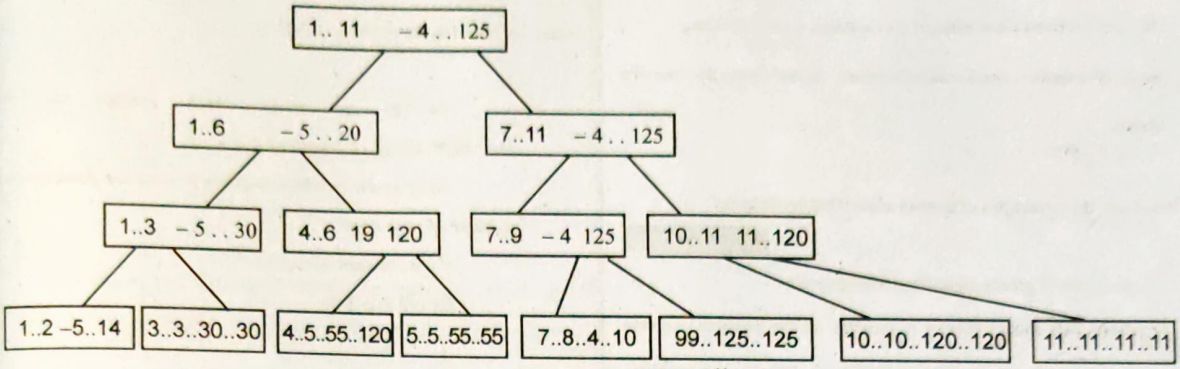
Algorithm rec-min-max(a, low, high)

```
{
    i = 1, j = n;
    if (i = j) then
    {
        min = max = a [i];
    }
    if (i = j – 1) then
    {
        min = a [i];
        max = a [j];
    }
    else
    {
        min = a [j];
        max = a [i];
    }
else
    {
        mid = (i + j) / 2 ;
        rec-min-max (a, 1, mid);
        rec-min-max (a, mid + 1, n);
    }
}
```

Q.34. For the following sequences of integers generate MIN-MAX tree.

14, – 5, 30, 120, 55, 19, 10, – 4, 125, 120, 11 ***CS : W-12(3M)***

Ans.



Complexity $= \log_2^n = \log_2^{11}$

GREEDY METHOD

Q.35. Explain basic strategy of greedy method.

Ans. Greedy method basic strategy :

- Greedy algorithms solve problems by making the choice that seems best at the particular moment.
- Many optimization problems can be solved using a greedy algorithm.
- Some problems have no efficient solution but a greedy algorithm may provide a solution that is close to optimal.
- A greedy algorithm works if a problem exhibits the following properties :

(1) **Greedy choice property :** A globally optimal solution can be arrived at by making a locally optimal solution.

(2) **Optimal substructure :** Optimal solutions contains optimal sub solutions.

Q.36. Discuss the general characteristics of a greedy algorithm and give the general format. ***CT : W-09(6M), W-11(4M)***

OR Explain greedy algorithm. Write down the characteristics of greedy algorithm. ***CT : W-07(5M), W-08(4M)***

Ans. Greedy algorithm :

- Greedy algorithms are simple and straight forward.
- They are short sighted in their approach in the sense that they take decisions on the basis of information in hand without worrying about the effect these decisions may have in the future.
- They are easy to implement and most of the time quite efficient.
- Many problems can be solved by greedy approach correctly.
- Greedy algorithms are used to solve optimization problems.

Algorithm Greedy(a, n)

// a[1 : n] contains the n inputs.

```
{
    solution := ϕ ; // Initialize the solution.
    for i := 1 to n do
    {
        x := Select(a);
        if Feasible(solution, x) then
        solution := Union(solution, x);
    }
    return solution;
}
```

Characteristics of Greedy algorithms :

- To construct the solution in an optimal way, algorithm maintains two sets.

Set 1 : Contains chosen items.

Set 2 : Contains rejected items.

- A greedy algorithm consists of four function to find optimal solution.

(1) A function that checks whether chosen set of item provides a solution.

(2) A function that checks the feasibility of a set.

(3) The selection function tells which of the candidates is most promising.

(4) An objective function which does not approach explicitly gives the value of a solution.

Q.37. What are the principles of greedy algorithm techniques? **CT : W-11(3M)**

Ans. The principles of greedy algorithm techniques are :

(1) **Feasible :** We check whether it satisfies all the possible problem constraints or not so as to obtain at least one solution to our problem.

(2) **Local optimal choice :** In this, choice should be the optimum which is selected from the current available feasible choices.

(3) **Unalterable :** Once the choice is made at any subsequent step that choice is not altered.

Q.38. Explain any three differences between greedy and divide and conquer methods of algorithm design. **CS : W-11(3M)**

Ans.

Sr. No.	Divide and conquer	Greedy
(1)	Divide and conquer break the problem in small problem.	Greedy algorithm are straight forward.
(2)	They take decisions which may or may not work.	They take decision on the basis of information immediately without worrying about the effect.
(3)	They are neat short sighted.	They are shortsighted.

Q.39. What are the applications of greedy algorithm? **CT : W-13(3M)**

Ans. Applications of Greedy Algorithm :

(1) Ordering projects by deadlines.

(2) Compressed video using Huffman coding.

(3) Packet scheduling.

(4) Packet routing.

(5) Removing noisy data for training neural Network.

(6) Image compression by using Huffman coding.

Q.40. Give the denominations of Rs. 100, Rs. 50, Rs. 20, Rs. 10, Rs. 5, Rs. 2 and Rs. 01 Design an algorithm using a greedy approach to find the number of notes of each denomination for a sum of 'n' rupee. **CT : S-09(6M)**

Ans.

```
Algorithm denomination(A,B,n)
{
    //A is an array which contain the value
[100,50,20,10,5,2,1], length of A is n.
    //B is an array which is going to store the denomination of
number of each notes.
    // n is the sum of rupees.
    for i=0 to n-1 do
        {
            if(n>=A[i])
            {
                r=n%A[i];
                q=n/A[i];
                B[i]=q;
                n=r;
            }
            else
                continue;
        }
}
```

ACTIVITY SELECTION PROBLEM

Q.41. Explain activity selection problem with algorithm. Also state its complexity.

Ans. Activity Selection Problem :

- An activity selection is the problem of scheduling a resource among several competing activity.
- The activity selection problem is also defined as, "Given a set 's' of 'n' activities with start time si and fi finish time of an i^{th} activity. The problem is to find the maximum size set of mutually compatible activities.

Activity selection problem :

- A set s = {1, 2, ... n} proposed activities.
- Everyone want the same resource used by one at a time.
- Each activity i has start time si and finish time fi where si $\leq$ fi half open interval [si, fi].
- Two activities i and j are compatible if they donot overlap.

Algorithm :

Greedy-Activity Selector (s, f)

Step 1 : n ← length [s]

Step 2 : A ← [i]

Step 3 : j ← 1

Step 4 : for i ← 2 to n

Step 5 : do if si ≥ fi

Step 6 : then A ← A ∪ {i}

Step 7 : j = i

Step 8 : return A.

Complexity : Sort the input activity by increasing finish time requires O(n lg n) time and greedy activity selector (s, f) requires O(n) time.

Q.42. Activities are given S = {p, q, r, s, t, u, v, w, x, y, z} start and finished time for proposed activities are (1, 4) (3, 5) (4, 6) (5, 7) (3, 8) (7, 9) (10, 11) (8, 12) (8, 13) (2, 14) and (13, 15). Apply greedy strategy for the solution to activity selection problem.

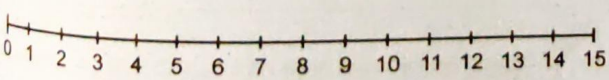
Ans. Given : s = {p, q, r, s, t, u, v, w, x, y, z}

n = 11 i.e. total number of activities.

Sr. No.	Activities (Si)	Start time (si)	Finish time (fi)
(1)	p	1	4
(2)	q	3	5
(3)	r	4	6
(4)	s	5	7
(5)	t	3	8
(6)	u	7	9
(7)	v	10	11
(8)	w	8	12
(9)	x	8	13
(10)	y	2	14
(11)	z	13	15

(1) Sort the input activities by increasing finish time

4 < 5 < 6 < 7 < 8 < 9 < 11 < 12 < 13 < 14 < 15.



(2) Call a procedure Greedy activity - selector (s, f)

n ← Length [s]

n ← 11

n = 11

A ← [i]

A ← 1

j ← 1

For i ← 2 to 11

do if si ≥ fi

then A ← A ∪ {i}

A = {1} fi = 4

A = {1, 3} fi = 6

A = {1, 3, 6} fi = 9

A = {1, 3, 6, 7} fi = 11

A = {1, 3, 6, 7, 11} fi = 15

Set j = i

j = i = 11

since A = {1, 3, 6, 7, 11}

return A

⇒ return {1, 3, 6, 7, 11}

Thus, this algorithm is greedy because it always picks the activity with the earliest compatiable finish.

KNAPSACK PROBLEM

Q.43. Explain what is knapsack problem. Give algorithm for it and time complexity for the algorithm. *CT : S-11(6M)*

OR Give Greedy algorithm for knapsack problem.

CT : W-07(3M)

OR Write algorithm for best approach. *CS : W-10(3M)*

OR Write are algorithm for evaluation of partial knapsack.

CS : S-14(4M)

Ans. Knapsack Algorithm :

- Knapsack is a bag of fixed capacity capable of storing certain object.
- The object can be stored completely or filled partially.
- There are three different approaches for knapsack problem :

(1) Select the object with maximum profit.

(2) Select the object with minimum weight.

(3) Select the object with maximum (pi/wi) ratio.

Knapsack algorithm :

Algorithm knapsack (w [1 . . n], p[1 ...n]: x[1, n]: w, weight)

{

for i : = 1 to n do

```
x [i] = 0
weight = 0
while (weight < w)
{
if(weight + w[i] ≤ w)
    {
        x [i] = 1
        weight = weight + w[i]
    }
    else
    {
    x[i] = (w – weight) / w[i]
    weight : = w
    }
    for i : = 1 to n
    net = p[i] * x[i]
    }
}
```

- Time complexity of knapsack = O(n log n)

Q.44. What is the significance of knapsack problem? Implement three approaches on following objects and find out the profit value. At total capacity = 16, Number of objects = 7.

i	1	2	3	4	5	6	7
Pi	10	15	12	4	6	16	8
Wi	2	4	5	4	2	3	3
Pi/Wi	5	3.75	2.4	1	3	5.33	2.66

CS : W-10(6M)

Ans. Significance of knapsack problem : Knapsack is a bag having fixed capacity. The main objective is to fill the bag such that maximum profit is achieved.

(1) Maximum profit :

Object	Profit	Weight	Capacity left (16)
06	16	3	13
02	15	4	9
03	12	5	4
01	10	2	2
07*	5.33	2	0
	58.33		

(2) Minimum weight :

Object	Profit	Weight	Capacity
01	10	2	14
05	6	2	12
06	16	3	9
07	8	3	6
02	15	4	2
04*	2	2	0
	57		

(3) Pi/Wi :

Object	Profit	Weight	Capacity
06	16	3	13
01	10	2	11
02	15	4	7
05	6	2	5
07	8	3	2
03*	4.8	2	0
	59.8		

Q.45. Solve the following knapsack problem

n = 3, m = 20

$(P_1, P_2, P_3) = (25, 24, 15)$

$(W, W_2, W_3) = (18, 15, 10)$

CT : W-07(5M), S-11(3M)

Ans. **Given :** m = 20

n = 3

Object	0_1	0_2	0_3
Profit	25	24	15
Weight	18	15	10

Select the object with highest profit.

Sr. No.	(x_1, x_2, x_3)	$\sum w_i x_i$	$\sum p_i x_i$
(1)	(1/2, 1/3, 1/4)	16.5	24.25
(2)	(1, 2/15, 0)	20	28.2
(3)	(0, 2/3, 1)	20	31
(4)	(0, 1, 1/2)	20	31.5

- Object 1 has the largest profit value $(p_1 = 25)$. So it is placed into the knapsack first. Then $x_1 = 1$ and profit of 25 is earned.
- Only 2 units of knapsack capacity are left. Object 2 has the next largest profit $(p_2 = 24)$.

However $w_2 = 15$ and it does not fit into the knapsack.

Using $x_2 = 2/15$ fills the knapsack exactly with part of object 2 and the value of the resulting solution is 28.2.

Q.46. Consider five items along their respective weights and values as follows :

Item	I_1	I_2	I_3	I_4	I_5
Weight	05	10	20	30	40
Value	30	20	100	90	160

The capacity of knapsack is 60

Find the solution for fractional knapsack problem using greedy approach. ***CT : S-12(7M)***

Ans. Capacity = 60

Item	I_1	I_2	I_3	I_4	I_5
Value (P_i)	30	20	100	90	160
Weight (W_i)	05	10	20	30	40
Ratio $\left(\frac{P_i}{W_i}\right)$	6	2	5	3	4

Method I : Solution with maximum profit.

Object	Profit	Weight	Capacity left
O_5	160	40	60 – 40 = 20
O_3	100	20	20 – 20 = 00
	260		

Method II : Solution with minimum weight.

Object	Profit	Weight	Capacity left
O_1	30	05	60 – 05 = 55
O_2	20	10	55 – 10 = 45
O_3	100	20	45 – 20 = 25
O_4 *	75	25	25 – 25 = 00
Partially	225		

Method III : Solution with maximum ratio

Object	Profit	Weight	Capacity left
O_1	30	05	60 – 05 = 55
O_3	100	20	55 – 20 = 35
O_5 *	140	35	35 – 35 = 00
Partially	270		

Q.47. Find an optimal solution to the Knapsek instance n = 7, m = 15

$(p_1, p_2, p_3, \ldots p_7) = (10, 5, 15, 7, 6, 18, 3)$ and

$(w_1, w_2, w_3, \ldots w_7) = (2, 3, 5, 7, 1, 4, 1)$ ***CT : S-10(8M)***

Ans.

	1	2	3	4	5	6	7
Profit	10	5	15	7	6	18	3
Weight	2	3	5	7	1	4	1

Number of objects n = 7

Capacity m = 15

	1	2	3	4	5	6	7
$p_i / w_i \Rightarrow$	5	1.66	3	1	6	4.5	3

Method I:- Solution with maximum profit.

Object	Profit	Weight	Capacity left
06	18	4	15 – 4 = 11
03	15	5	11 – 5 = 6
01	10	2	6 – 2 = 4
04*	4	4	4 – 4 = 0
Partially	47		

Method II:- Solution with maximum weight.

Object	Profit	Weight	Capacity left
05	6	1	15 – 1 = 14
07	3	1	14 – 1 = 13
01	10	2	13 – 2 = 11
02	5	3	11 – 3 = 8
06	18	4	8 – 4 = 4
03 *	12	4	4 – 4 = 0
(Partially)	54		

Method III:- Solution with maximum ratio

Object	Profit	Weight	Capacity left
05	6	1	15 – 1 = 14
01	10	2	14 – 2 = 12
06	18	4	12 – 4 = 8
03	15	5	8 – 5 = 3
07	3	1	3 – 1 = 2
02*	3.3	2	2 – 2 = 0
(Partially)	55.3		

Q.48. Find optimal solution to the knapsack instance n = 7, m = 15

$(P_1, P_2, P_3, \ldots P_7) = (15, 20, 10, 7, 6, 18, 3)$ and

$(W_1, W_2, W_3, \ldots W_7) = (2, 3, 5, 7, 1, 4, 1)$. *CS: S-10(8M)*

Ans. **Capacity = 15**

Item	I_1	I_2	I_3	I_4	I_5	I_6	I_7
Profit	15	20	10	7	6	18	3
Weight	2	3	5	7	1	4	1
Ratio (Pi/Wi)	7.5	6.66	2	1	6	4.5	3

Method I : Solution with maximum profit

Object	Profit	Weight	Capacity left
02	20	3	12 (15 – 3)
06	18	4	8 (12 – 4)
01	15	2	6 (8 – 2)
03	10	5	1 (6 – 5)
04*	1	1	0 (1 – 1)
Partially	64		

Method II : Solution with maximum weight

Object	Profit	Weight	Capacity left
05	6	1	14
07	3	1	13
01	15	2	11
02	20	3	8
06	18	4	4
03 *	8	4	0
(Partially)	70		

Method III : Solution with maximum ratio

Object	Profit	Weight	Capacity left
01	15	2	13
02	20	3	10
05	6	1	9
06	18	4	5
07	3	1	4
03*	8	4	0
(Partially)	70		

Q.49. For the following set of objects implement partial knapsack problem, with maximum capacity of 18 :

Object	Profit	Weight	Pi/Wi
1	9	2	4.5
2	15	3	5
3	12	5	2.4
4	4	4	1
5	6	3	2
6	16	6	2.66
7	8	3	2.66

Explain all three methods and comment on their profit value.

CT : S-14(8M)

Ans.

(1) Minimum profit :

Object	Profit	Weight	Capacity (18)
06	16	6	12
02	15	3	9
03	12	4	5
01	9	2	3
07	8	3	0
	60		

(2) Minimum weight :

Object	Profit	Weight	Capacity (18)
01	9	2	16
02	15	3	13
07	8	3	10
05	6	3	7
04	4	4	3
03	7.2	3	0
	49.2		

(3) Pi/Wi :

Object	Profit	Weight	Capacity (18)
02	15	3	15
01	9	2	13
06	16	6	7
07	8	3	4
07	8	3	4
03*	9.6	4	0
	57.6		

All of the above method pi/wi is the best method.

Q.50. Explain job scheduling problem with example.

CT : S-10(8M), W-10(5M), CS : S-11(3M)

OR Describe job scheduling problem in brief.

CT : S-06(6M), W-08(4M)

Ans. Job scheduling :

- In job scheduling there are various jobs to be processed using one processor.
- The main objective is to reduce processing time.

There are two methods of job scheduling :

(1) Job scheduling without deadline.

(2) Job scheduling with deadline.

(1) **Job scheduling without deadline :**

There is no deadline associated with the jobs but the jobs should be so scheduled that their processing time should be minimum.

(2) **Job scheduling with deadline :**

- With each job a deadline and profit value is associated.
- The profit value is valid if the job is executed before or at the deadline.
- Each job will require one cycle for execution.
- Job scheduling problem is the problem of optimally scheduling unit time takes on a single processors, where each task has a deadline, along with a penalty that must be paid if the deadline is missed.
- For the solution of this matroids can be used which follows greedy strategy.
- In job scheduling problem a "unit time task (job) is run on a computer.
- Let us assume that P be a finite set of unit time task (job) a schedule for P is a permutation of P which specifies the order in which these tasks are to be performed.
- The first task begins at time 0 and finishes at time 1, the second task begins at time 1 and finishes at time 2 and so on.

$d_i \Rightarrow$ Represent deadline

$w_i \Rightarrow$ Weight.

Example :

Let,

$n = 5, (P_1, P_2, P_3, P_4, P_5) = (20, 15, 10, 5, 1)$

$(d_1, d_2, d_3, d_4, d_5) = (2, 2, 1, 3, 3)$.

Find the optimal schedule.

i	1	2	3	4	5
d_i	2	2	1	3	3
p_i	20	15	10	5	1

Here, maximum time for computer is 3. Sort in decreasing order of profit.

Step 1:

	T_1	T_2	T_4	Profit
Time →	0 1	2	3	40

$P_1 + P_2 + P_4$

Step 2: $d_1 = 2 \quad d_2 = 2 \quad d_3 = 3$

	T_1	T_2	T_3
Time →	0 1	2	3

Profit : $P_1 + P_2 + P_3$

$\Rightarrow 20 + 10 + 5$

$\Rightarrow 35$

Step 3:

	T_1	T_2	T_3
Time →	0 1	2	3

Profit : $(P_1 + P_2 + P_5)$

$= 36$

Thus, we can find maximum profit 40 by implementing schedule as (1, 2, 4). Therefore (1, 2, 4) is optimal schedule.

Q.51. What is the best possible sequence generated by the job scheduling approach when n = 7,

$(P_1, P_2, \ldots\ldots, P_7) = (3, 5, 20, 18, 1, 6, 30)$

and $(d_1, d_2, \ldots. d_7) = (1, 3, 4, 3, 2, 1, 2)$.

CT : W-10(7M), W-12(3M), CS : S-12(5M)

Ans.

Jobs	1	2	3	4	5	6	7
Profit	3	5	20	18	0	6	30
Deadline	1	3	4	3	2	1	2

Maximum deadline 4

Step 1: Number of slots

Step 2: Sort the jobs in descending order of profit.

	J_7	J_3	J_4	J_6	J_2	J_1	J_5
Jobs	7	3	4	6	2	1	5
Profit	30	20	18	6	5	3	1
Deadline	2	4	3	1	3	1	2

Step 3:

$\frac{J_3 = 20}{D = 4}$

$J_7 = 30$	$J_3 = 20$		

$\frac{J_4 = 18}{D = 3}$

		S_3	
$J_7 = 30$	$J_3 = 20$	$J_4 = 18$	

$\frac{J_6 = 6}{D = 1}$

$J_6 = 1$	$J_7 = 30$	$J_4 = 18$	$J_3 = 20$

We can't move J_4 since slot number and job number matches i.e. 3.

The sequence is $J_6 - J_7 - J_4 - J_3$

Q.52. Give the greedy algorithm for job sequencing with deadline.

CT : W-06(8M), CS : S-10,14(5M), W-13(6M)

OR Design an algorithm using the Greedy approach for a job scheduling problem with deadline. Work out the algorithm for the given below six jobs to obtain the maximum profit and respecting the deadlines.

Job	1	2	3	4	5	6
Gi(profit)	20	15	10	7	5	3
Di(deadline)	3	1	1	3	1	3

CT : W-09(7M), W-12(3M)

CS : S-13(8M), W-13(6M)

Ans. Job scheduling with deadline :

```
Algorithm JS [d [0 . .n] : J [0 . .n]]
{
d [0] = 0; J [0] = 0;
K = 1, J [1] = 1,   // First Job always selected
for i = 2 to n do
{
        r = k
        while {(d [J [r] ] ) > max. (d [i], r)}
            r = r – 1
        if (d[i] > r) then
        for m = k to r + 1
        j [r + 1] = i
                k = k + 1
    }
}
```

Example :

Job No	gi (profit)	di (deadline)
1	20	3
2	15	1
3	10	1
4	7	3
5	5	1
6	3	3

Soln. Profit (maximum) + Deadline

J ⇨

2	1	4
J(1)	J(2)	J(3)

Total profit = 20 + 15 + 7 = 42

T = 1

1	2	3
J(1)		
(3)		

T = 2

T = 1	T = 2	T = 3
~~J_2~~	J_1	
1		

T = 3

J_2	J_1	J_4
2	1	4

Q.53. Find the best possible sequence for the following deadlines.

Job	Gain	Deadline
1	35	3
2	20	1
3	18	3
4	16	4
5	12	2
6	10	2
7	8	1

Ans. ***CS : S-11(6M)***

Step 1 : Maximum deadline = 4

Number of slots = 4

Step 2 : Sort the jobs in descending order of profit. Already in sorted form.

Step 3 :

J_1			

D = 3

J_1	J_2		

D = 3 D = 1

J_2	J_1	J_3	

D = 1 D = 3 D = 3

J_2	J_1	J_3	J_4

D = 1 D = 3 D = 3 D = 4

The sequence is $J_2 - J_1 - J_3 - J_4$

Q.54. For the following sequence of job, give the snapshot of execution which will achieve maximum profit :

Gain	15	10	20	7	5	3
Deadline	1	1	3	3	1	3

CT : S-14(4M)

Ans.

(1) Profit (maximum) + Deadline.

Number of slots = 3 *[∵ Maximum deadline = 3]*

(2) Arrange the jobs in decreasing order of profit.

Jobs	1	2	3	4	5	6
Gain	20	15	10	7	5	3
Deadline	3	1	1	3	1	3

Refer job 1 — Job 1 can be delayed up 3rd cycle.

S_1 S_2 S_3

J_1		

P = 20

d = 3

Refer job 2 — The deadline of job 2 is D = 1 But the slot 1 is already occupied by job 1. Deadline of job 1 is D = 3 Hence job 1 can be shifted to slot – 2.

S_1 S_2 S_3

J_2	J_1	

P = 15 P = 20

D = 1 D = 3

Refer job 3 — Require slot 1, since slot 1 occupied by higher profit job ∴ job 3 is rejected.

P = 10 D = 1

Refer job 4

Slot 3 : free

J_2	J_1	J_4

P = 15 P = 20 P = 7

D = 1 D = 3 D = 3

Total profit = 42

Sequence = $J_2 - J_1 - J_4$

MINIMUM COST SPANNING TREE

Q.55. What is minimum cost spanning tree? What are its types?

CT : S-06(3M), W-06,08,12, S-11,13(2M), CS : S-14(2M)

Ans. Minimum cost spanning tree (MCST) :

- A minimum spanning tree or minimum weight spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree.
- It is a tree generated from graph with following properties.

(1) There is no loop or tree is acyclic.

(2) All the vertices present in the graph will be present in the tree.

(3) The total number of edges will be equal to (n – 1) if there are n vertices in the graph.

(4) The tree will be generated by considering the edges of minimum cost.

- There are two methods of MCST :

  (i) Kruskal method.

  (ii) Prims method

(i) Kruskal method :

- This method will select the edge with minimum weight to begin tree.
- The processes will continue with next edge with minimum weight.
- The tree generated is called as minimum cost spanning tree.
- This may generate isolated components in the graph which will be finally connected.

(ii) **Prims method :**

- In this algorithm the spanning tree will be generated by selecting the edge with minimum cost.
- The process is continued from one of vertices of the selected edge which leads to minimum cost.

Q.56. Give an algorithm to find the minimum cost spanning tree using Kruskal's method. Discuss the time complexity.

CT : S-07,08(4M), S-09(7M), S-13(5M), W-08(5M)

OR Write an algorithm for any one method for finding the minimum cost spanning tree. Also discuss its complexity.

CT : S-06(5M)

Ans. Kruskal algorithm :

Kruskal (G, n : T)

//G – Graph of 'n' vertices

//T – Tree

Step 1 : Sort edges on ascending order of weights

Step 2 : Let 'n' = Number of vertices

Step 3 : T = NULL

Step 4 : repeat

(F) ← e = {u, v } ⟶ selected edges

(F) ← x = u comp (u)

y = v comp (v)

if (x ≠ y)

T = T ∪ {e}

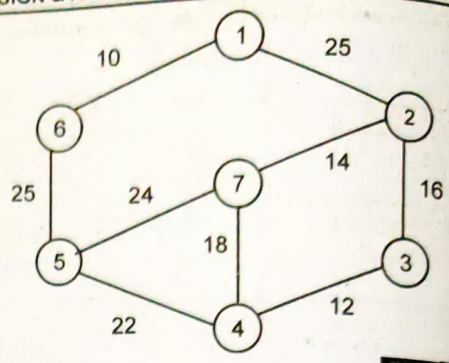
merge (x, y)

initial {T contains n –1 edges}

Efficiency of Kruskal's algorithm (Time and space complexity) :

- Use of priority queue based on weight of edges forming the initial priority queue is O(e log e).
- Removing the minimum-weight arc and adjusting the priority queue O(log e).
- Locating the root of a tree is O(log n). Initial formation of the n trees is O(n).
- Thus, assuming that n < e, as it is true for most graphs, kruskal's algorithms is O(e log e).

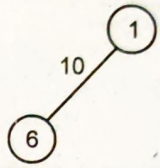
Q.57. Write algorithm to find out minimum cost spanning tree for the following graph, using PRIMS algorithm. What are the differences between PRIMS and KRUSKAL algorithm?



CS : S-11,12(8M)

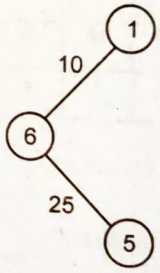
Ans. Minimum edge = 10

Step 1 :



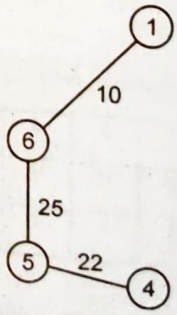
Adj [6] = 5
cost = 25

Step 2 :



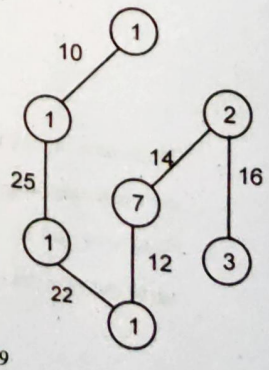
Adj [5] = [4,7]
min = 4

Step 3 :



Adj [4] = [3, 7]
min = 7

Step 4 :



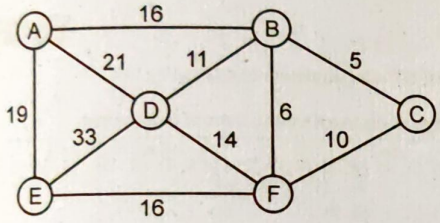
Total cost = 99

Differences :

(1) It is method to find out minimum cost spanning tree for a given graph considering, vertex of a graph as a root hence, it is called as single source minimum cost spanning tree. With this method tree is constructed in natural way by assuming vertices of previously considered edges as a root for new edges to be newly added, viceversa.

(2) The kruskal algorithm is based on generating spanning tree by considering the edges in ascending order of weight.

Q.58. Compute a minimum cost spanning tree for the following graph using Kruskal algorithm and explain its complexity.

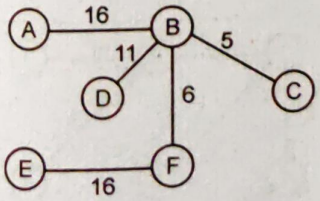


CT : S-10(7M)

Ans. Arrange edges in ascending order of their weight.

{BC} {BF} {CF} {BD} {DF} {EF} {AB} {AE} {AD} {DE}.

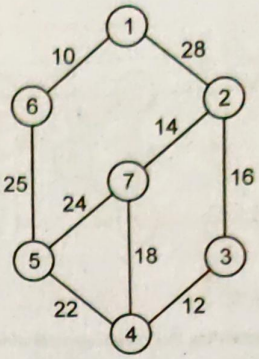
Edge	Cost	Accept Reject	Tree
-	-	-	{A}{B}{C}{D}{E}{F}
BC	5	Accept	{A}{B,C}{D}{E}{F}
BF	6	Accept	{A} {B,C,F} {D}{E}
CF	10	Reject	
BD	11	Accept	{A}{B,C,D,F} {E}
{DF}	14	Reject	
EF	16	Accept	{A}{B,C,D,E,F}
AB	16	Accept	{A,B,C,D,E,F}



Total cost = 5 + 6 + 11 + 16 + 16

= 54.

Q.59. Solve the following using Kruskal's method.



CT : S-13(6M)

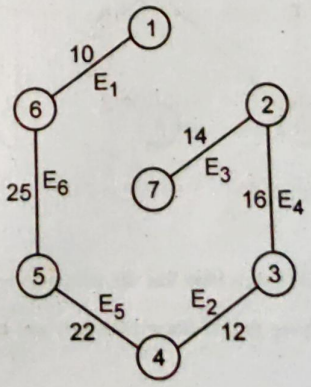
Ans. Arrange the edges in ascending order of their weight.

{1, 6} {3, 4} {2, 7} {2, 3} {4, 7} {5, 4} {5, 7}

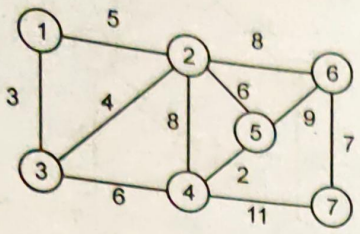
{6, 5} {1, 2}

Edge	Cost	Accepted/ Rejected	Tree
{1, 6}	10	Accepted	{1} {2} {3} {4} {5} {6} {7}
{3, 4}	12	Accepted	{1, 6} {2} {3} {4} {5} {7}
{2, 7}	14	Accepted	{1, 6} {2, 7} {3, 4} {5}
{2, 3}	16	Accepted	{1, 2, 3, 4, 6, 7} {5}
{4, 7}	18	Rejected	{1, 2, 3, 4, 6, 7} {5}
{5, 4}	22	Accepted	{1, 2, 3, 4, 5, 6, 7}
{5, 7}	24	Rejected	{1, 2, 3, 4, 5, 6, 7}
{6, 5}	25	Accepted	{1, 2, 3, 4, 5, 6, 7}
{1, 2}	28	Rejected	{1, 2, 3, 4, 5, 6, 7}

Total Cost = 99



Q.60. Draw spanning tree for following graph using Kruskal's algorithm.

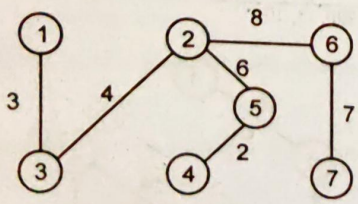


CT : S-14(5M)

Ans. Arrange edges according to Ascending order of weight

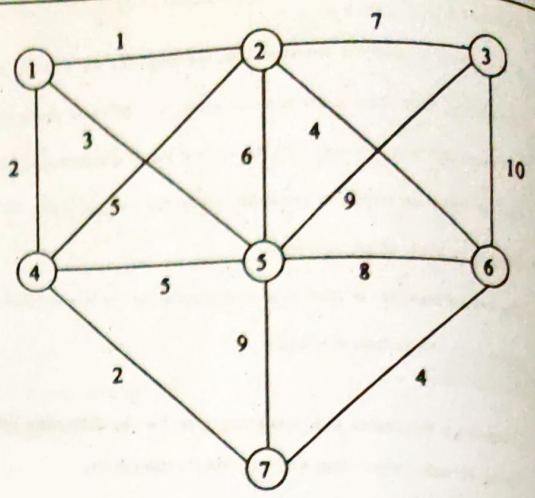
{4, 5} {1, 3} {2, 3} {1, 2} {2, 5} {3, 4} {6, 7}
{2, 4} {2, 6} {5, 6} {4, 7}

Edge	Cost	Accepted/ Rejected	Tree
			{1}{2}{3}{4}{5}{6}{7}
{4, 5}	2	Accept	{1}{2}{3}{4,5}{6}{7}
{1, 3}	3	Accept	{1, 3}{2}{4, 5}{6}{7}
{2, 3}	4	Accept	{1, 2, 3}{4, 5}{6}{7}
{1, 2}	5	Reject	{1, 2, 3, 4, 5}{6}{7}
{2, 5}	6	Accept	{1, 2, 3, 4, 5}{6}{7}
{3, 4}	6	Reject	{1, 2, 3, 4, 5}{6}{7}
{6, 7}	7	Accept	{1, 2, 3, 4, 5}{6, 7}
{2, 4}	8	Reject	{1, 2, 3, 4, 5}{6, 7}
{2, 6}	8	Accept	{1, 2, 3, 4, 5, 6}{7}
{5, 6}	9	Reject	{1, 2, 3, 4, 5, 6}{7}
{4, 7}	11	Reject	{1, 2, 3, 4, 5, 6}{7}



Total cost = 30

Q.61. Using Kruskal's algorithm find the minimum cost spanning tree for the following graph. Show that how you have selected the edges.



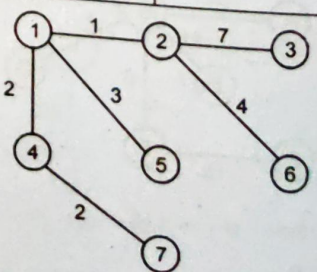
CT : W-06(2M)

Ans. For MCST refer minimum cost spanning tree.

Arrange edges in ascending order of their weight.

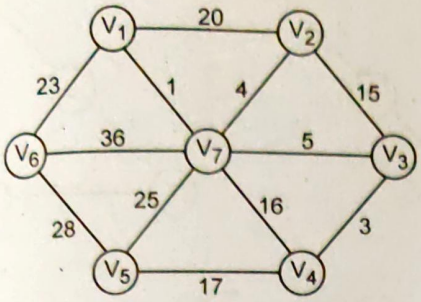
{1, 2} {1, 4} {4, 7} {1,5} {2, 6} {6, 7} {2, 4} {4, 5} {2, 5} {2, 3} {5, 6} {3, 5} {5, 7}

Edge	Cost	Accept Reject	Tree
(1, 2)	1	Accept	{1,2}{3},{4},{5},{6},{7}
(1, 4)	2	Accept	{1,2,4} {3} {5} {6} {7}
(4, 7)	2	Accept	{1,2,4,7}{3}{5}{6}
(1, 5)	3	Accept	{1,2,4,5,7} {3}{5}{6}
(2, 6)	4	Accept	{1,2,4,5,6,7}{3}
(6, 7)	4	Reject	
(2, 4)	5	Reject	
(4, 5)	5	Reject	
(2, 5)	6	Reject	
(2, 3)	7	Accept	{1,2,3,4,5,6,7}
(5, 6)			
(3, 5)			
(5, 7)		Reject	



Total cost = 1 + 2 + 2 + 3 + 4 + 7 = 19

Q.62. Obtain MCST for the given graph.



CT : S-07(4M)

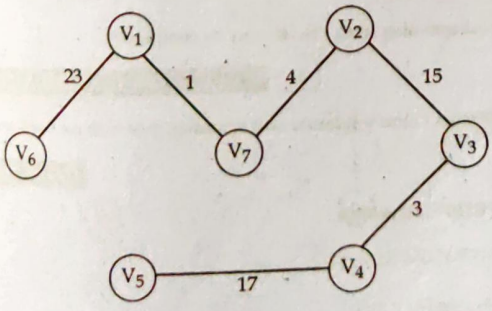
Ans. (1) Ascending order of weight.

$\{V_1, V_7\}\{V_3, V_4\}\{V_2, V_7\}\{V_3, V_7\}\{V_2, V_3\}\{V_4, V_7\}$

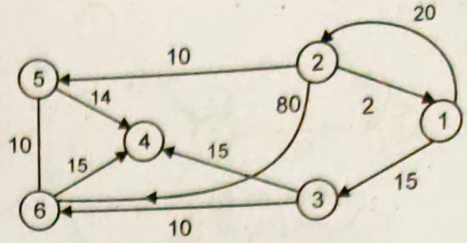
$\{V_4, V_5\}\{V_1, V_2\}\{V_1, V_6\}\{V_5, V_6\}$

Edge	Cost	Accept Reject	Tree
(V_1, V_7)	1	Accept	{1},{2},{3},{4},{5},{6},{7}
(V_3, V_4)	3	Accept	{1, 7},{2},{3},{4},{5},{6}
(V_2, V_7)	4	Accept	{1, 7},{2},{3, 4},{5},{6}
(V_3, V_7)	9	Reject	{1, 2, 7},{3, 4},{5},{6}
(V_2, V_3)	15	Accept	{1, 2, 7},{3, 4},{5},{6}
(V_4, V_7)	16	Reject	{1, 2, 3, 4, 7},{5},{6}
(V_4, V_5)	17	Accept	{1, 2, 3, 4, 7},{5},{6}
(V_1, V_2)	20	Reject	{1, 2, 3, 4, 5, 7},{6}
(V_1, V_6)	23	Accept	{1, 2, 3, 4, 5, 7},{6}
(V_5, V_6)	28	Reject	{1, 2, 3, 4, 5, 6, 7}

Total Cost = 65

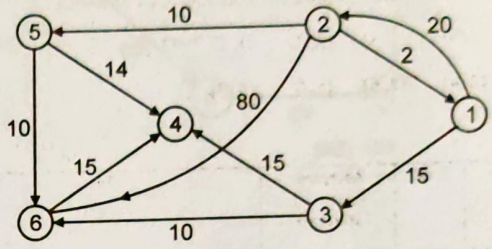


Q.63. Find the minimum cost spanning tree for the following graph.



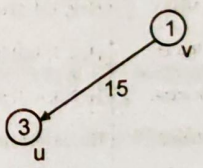
CT : W-10(6M)

Ans.



(1) Start with source vertex 1.

(2) We locate the vertex closet to it i.e. we find vertex from the adjacent vertices of 1 for which the length of the edge is minimum. There are two adjacent vertices of 1 i.e. 2 and 3 but the minimum length is 15.

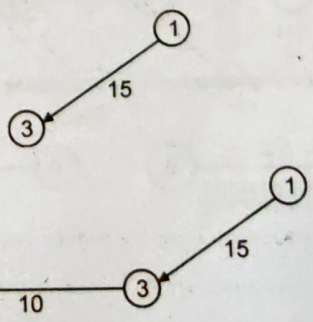


(3) Among the set of all vertices find other vertex V_i that is not included such that (V_i, V_j) is minimum and add it to V.

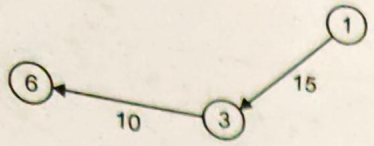
min (12, 34, 36)

$\Rightarrow$ min (20, 15, 10)

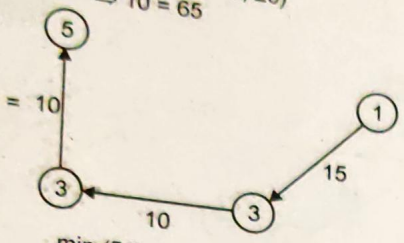
$\Rightarrow$ 10 = 36



(4) Continue till we get MST

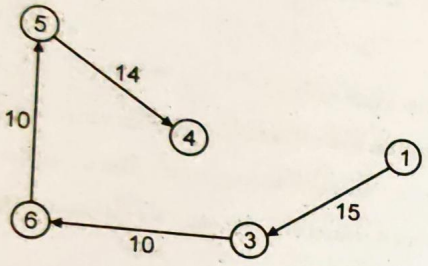


min (64, 65,12)
⇒ min (15, 10, 20)
⇒ 10 = 65



min (54)
⇒ 14
⇒ 54

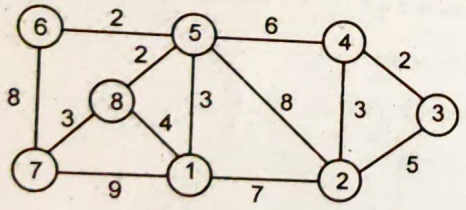
(5)



So, minimum cost W (T) = $\sum_{u,V \in T} W(T)$

= 10 + 10 + 15 + 14 = 49

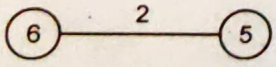
Q.64. What is the use of spanning tree? Draw spanning tree for following graph using PRIMS method.



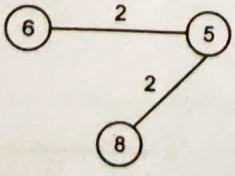
CS : W-11(5M)

Ans. The use of spanning tree is for broadcasting.

Step 1 :

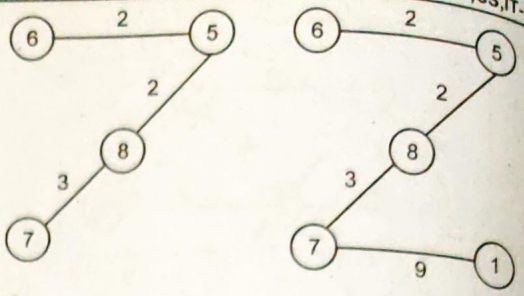


Step 2 :

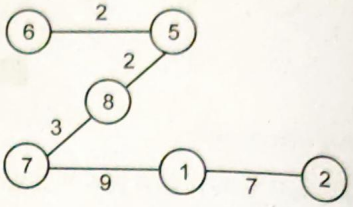


Step 3 :

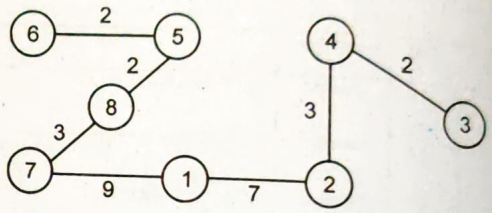
Step 4 :



Step 5 :

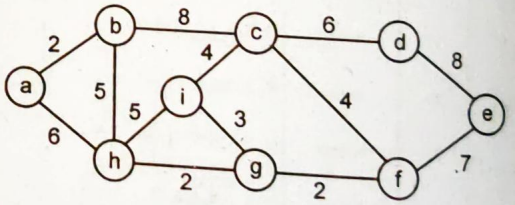


Step 6 :



Total cost = 28.

Q.65. Write PRIM's algorithm to generate spanning tree. Implement the algorithm on following graph. Explain the complexity.



CS : W-12(4M)

Explain the steps in execution. *CS : W-10(8M), W-13(3M)*

OR **Write an algorithm for prim's method for finding the minimum cost spanning tree. Also discuss its complexity.**

CS : S-14(5M), W-11(4M), S-13(3M)

OR **Discuss Prims minimum cost spanning tree with an example.**

CT : W-06(5M)

Ans. **PRIM'S algorithm :**

MST-PRIM (G,w,r)

Step 1 : For each u ∈ v [G]

Step 2 : do key [u] $\leftarrow \infty$

Step 3 : π [u] $\leftarrow$ NIL

Step 4 : key [r] $\leftarrow 0$

Step 5 : Q $\leftarrow$ v [G]

Step 6 : while Q $\neq \phi$

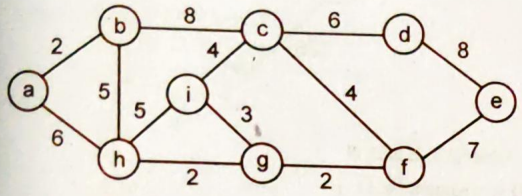
Step 7 : do u $\leftarrow$ EXTRACT-MIN (Q)

Step 8 : for each v $\in$ Adj [u]

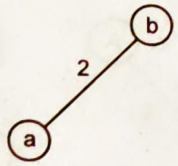
Step 9 : do if v $\in$ Q and W(u,v) < key[v]

Step 10 : then π [v] $\rightarrow$ u

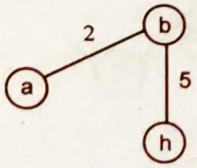
Step 11 : key [v] $\leftarrow$ W [u, v]



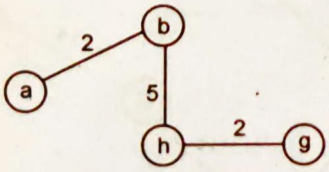
Step 1 :



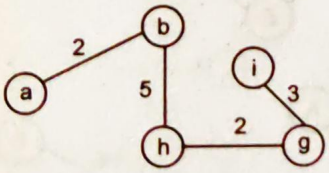
Step 2 :



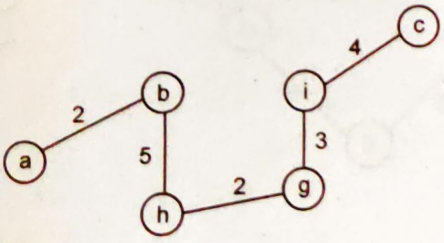
Step 3 :



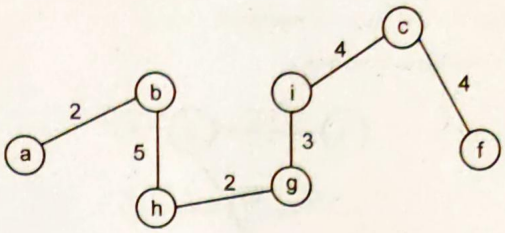
Step 4 :



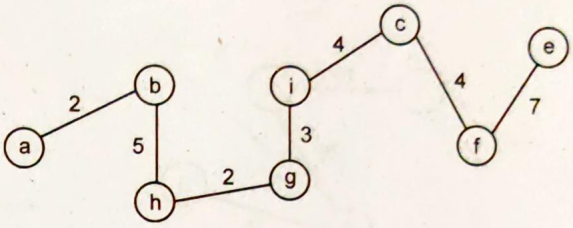
Step 5 :



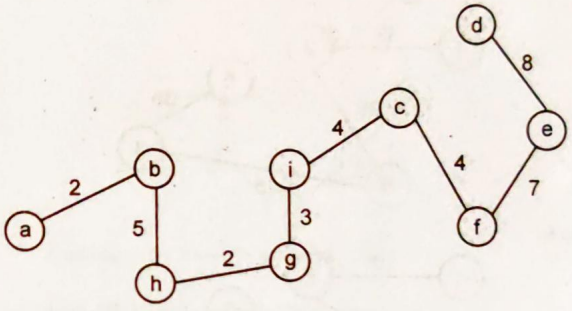
Step 6 :



Step 7 :

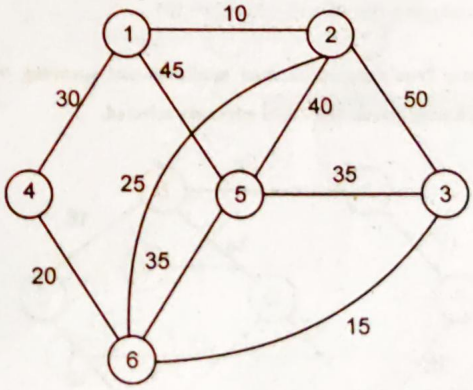


Step 8 :



Total cost = 2 + 5 + 2 + 3 + 4 + 4 + 7 + 8 = 35

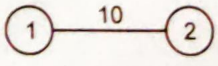
Q.66. Find minimum cost spanning tree for the following graph using Prim's algorithm.



CS : S-10(7M)

Ans. Minimum distance = 10 :

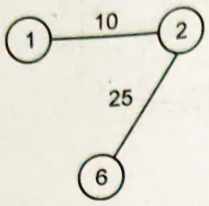
Step 1 :



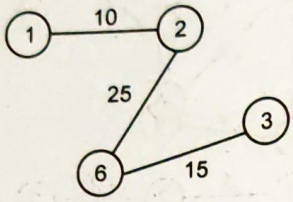
Step 2 : Adjacent of vertex 2 is 3, 5, 6

Edge 23 = 50 & edge 25 = 4, Edge 25 = 25

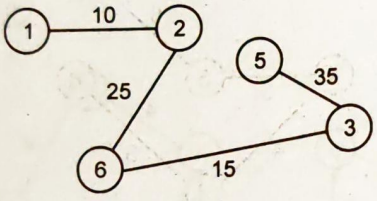
∴ Next Edge is 26 and so on



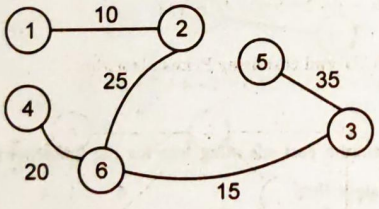
Step 3 :



Step 4 :

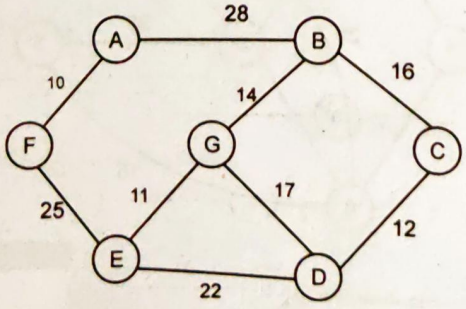


Step 5 :



Total cost = 10 + 25 + 15 + 35 + 20 = 105

Q.67. Using Prim's algorithm, find minimum-cost spanning tree for following graph. Show how edges are selected.

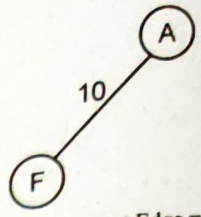


CT : S-1106MJ

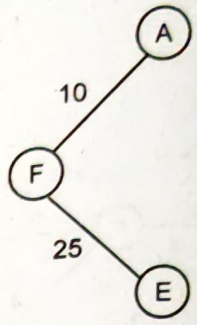
Ans. Minimum weight = 10

∴ Edge AF is selected.

Step 1 :



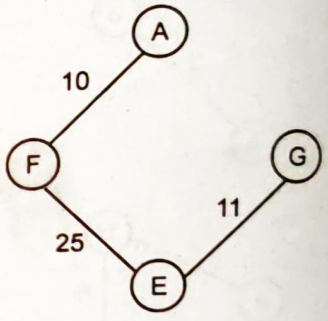
Step 2 : Adjacent of F is E i.e. 25 so next Edge = FE



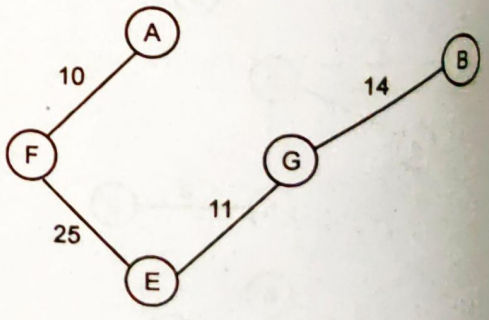
Step 3 : Adjacent to E is D, G

G is minimum i.e. 11

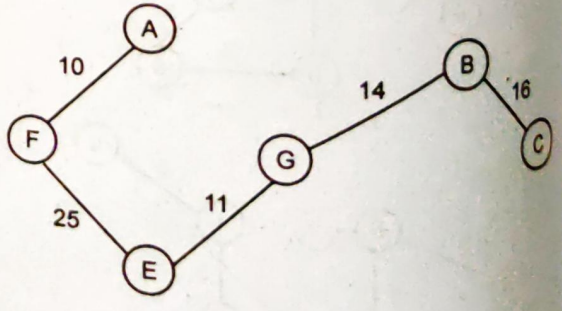
so next Edge = EG



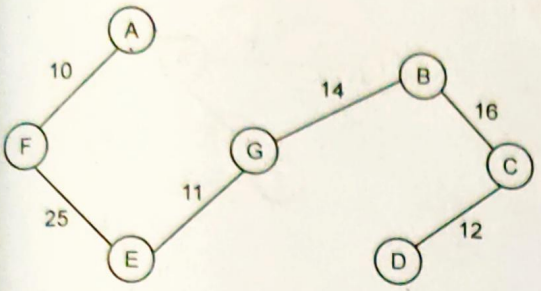
Step 4 :



Step 5 :

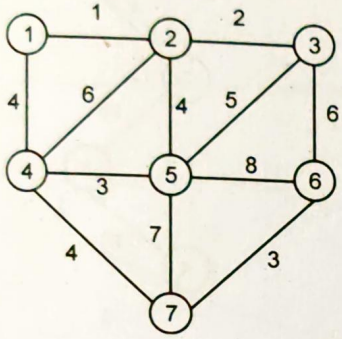


Step 6 :



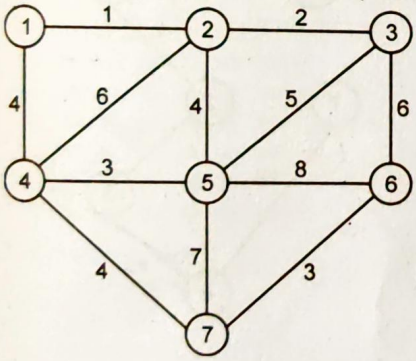
Total cost = 10 + 25 + 11 + 14 + 16 + 12 = 88.

Q.68. Show the snapshots of Prims algorithm to find minimum cost of spanning tree for the given graph.

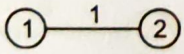


CT : W-12(6M)

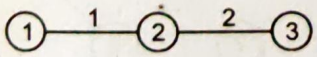
Ans. A minimum cost spanning tree or minimum weight spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree.



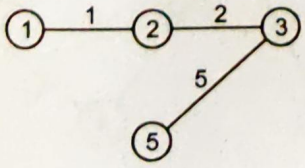
Step 1 :



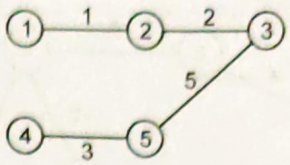
Step 2 :



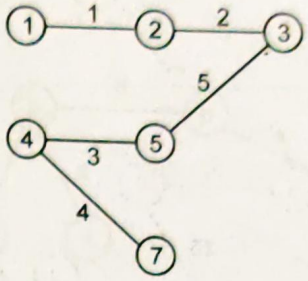
Step 3 :



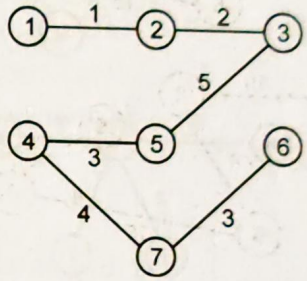
Step 4 :



Step 5 :

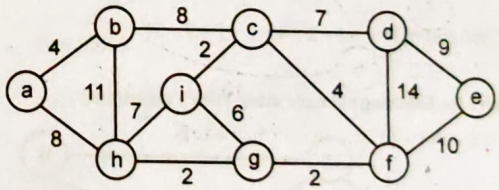


Step 6 :



Total cost = 1 + 2 + 5+ 3+ 4+ 3 = 18

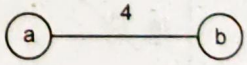
Q.69. Obtain MCST and cost using Prims algorithms.



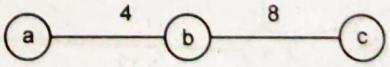
CT : W-13(7M), CS : W-13(4M)

Ans.

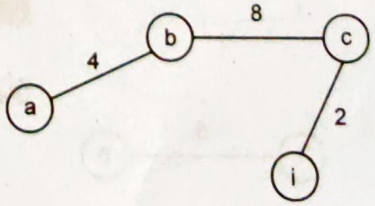
Step 1 :



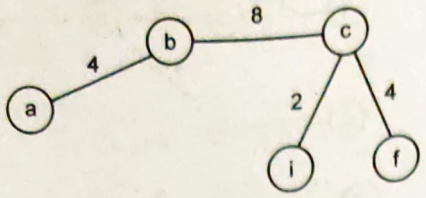
Step 2 :



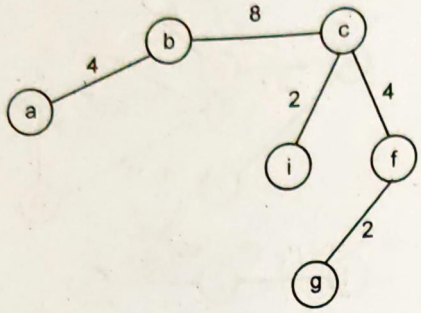
Step 3 :



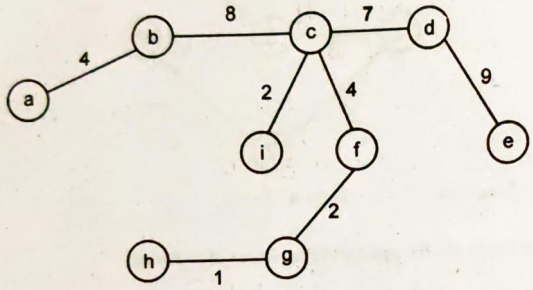
Step 4 :



Step 5 :

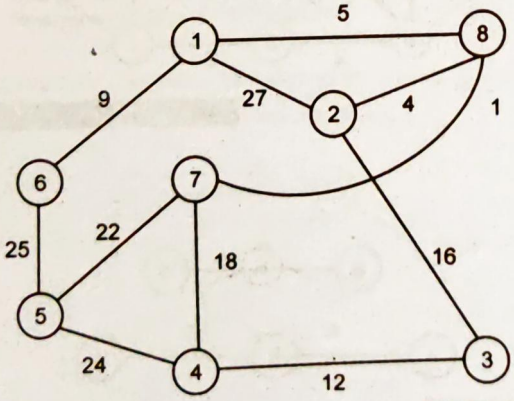


Step 6 :



Total cost = 1+ 4 + 8 + 2 + 4 + 2 + 7 + 9 = 37

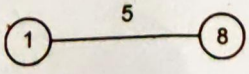
Q.70. Solve the following example using Prim's algorithm.



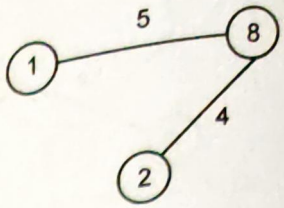
CS : S-14(5M)

Ans.

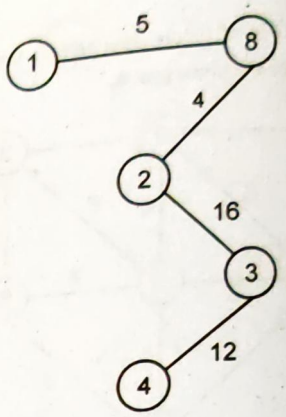
Step 1 :



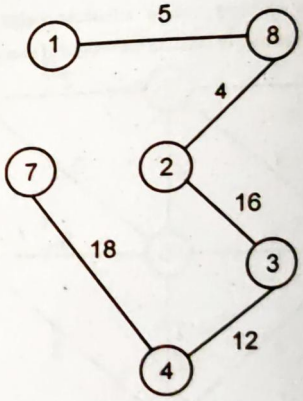
Step 2 :



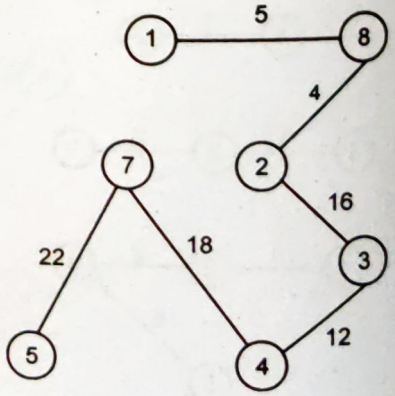
Step 3 :



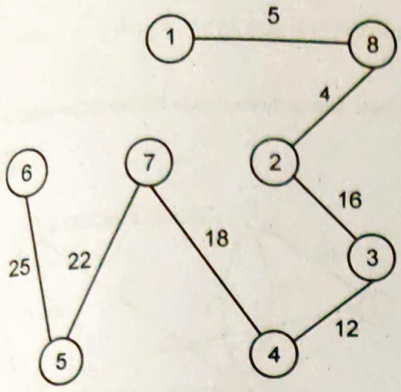
Step 4 :



Step 5 :

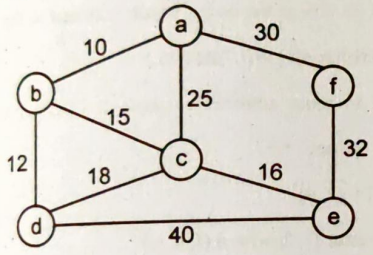


Step 6 :



Total cost = 5 + 4 + 16 + 12 + 18 + 22 + 25
= 102.

Q.71. **Design cost matrix, near array and output matrix. Write Prim's algorithm.**



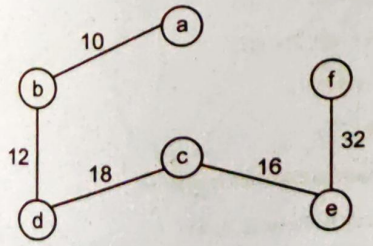
CS : W-12(4M)

Ans.

$$\text{cost} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 99 & 10 & 25 & 99 & 99 & 30 \\ 10 & 99 & 15 & 12 & 99 & 99 \\ 25 & 15 & 99 & 18 & 16 & 99 \\ 99 & 12 & 18 & 99 & 40 & 99 \\ 99 & 99 & 16 & 40 & 99 & 32 \\ 30 & 99 & 99 & 99 & 32 & 99 \end{bmatrix} \end{array}$$

near = [0 0 2 1 2 2] value
a b c d e f index

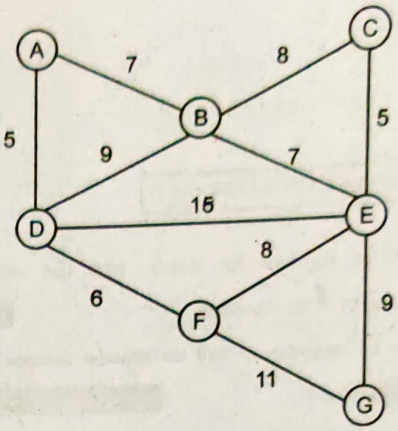
$$\text{output} = \begin{bmatrix} a & b & 10 \\ b & d & 12 \\ d & c & 18 \\ c & e & 16 \\ e & f & 32 \end{bmatrix}$$



Total cost = 10 + 12 + 18 + 16 + 32 = 88

Q.72. **Implement Prims algorithm on the following graph. Find out the content of near array at each step of processing.**

CS : S-13(4M)



Ans. Mincost = cost [K, L] = 5 i.e. AD.

K = A, L = D.

Cost [A, D], t [A, A] = k;

Cost [A, A] < cost [D, A]

Near [i] = L i.e.

Near [A] = D

$$\text{Near} = \begin{bmatrix} D & & & & & & \\ A & B & C & D & E & F & G \end{bmatrix}$$

Near [B] = A

$$\text{Near} = \begin{bmatrix} D & A & & & & & \\ A & B & C & D & E & F & G \end{bmatrix}$$

Near [C] = E

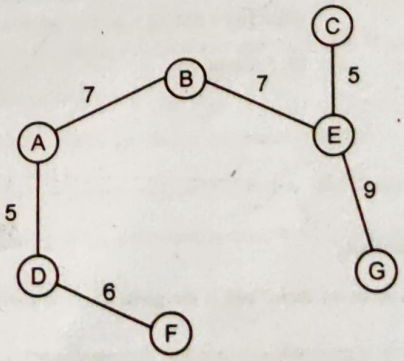
$$\text{Near} = \begin{bmatrix} D & A & E & F & F & E & E \\ A & B & C & D & E & F & G \end{bmatrix}$$

Near [D] = F

Near [E] = F

Near [F] = D

Near [G] = E



$$t = \begin{bmatrix} A & D & 5 \\ B & A & 7 \\ C & E & 5 \\ D & F & 6 \\ E & F & 8 \\ F & E & 8 \end{bmatrix}$$

$t \Rightarrow 7 + 5 + 7 + 8 + 5 + 8 + 9 = 39$

SINGLE SOURCE SHORTEST PATH

Q.73. Write an algorithm for shortest path tree and explain the complexity of the algorithm. **CT: S-11(4M)**

OR Give an algorithm to find the shortest distance using greedy approach. **CT : W-09(6M), CS : W-10(3M)**

Ans. **Algorithm :** Single source shortest path.

Algorithm SSSP (cost, n, s : dist)

(Assume v = source vertex)

Step 1 : For i = 1 to n do

{

S [i] = 0

dist [i] = cost [v, i]

}

Step 2 : S [v] = i

dist [v] = 0

Step 3 : for i = 2 to n do.

{

[Determine (n – 1) from vertex "v" and let the vertices be denoted as "u" from the vertices "u" select a vertex such that S(u) ≠ 1 and dist [u] = min]

for (each "w" adjacent to vertex "u")

{

if dist [w] > dist [u] + cost [u, w] then dist [w] = dist [u] + cost [u, w]

}

}

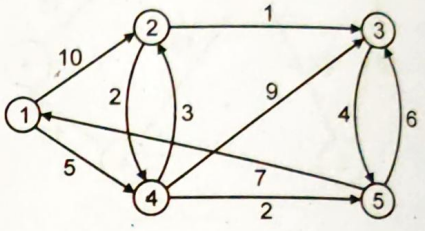
}

Complexity :

It depends on step 2 and if the given graph is complete then it is $O(n^2)$ if the graph is incomplete the complexity is O (n + e).

As in the above problem cost from edge s z is not given and both the direction from edge v y are same so solution is not possible.

Q.74. Find single source shortest path for the following diagram. **CT : S-11(4M)**



Ans.

Step 1 : Given initial graph G = (V, E). All nodes have infinite cost except the source node S, which has 0 cost.

Step 2 : First we choose the node , which is closest to the source node, 1.

We initialize d[1] to 0. Add it to 1.

Relax all nodes adjacent to source 1. Update predecessor for all nodes updates.

Adj[1] = {2, 4}

Apply relax (1, 2, w) and (1, 4, w)

Step 3 : Choose the closest node 4. Relax all nodes adjacent to node 4.

Update predecessors for nodes 2, 3 and 5.

Adj [4] = {2, 3, 5}

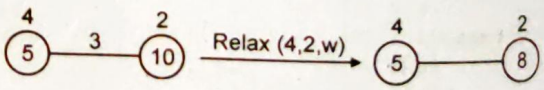
apply Relax (4, 2, w), Relax (4, 3, w), Relax (4, 5, w)

Relax (4, 2, w)

d[4] = 5

d[2] = 10

w(4, 2) = 3



If d[2] + w[2, 3] < d[3]

If d[4] + w[4, 2] < d[2]

If 5 + 3 < 10

8 < 10 True.

i.e. condition examined is satisfied.

Similarly for Relax (2, 5, w)

d[2] = 5

d[5] = ∞

w[2, 5] = 2

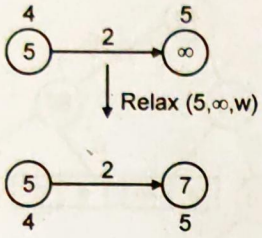
If d[2] + w[2, 5] < d[5]

If 5 + 2 < ∞

7 < ∞

i.e. condition examined is satisfied.

Since 7 is less than infinity.



Again Relax (4, 3, w)

d[4] = 5

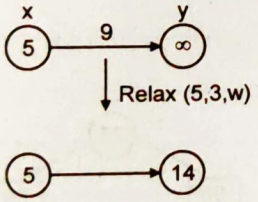
d[3] = ∞

w[4, 3] = 9

If (5 + 9) < ∞

14 < ∞

Relax (4, 3, w)

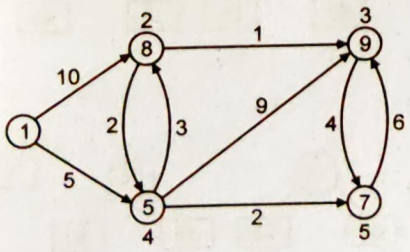


Finally

If d[2] + w [2, 3] < d [3]

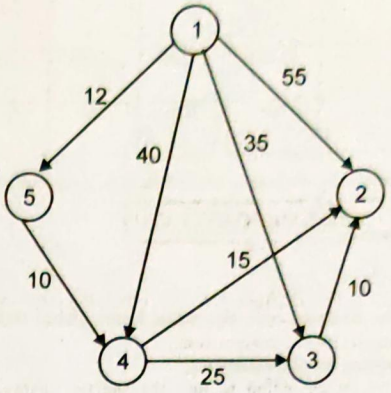
If 8 + 1 < 13

9 < 13



Thus, we find that dsv = 9

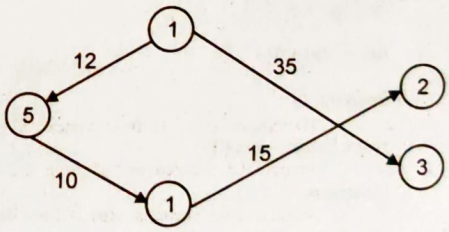
Q.75. Implement the single source shortest path algorithm on following graph and explain execution.



CS : W-10(3M)

Ans. We can implement single source shortest path algorithm on following graph.

Steps	Cost	
1	1	-
2	1, 5	12
3	1, 5, 4	12 + 10 = 22
4	1, 5, 4, 2	22 + 15 = 37
5	1, 3	37 + 15 = 32



HUFFMAN CODE

Q.76. Explain the concept of Huffman code with suitable example.

Ans. Huffman code :

- Huffman code is a technique for compressing data.
- Huffman's greedy algorithm looks at the occurrence of each character and it acts as a binary string in an optimal way.

(a) Fixed length code.

(b) Prefix code.

- In fixed length code it needs 3 bits to represent 6 characters.

	a	b	c	d	e
Frequency	45,000	13,000	12,000	16,000	9,000
Fixed length code	000	001	010	011	100

- In prefix codes the code in which there is no codeword is a prefix of other codeword.

Q.77. Write Huffman code algorithm. Find optimal Huffman code for following set of frequencies.

a : 20, b : 15, c : 5, d : 25, e : 35 **CT : W-13(5M)**

Ans. Algorithm :

HUFFMAN (C)

Step 1 : n ← |C|

Step 2 : Q ← C

Step 3 : for i ← 1 to n – 1

Step 4 : z ← Allocate-Node ( )

Step 5 : x ← Extract-min (Q)

Step 6 : y ← Extract-min (Q)

Step 7 : Left [z] ← x

Step 8 : right [z] ← y

Step 9 : f(z) ← f(x) + f(y)

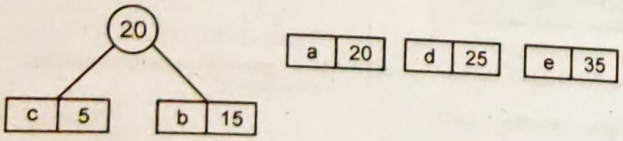
Step 10 : Insert (Q, Z)

Step 11 : return Extract - min (Q)

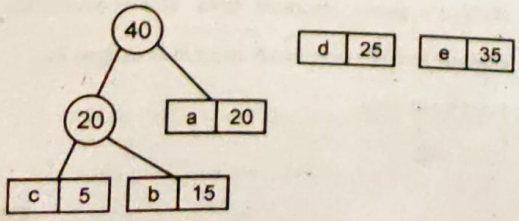
Given that,

C = {a, b, c, d, e}

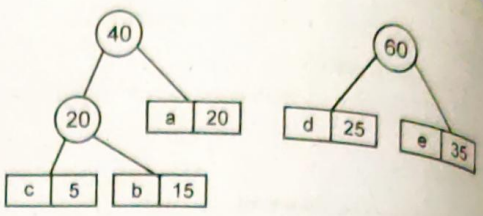
Step 1 :



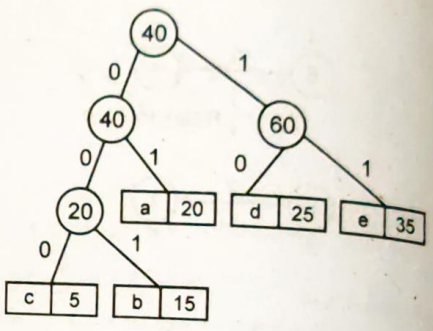
Step 2 :



Step 3 :



Step 4 :



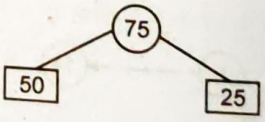
Prefix code : a = 01, b = 0.01, c = 000, d = 01, e = 11

Q.78. What are the optimal Huffman codes for following set of frequencies and discuss its complexity?

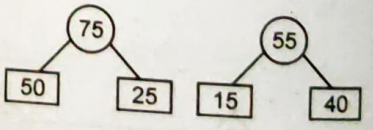
a : 50, b : 25, c : 15, d : 40, e : 75 **CT : W-12(6M)**

Ans.

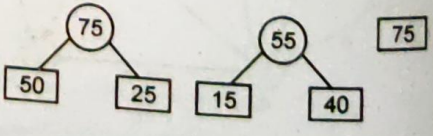
Step 1 :



Step 2 :

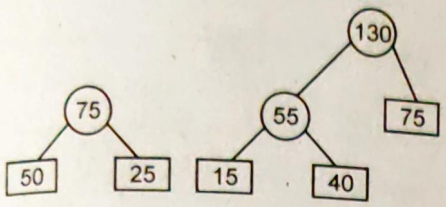


Step 3 :

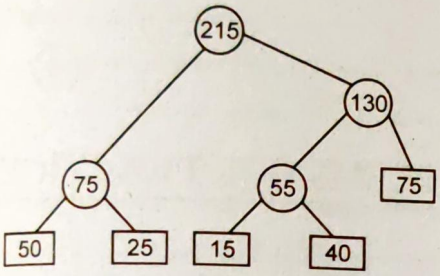


VBD

Step 4 :



Step 5:



Q.79. Generate a Huffman code for the following characters given below :

Symbol	Frequency
a	45
b	13
c	12
d	16
e	9
f	5

CS : S-12(6M)

Ans. Given that

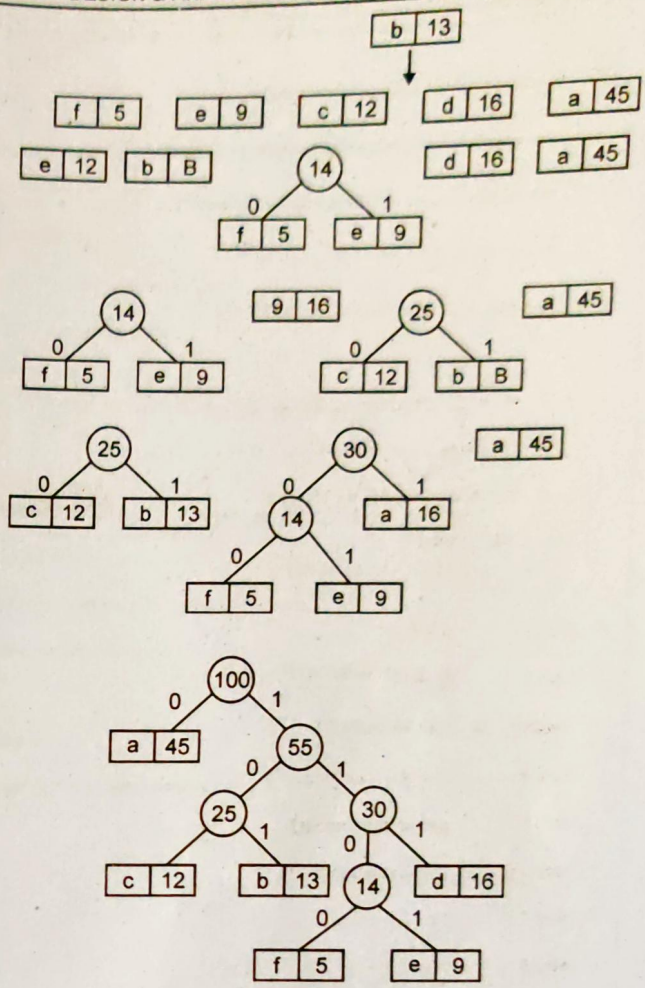
C = {a, b, c, d, e, f}

f(C) = {45, 13, 12, 16, 9, 5}

n = 6 by line 1 as n ← |C|

n ← 6

Q ← C



Prefix code :

a = 0 d = 1 1 1

b = 1 0 1 e = 1 1 0 1

c = 1 0 0 f = 1 1 0 0

OPTIMAL MERGE PATTERN

Q.80. What is optimal merge pattern? Implement the merging on following files with sizes as specified.

28, 32, 12, 5, 84, 53, 91, 35, 3, 11 **CS : S-11, W-11(4M)**

Ans. Optimal merge pattern :

- Let X_1 and X_2 be two sorted files of size 'n' and 'm' respectively.
- If it is required to merge these files the total record movement will be m + n.

- If there exist more than two files and it is required to merge these files then the concept of optimal merge pattern is used.
- This concept is based on minimum movement of records.
- The merging is based on selecting the files of lowest size.

Given : 28, 32, 12, 5, 84, 53, 91, 35, 3, 11

Sort = [3, 5, 11, 12, 28, 32, 35, 53, 84, 91]

\ /

[8, 11, 12, 28, 32, 35, 53, 84, 91]

\ /

[19, 12, 28, 32, 35, 53, 84, 91]

Sort [12 19 28 32 35 53 84 91]

\ /

31 28 32 35 53 84 91

Sort [28 31 32 35 53 84 91

\ /

[59 32 35 53 84 91]

Sort [32 35 53 59 84 91]

\ /

[67 53 59 84 91]

Sort [53 59 67 84 91]

\ /

[112 67 84 91]

Sort [67 84 91 112]

\ /

[151 91 112]

Sort [91 112 151]

\ /

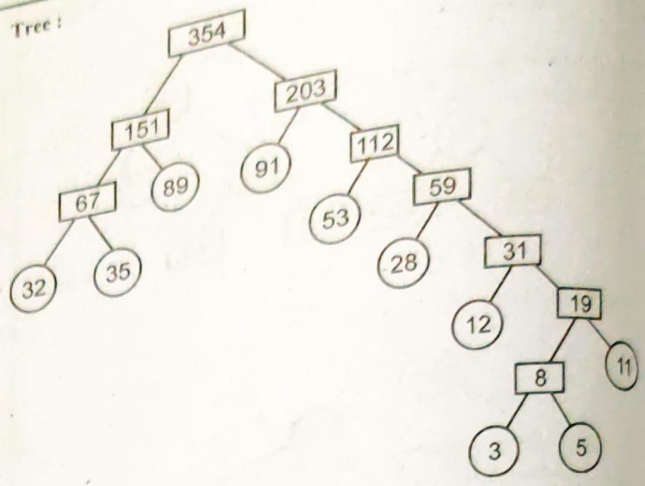
[203 151]

Sort [151 203]

\ /

[354]

Tree :



PROBLEMS FOR PRACTICE :

Q.1. Use divide and conquer technique to sort following list of numbers.

A = 10, 3, 25, 13, 1, 28, 11, 14, 24

Q.2. Perform binary search on following set of data to search element 125.

– 15, – 6, 0, 7, 9, 23, 54, 82, 101, 112, 125, 131

Q.3. Implement quick sort on following array

5, 8, 7, 3, 4, 9, 10

Q.4. Sort the following array using merge sort.

10, 25, 40, 5, 20, 1, 100, 60, 30, 50

Q.5. Sort the following data using heap sort

88, 12, 91, 23, 10, 36, 45, 55, 15, 39, 81

Q.6. Draw the tree structure of recursive call of max min for given array :

22, 13, – 5, – 8, 15, 60, 17, 31, 47

Q.7. Find optimal merge pattern for following data :

28, 32, 12, 5, 84, 53, 91, 35, 3, 11

POINTS TO REMEMBER :

(1) Divide and conquer strategy breaks the problems into several subproblems that are similar to original problem but smaller in size, solve the subproblems recursively and then combine these solutions.

(2) Binary search uses the method of divide and conquer to search the elements from array of size n.

(3) Quick sort is a sorting algorithm that uses divide and conquer strategy where division is dynamically carried out.

(4) Merge sort is a sorting technique which divides the array into subarrays of size 2 & merge adjacent pairs.

(5) Heap sort is a sorting algorithm which works in two stages :

(i) Heap construction.

(ii) Deletion of maximum key.

(6) Matrix operation is perfomed by using Starassen's matrix operation algorithm.

(7) MinMax is a recursive algorithm that finds the maximum and minimum of the set of elements.

(8) Greedy method suggests that one can devise an algorithm that works in stages, considering one input at a time.

(9) An activity selection is a problem of scheduling a resource among several competing activity.

(10) Knapsack is a bag of fioced capacity of storing certain object.

(11) In job sequencing there are various jobs to be processed using one processor.

(12) Minimum spanning tree is one whose weight is less than or equal to weight of every other spanning tree.

(13) Minimum spanning tree used 2 methods :

(i) Kruskal's method.

(ii) Prims method.

(14) Huffman's code is a technique used for compressing data.

(15) Optimal merge pattern is a pattern that relates to the merging of two or more sorted files in a sorted file.