UNIT-III

Greedy Method :- The General Method, Knapsack problem, Job Sequencing with deadlines, Minimum-cost Spanning Trees, Prim's Algorithm, Kruskal's Algorithm, optimal merge patterns, Single source shortest Paths.

Greedy Method (General Method) :-

- The Greedy method is one of the approach for solving a problem.
- Greedy method is also known as optimization problem.
- optimization problem is nothing but which requires minimum (or) maximum result.
- optimization problem gives a solution it is called optimal solution.
- In the greedy method we can find out number of solutions, from these solutions we can select only feasible solutions.
- The Greedy method is also known as selection procedure problem.

Algorithm :-

```
Algorithm Greedy (a,n)
{
    Solution = 0 ;
    for i = 1 to n do
    {
        x = Solution(a);
        if feasible (Solution, x) then
            Solution = union (Solution, x);
    }
    return solution;
}
```

Knapsack problem :-

- → Knapsack is nothing but BAG.
- → Knapsack problem is also known as container loading problem.
- → Knapsack problem is also known as fractional Knapsack problem.
- → 0 (zero) represents item is not considerable
  1 (one) represents item is considerable.
- → the Knapsack problem follows :-

The objective of the algorithm is to

$$\boxed{\text{Maximize} \sum_{1 \le i \le n} p_i x_i \text{ subject to}^{\text{constraint}} \sum_{1 \le i \le n} w_i x_i \le m}$$

Here x_i is the fraction of item

$$\& \; 0 \le x_i \le 1$$

Find optimal solution for the following knapsack problem $n = 7$; $m = 15$; profits $(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 5, 15, 7, 6, 18, 3)$ weights $(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (4, 3, 6, 6, 2, 5, 1)$.

Solu) Given that no. of objects $n = 7$

For each item profits are $P_1 = 10$; $P_2 = 5$; $P_3 = 15$; $P_4 = 7$; $P_5 = 6$; $P_6 = 18$; $P_7 = 3$.

For each object weights are $w_1 = 4$; $w_2 = 3$; $w_3 = 6$; $w_4 = 6$; $w_5 = 2$; $w_6 = 5$; $w_7 = 1$

Consider $\frac{\text{profit}}{\text{weight}}$ ratio $\frac{P_i}{w_i}$

$\frac{P_1}{w_1} = \frac{10}{4} = 2.5$; $\frac{P_2}{w_2} = \frac{5}{3} = 1.6$; $\frac{P_3}{w_3} = \frac{15}{6} = 2.5$

$\frac{P_4}{w_4} = \frac{7}{6} = 1.1$; $\frac{P_5}{w_5} = \frac{6}{2} = 3$; $\frac{P_6}{w_6} = \frac{18}{5} = 3.6$

$\frac{P_7}{w_7} = \frac{3}{1} = 3$.

objects O_i	1	2	3	4	5	6	7
profits P_i	10	5	15	7	6	18	3
weights W_i	4	3	6	6	2	5	1
P_i/w_i	2.5	1.6	2.5	1.1	3	3.6	3

Now the solutions are $x_1, x_2, x_3, x_4, x_5, x_6, x_7$

where $0 \le x_i \le 1$

The bag capacity $m = 15$

According to $\frac{\text{profit}}{\text{weight}}$ ratio method we consider maximum element first

Now add object 6 ; 5 Kgs to the Knapsack,

Hence $\boxed{x_6 = 1}$ & $m = 15 - 5 = 10$

Now add object 5 ; 2 Kgs to the Knapsack,

Hence $\boxed{x_5 = 1}$ & $m = 10 - 2 = 8$

Now add object 7 ; 1 Kg to the Knapsack,

Hence $\boxed{x_7 = 1}$ & $m = 8 - 1 = 7$

Now add object 1 ; 4 Kgs to the Knapsack,

Hence $\boxed{x_1 = 1}$ & $m = 7 - 4 = 3$

Now add object 3 ; 6 Kgs to the Knapsack,

Hence $\boxed{x_3 = 3/6 = 1/2}$ and $m = 3 - 3 = 0$

we can not add object 2 and 4 because the bag Contains 15 Kgs

Hence $\boxed{x_2 = x_4 = 0}$

Now
$$\sum w_i x_i = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7$$
$$= 4\times1 + 3\times0 + 6\times1/2 + 6\times0 + 2\times1 + 5\times1 + 1\times1$$
$$= 4 + 0 + 3 + 0 + 2 + 5 + 1$$
$$= 15$$

Hence $\sum w_i x_i \le m$

Now
$$\sum p_i x_i = p_1x_1 + p_2x_2 + p_3x_3 + p_4x_4 + p_5x_5 + p_6x_6 + p_7x_7$$
$$= 10\times1 + 5\times0 + 15\times1/2 + 7\times0 + 6\times1 + 18\times1 + 3\times1$$
$$= 10 + 0 + 7.5 + 0 + 6 + 18 + 3$$
$$= 44.5$$

Hence the optimal solution is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (1, 0, 1/2, 0, 1, 1, 1)$ and Maximum profit is 44.5

Scanned with OKEN Scanner

Find maximum optimal solution for $n = 3$ & $m = 20$
profits $(P_1, P_2, P_3) = (25, 24, 15)$ weights $(w_1, w_2, w_3) = (18, 15, 10)$

Soln: Given that no. of objects $n = 3$

For each item profits are $P_1 = 25$; $P_2 = 24$; $P_3 = 15$.

For each object weights are $w_1 = 18$; $w_2 = 15$; $w_3 = 10$.

Consider $\frac{\text{profit}}{\text{weight}}$ ratio $\frac{P_i}{w_i}$

$\frac{P_1}{w_1} = \frac{25}{18} = 1.38$; $\frac{P_2}{w_2} = \frac{24}{15} = 1.6$; $\frac{P_3}{w_3} = \frac{15}{10} = 1.5$

objects O_i	1	2	3
Profits P_i	25	24	15
weights w_i	18	15	10
$\frac{\text{profits}}{\text{weights}} = \frac{P_i}{w_i}$	1.38	1.6	1.5

Now the solutions are x_1, x_2, x_3 where $0 \le x_i \le 1$

The Bag capacity $m = 20$

According to $\frac{\text{profit}}{\text{weight}}$ ratio method we consider maximum element first

Now add object 2; 15 kg's to the knapsack,

Hence $\boxed{x_2 = 1}$ & $m = 20 - 15 = 5$

Now add object 3; 10 kg's to the knapsack

Hence $\boxed{x_3 = 5/10 = 1/2}$ and $m = 5 - 5 = 0$.

We can not add object 1 because the bag contains 20 kg's

Hence $\boxed{x_1 = 0}$

Now $\sum w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3$

$= 18 \times 0 + 15 \times 1 + 10 \times \frac{1}{2}$

$= 0 + 15 + 5$

$= 20$

Hence $\sum w_i x_i \le m$

Now $\sum p_i x_i = p_1x_1 + p_2x_2 + p_3x_3$

$= 25 \times 0 + 24 \times 1 + 15 \times \frac{1}{2}$

$= 0 + 24 + 7.5$

$= 31.5$

Hence the optimal solution is $(x_1, x_2, x_3) = (0, 1, 1/2)$ and the maximum profit is 31.5

(3) Find out optimal solution for the knapsack problem $n = 7$; $m = 15$; profits: 10, 5, 15, 7, 6, 18, 3. weights: 2, 3, 5, 7, 1, 4, 1.

Solⁿ Given that no. of objects $n = 7$

For each item profits are $p_1 = 10$; $p_2 = 5$; $p_3 = 15$; $p_4 = 7$; $p_5 = 6$; $p_6 = 18$; $p_7 = 3$.

For each object weights are $w_1 = 2$; $w_2 = 3$; $w_3 = 5$; $w_4 = 7$; $w_5 = 1$; $w_6 = 4$; $w_7 = 1$

Consider $\frac{\text{profit}}{\text{weight}}$ ratio $\frac{p_i}{w_i}$

$\frac{p_1}{w_1} = \frac{10}{2} = 5$; $\frac{p_2}{w_2} = \frac{5}{3} = 1.6$; $\frac{p_3}{w_3} = \frac{15}{5} = 3$

$\frac{p_4}{w_4} = \frac{7}{7} = 1$; $\frac{p_5}{w_5} = \frac{6}{1} = 6$; $\frac{p_6}{w_6} = \frac{18}{4} = 4.5$

$\frac{p_7}{w_7} = \frac{3}{1} = 3$

objects o_i	1	2	3	4	5	6	7
profits p_i	10	5	15	7	6	18	3
weights w_i	2	3	5	7	1	4	1
p_i/w_i	5	1.6	3	1	6	4.5	3

Now the solutions are $x_1, x_2, x_3, x_4, x_5, x_6, x_7$

where $0 \le x_i \le 1$

the bag capacity $m = 15$

According to $\frac{\text{profit}}{\text{weight}}$ ratio method, we consider

maximum element first

now add object 5 ; 1 kg to the knapsack

Hence $\boxed{x_5 = 1}$ & $m = 15 - 1 = 14$

now add object 1 ; 2 kgs to the knapsack

Hence $\boxed{x_1 = 1}$ & $m = 14 - 2 = 12$

now add object 6 ; 4 kgs to the knapsack

Hence $\boxed{x_6 = 1}$ & $m = 12 - 4 = 8$

now add object 3 ; 5 kgs to the knapsack

Hence $\boxed{x_3 = 1}$ & $m = 8 - 5 = 3$

now add object 7 ; 1 kg to the knapsack

Hence $\boxed{x_7 = 1}$ & $m = 3 - 1 = 2$

now add object 2 ; 3 kg to the knapsack

Hence $\boxed{x_2 = 2/3}$ & $m = 2 - 2 = 0$.

we can not add object 4 because the bag containing 15 kgs

Hence $\boxed{x_4 = 0}$

now $\sum w_i x_i = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7$

$= 2\times1 + 3\times\frac{2}{3} + 5\times1 + 7\times0 + 1\times1 + 4\times1 + 1\times1$

$= 2 + 2 + 5 + 0 + 1 + 4 + 1$

$= 15$

Hence $\sum w_i x_i \le m$

now $\sum p_i x_i = p_1x_1 + p_2x_2 + p_3x_3 + p_4x_4 + p_5x_5 + p_6x_6 + p_7x_7$

$= 10\times1 + 5\times\frac{2}{3} + 15\times1 + 7\times0 + 6\times1 + 18\times1 + 3\times1$

$= 10 + 3.33 + 15 + 0 + 6 + 18 + 3$

$= 55.33$

Hence the optimal solution is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (1, 2/3, 1, 0, 1, 1, 1)$ and the maximum profit is 55.33.

Algorithm :-

```
// p[1:n] & w[1:n] contain profits & weights
// n objects ordered such that p[i]/w[i] ≥ p[i+1]/w[i+1]
// m is the knapsack bag size and x[1:n] is the solution vector

Algorithm Greedy knapsack (m, n)
{
    for i = 1 to n do
        x[i] = 0.0 ;
    U = m ;
    for i = 1 to n do
    {
        if (w[i] > U) then break ;
        x[i] = 1.0 ;
        U = U - w[i]
    }
    if (i <= n) then
        x[i] = U / w[i]
}
```

Job Sequencing with Deadlines :-

→ Consider that there are "n" jobs that are to be executed.

→ At any time T = 1, 2, 3, ... only exactly one job is to be executed.

→ Each job takes 1 unit of time.

→ If job starts before (or) at its deadline profit is obtain otherwise no profit.

→ Goal is to schedule jobs to maximize the total profit.

what is the Job sequencing with deadlines. Let $n=5$ profits $(P_1, P_2, P_3, P_4, P_5) = (20, 13, 10, 4, 1)$ and deadlines $(D_1, D_2, D_3, D_4, D_5) = (2, 1, 2, 3, 3)$.

Given $n = 5$

$P_1 = 20$; $P_2 = 13$; $P_3 = 10$; $P_4 = 4$; $P_5 = 1$

$D_1 = 2$; $D_2 = 1$; $D_3 = 2$; $D_4 = 3$; $D_5 = 3$

we have to arrange the Jobs in non increasing order based on profit values.

Job J_i	J_1	J_2	J_3	J_4	J_5
profit P_i	20	13	10	4	1
Deadlines D_i	2	1	2	3	3

J_2	J_1	J_4

0 1 2 3

Assign - $(r-1, r)$

J	Assigned slots	Job selected	Action	profit
ϕ	None	J_1	[1,2]	0
$\{J_1\}$	[1,2]	J_2	[0,1]	20
$\{J_1, J_2\}$	[1,2], [0,1]	J_3	Reject it	33
$\{J_1, J_2\}$	[1,2], [0,1]	J_4	[2,3]	33
$\{J_1, J_2, J_4\}$	[0,2], [1,2], [2,3]	J_5	Reject it	37

Hence the optimal solution is $\{J_1, J_2, J_4\}$ and the maximum profit is 37.

Find an optimal solution and maximum profit for the following Greedy Job sequencing with deadlines let $n=4$, profits $(P_1, P_2, P_3, P_4) = (100, 10, 5, 27)$, deadlines $(D_1, D_2, D_3, D_4) = (2, 1, 2, 1)$.

Given that no. of Jobs $n = 4$

we have to arrange the Jobs in non-increasing based on profit values.

Job J_i	J_1	J_2	J_3	J_4
profit P_i	100	27	10	5
Deadlines D_i	2	1	1	2

Assign - $(r_i - 1, r_i)$

J	assigned slots	Job selected	action	profit
ϕ	None	J_1	[1,2]	0
J_1	[1,2]	J_2	[0,1]	100
J_1, J_2	[0,1], [1,2]	J_3	Reject it	127
J_1, J_2	[0,1], [1,2]	J_4	Reject it	127

the optimal solution is $J = \{1, 2\}$ with a profit of 127.

(3) Find an optimal solution and maximum profit for the following Greedy Job sequencing with deadlines

(i) Let $n = 5$; $(P_1, P_2, P_3, P_4, P_5) = (20, 15, 10, 5, 1)$ and $(D_1, D_2, D_3, D_4, D_5) = (2, 2, 1, 3, 3)$

(ii) Let $n = 6$; $(P_1, P_2, \ldots, P_6) = (3, 5, 20, 18, 1, 6)$ and $(d_1, d_2, \ldots, d_6) = (1, 3, 4, 3, 2, 1)$.

(iii) Let $n = 9$; $(P_1, P_2, \ldots, P_9) = (15, 20, 30, 18, 18, 10, 23, 16, 25)$ and $(d_1, d_2, \ldots, d_9) = (7, 2, 5, 3, 4, 5, 2, 7, 3)$

<u>Note</u>:- the time complexity for the Greedy knapsack is $O(n)$

Algorithm :-

Algorithm GreedyJob (d, J, n)

// J is a set of jobs that can be completed by their deadlines

{

J = {1};

for i = 2 to n do

{

if (all Jobs in J ∪ {i} can be completed by their deadlines) then

J = J ∪ {i};

}

}.

Minimum - cost Spanning Trees :

Spanning Trees :-

-> A spanning tree of a graph G = (V, E) is a subgraph of G that is a tree and contains all the vertices of G contain no circuit.

-> An edge of spanning tree is called a branch.

-> An edge in the graph that is not in the spanning tree is called a chord.

-> Removing one edge from the spanning tree will make it disconnected.

-> Adding one edge to the spanning tree will create a loop.

-> A complete undirected graph can have n^{n-2} no. of spanning trees.

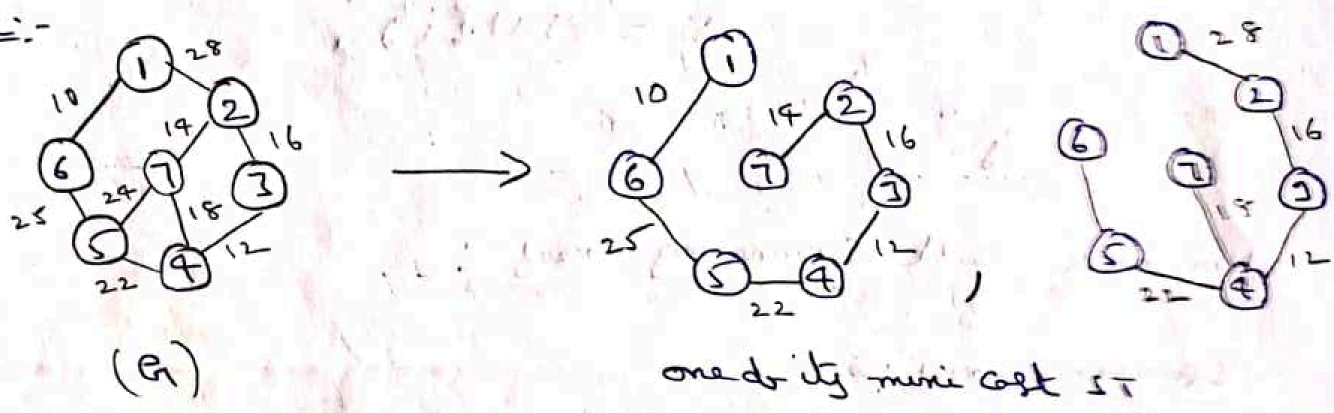
-> Every connected and undirected graph has atleast one spanning tree.

-> Disconnected graph does not have any trees.

-> From a Complete graph by removing Max $(e-n+1)$ edges we can construct a spanning tree.

-> Spanning trees are represented by using two graph searching algorithms
   (a) Breadth first search (BFS)
   (b) Depth first search (DFS).

Ex:-



Minimum Cost Spanning trees :-

If a graph $G = (V, E)$ is weighted graph then which containg least weight among all spanning trees is called minimum cost spanning tree.

there are two important algorithms to obtain minimum cost spanning tree. they are
1. Prims Algorithm
2. Kruskal's Algorithm

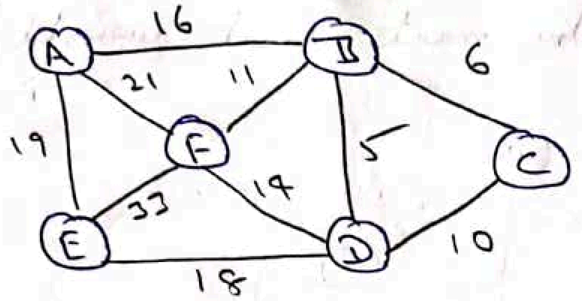
Prims Algorithm :-

-> In this method least weight edge is selected.

-> Adjacent minimum weight edge is selected.

-> In this way the procedure is continue untill all the vertices are connected without forming cycles.

Ex:-

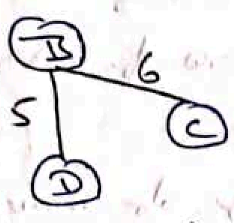


step-1 :- Initially the total weight is "0"

step-2 :- Identify least weight edge

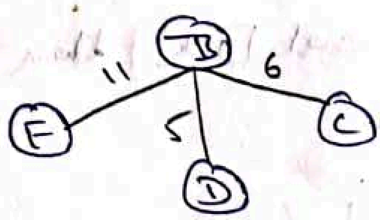
Now the total weight is 0 + ~~10~~ 5 = ~~10~~ 5

step-3 : Identify next least weight edge adjacent to B, D



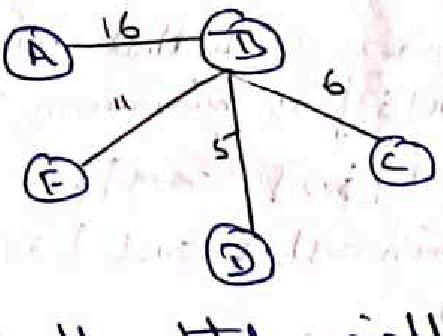
Now the total weight = 0 + 5 + 6 = 11

step 4 :- Identify next least weight edge adjacent to B, C, D



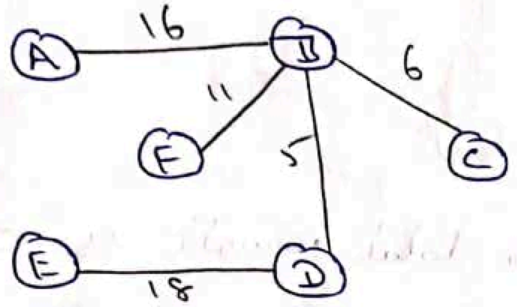
now the total weight is 5 + 6 + 11 = 22

step-5 Identify next least weight edge adjacent to B, C, D & F



now the total weight is 5 + 6 + 11 + 16 = 38

Step-6 :- Identify next least weight edge adjacent A, B, C, D & F



now the total weight is 5+6+11+16+18 = 56

Hence the total minimum cost spanning tree = 56

Algorithm prim (E, cost, n, t)

// where E is the set of edges in given graph
// cost = cost of adjacency Matrix
// n = no. of vertices
// t = used to store no. of edges in minimum spanning tree.

{

let (k, l) be an edge of minimum cost in E;

min cost = cost [k, l];

t[1,1] = k ; t[1,2] = l ;

for i = 1 to n do

if (cost [i, k] < cost [i, l]) then

near [i] = k;

else

near [i] = l ;

near [k] = near [l] = 0 ;

for i = 2 to n-1 do

{

let j be an index such that near [j] ≠ 0 and

cost [j, near [j]] is minimum ;

t [i, 1] = j ; t [i, 2] = near [j];

min cost = min cost + cost [j, near [j]]

near [j] = 0 ;

for k = 1 to n do

if ((near[k] ≠ 0) and (cost[k, near[k]] > cost[k, j]))

then near[k] = j ;

}

return mincost ;

}

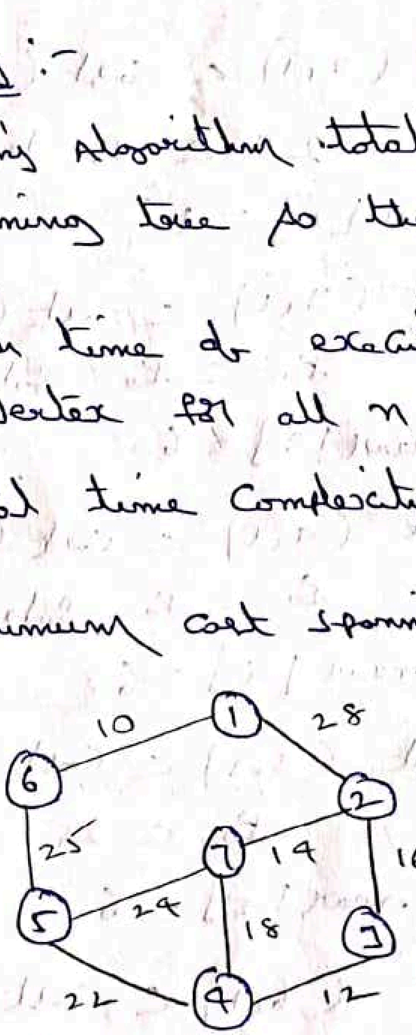
Time Complexity :-

→ In prim's Algorithm totally $n-1$ edges are added to spanning tree so the algorithm execute $n-1$ times

→ In each time of excution it needs to calculate nearest vertex for all n vertices in the graph

→ the total time complexity is $(n-1)n = n^2 - n \cong O(n^2)$

Ex:- Construct minimum cost spanning tree for the given graph



Sol:

Consider

$$\begin{array}{c|ccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
1 & 0 & 28 & \infty & \infty & \infty & 10 & \infty \\
2 & 28 & 0 & 16 & \infty & \infty & \infty & 14 \\
3 & \infty & 16 & 0 & 12 & \infty & \infty & \infty \\
4 & \infty & \infty & 12 & 0 & 22 & \infty & 18 \\
5 & \infty & \infty & \infty & 22 & 0 & 25 & 24 \\
6 & 10 & \infty & \infty & \infty & 25 & 0 & \infty \\
7 & \infty & 14 & \infty & 18 & 24 & \infty & 0
\end{array}$$

Edges of ST.

t [i, j]

source destination

min cost edge is (1,6).
t[1,1] = 1; t[1,2] = 6
min cost = cost (1,6) = 10.

for i = 1 to 7 do

i = 1   if cost (1,1) < cost (1,6)
0 < 10 true
near[1] = 1;

i = 2   if cost (2,1) < cost (2,6)
28 < ∞ true
near[2] = 1;

i = 3   if cost (3,1) < cost (3,6)
∞ < ∞ false
near[3] = 6.

i = 4   if cost (4,1) < cost (4,6)
∞ < ∞ false
near[4] = 6.

i = 5   if cost (5,1) < cost (5,6)
∞ < 25 false
near[5] = 6.

i = 6   if cost (6,1) < cost (6,6)
10 < 0 false
near[6] = 6.

i = 7   if cost (7,1) < cost (7,6)
∞ < ∞ false
near[7] = 6.

1	2	3	4	5	6	7
1	1	6	6	6	6	6

near [1] = 0 ; near [6] = 0.

1	2	3	4	5	6	7
0	1	6	6	6	0	6

j = 1 near [1] ≠ 0 false

j = 2 near [2] ≠ 0 true

cost (2, near [2]) = cost (2, 1) = 28.

j = 3 near [3] ≠ 0 true

cost (3, near [3]) = cost (3, 6) = ∞

j = 4 near [4] ≠ 0 true

cost (4, near [4]) = cost (4, 6) = ∞

j = 5 near [5] ≠ 0 true

cost (5, near [5]) = cost (5, 6) = 25

j = 6 near [6] ≠ 0 false

j = 7 near [7] ≠ 0 true

cost (7, near [7]) = cost (7, 6) = ∞

Select j = 5

t [2, 1] = 5 ; t [2, 2] = 6 ;

min cost = min cost + cost [5, 6]

= 10 + 25

= 35

near [5] = 0.

1	2	3	4	5	6	7
0	1	6	6	0	0	6

$k=1$ near[1] $\neq 0$ false

$k=2$ near[2] $\neq 0$ true

cost(2, near[2]) > cost(2,5)

cost(2,1) > cost(2,5)

28 > ∞ false

$k=3$ near[3] $\neq 0$ true

cost(3, near[3]) > cost(3,5)

cost(3,6) > cost(3,5)

∞ > ∞ false

$k=4$ near[4] $\neq 0$ true

cost(4, near[4]) > cost(4,5)

cost(4,6) > cost(4,5)

∞ > 22 true

near[4] = 5;

$k=5$ near[5] $\neq 0$ false

$k=6$ near[6] $\neq 0$ false

$k=7$ near[7] $\neq 0$ true

cost(7, near[7]) > cost(7,5)

cost(7,6) > cost(7,5)

∞ > 24 true

near[7] = 5;

1	2	3	4	5	6	7
0	1	6	5	0	0	5

Step 2 :-

$j = 2$ near $[2] \neq 0$ true

cost $(2, near[2])$ = cost $(2,1)$ = 28

$j = 3$ near $[3] \neq 0$ true

cost $(3, near[3])$ = cost $(3,6)$ = 20

$j = 4$ near $[4] \neq 0$ true

cost $(4, near[4])$ = cost $(4,5)$ = 22

$j = 7$ near $[7] \neq 0$ true

cost $(7, near[7])$ = cost $(7,5)$ = 24

Select $j = 4$

$t[3,1] = 4$; $t[3,2] = 5$.

min cost = min cost + cost $(4,5)$

= 35 + 22

= 57.

near $[4] = 0$.

1	2	3	4	5	6	7
0	1	6	0	0	0	5

$k = 2$ near $[2] \neq 0$ true

cost $(2, near[2])$ > cost $(2,4)$

cost $(2,1)$ > cost $(2,4)$

28 > 20 false

$k = 3$ near $[3] \neq 0$ true

cost $(3, near[3])$ > cost $(3,4)$

cost $(3,6)$ > cost $(3,4)$

20 > 12 true

near $[3] = 4$.

k = 7   near[7] ≠ 0 true

cost (7, near[7]) > cost (7, 4)

cost (7, 5) > cost (7, 4)

24 > 18 true

near[7] = 4.

1	2	3	4	5	6	7
0	1	4	0	0	0	4

step-3 :

j = 2   near[2] ≠ 0 true

cost (2, near[2]) = cost (2, 1) = 28

j = 3   near[3] ≠ 0 true

cost (3, near[3]) = cost (3, 4) = 12

j = 7   near[7] ≠ 0 true

cost (7, near[7]) = cost (7, 4) = 18

select j = 3

t[4,1] = 3 ; t[4,2] = 4

min cost = min cost + cost (3, 4)

= 57 + 12

= 69.

near[3] = 0.

1	2	3	4	5	6	7
0	1	0	0	0	0	4

k = 2   near[2] ≠ 0 true

cost (2, near[2]) > cost (2, 3)

cost (2, 1) > cost (2, 3)

28 > 16 true

near [2] = 3

k = 7 near [7] ≠ 0 true

cost (7, near[7]) > cost (7,3)

cost (7, 4) > cost (7,3)

18 > 26 false

1	2	3	4	5	6	7
0	3	0	0	0	0	4

Setep - 4 :-

j = 2 near [2] ≠ 0 true

cost (2, near[2]) = cost (2,3) = 16

j = 7 near [7] ≠ 0 true

cost (7, near[7]) = cost (7,4) = 18

select j = 2

t [5,1] = 2 ; t [5,2] = 3

min cost = min cost + cost (2,3)

= 69 + 16

= 85

near [2] = 0.

1	2	3	4	5	6	7
0	0	0	0	0	0	4

k = 7 near [7] ≠ 0 true

cost (7, near[7]) > cost (7,2)

cost (7, 4) > cost (7,2)

18 > 14 true

near [7] = 2

1	2	3	4	5	6	7
0	0	0	0	0	0	2

Satep -5 :-

$j = 7$ near $[7] \neq 0$ true

$cost(7, near[7]) = cost(7,2) = 14$

select $j = 7$

$t[6,1] = 7 ; t[6,2] = 2$

$$\begin{aligned} \text{min cost} &= \text{min cost} + cost(7,2) \\ &= 85 + 14 \\ &= 99 \end{aligned}$$

near $[7] = 0$.

1	2	3	4	5	6	7
0	0	0	0	0	0	0

Hence $t[1,1] = 1 ; t[1,2] = 6$

$t[2,1] = 5 ; t[2,2] = 6$

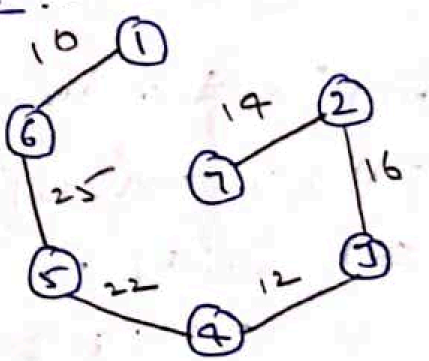
$t[3,1] = 4 ; t[3,2] = 5$

$t[4,1] = 3 ; t[4,2] = 4$

$t[5,1] = 2 ; t[5,2] = 3$

$t[6,1] = 7 ; t[6,2] = 2$

Minimum spanning tree :-



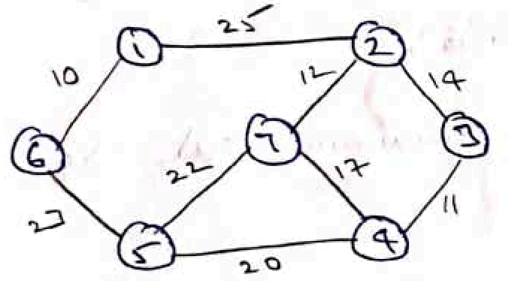
Kruskal's Algorithm :-

→ In a Kruskal's algorithm always the minimum cost edge has to be selected.

→ It is not necessary that selected optimum edge is adjacent.

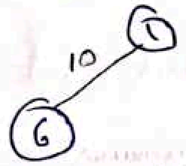
→ In this way the procedure is continued until all vertices are connected without forming cycles.

Ex:-



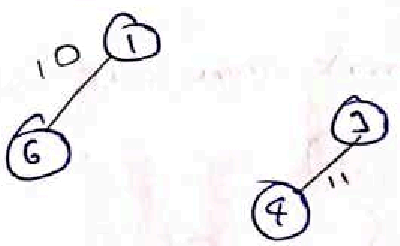
Soln Step 1: Initially the total weight is 0

Step 2: Identify minimum cost edge from the graph



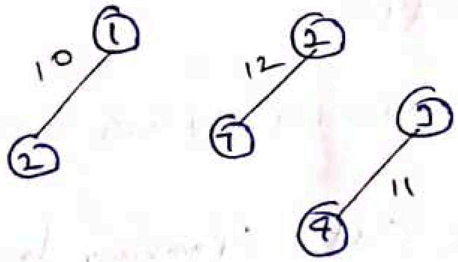
Total weight = 0 + 10 = 10

Step 3: Identify minimum cost edge from the graph



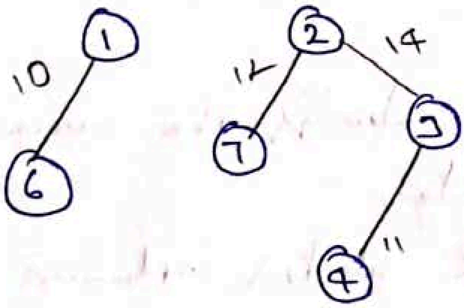
Total weight = 0 + 10 + 11 = 21

Step 4: Identify minimum cost edge from the graph



Total weight = 0 + 10 + 11 + 12
= 33

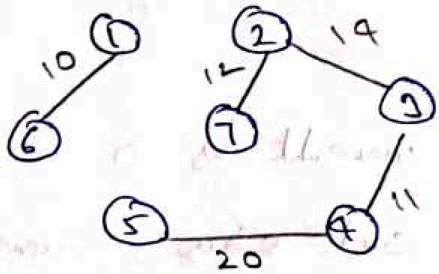
Step 5: Identify minimum cost edge from the graph



Total weight = 0+10+11+12+14
= 47

Step 6: In this step, the next minimum edge cost makes from 7 to 4. It makes a closed cycle. Hence not selected.

Step 7: Identify next minimum cost edge from the graph

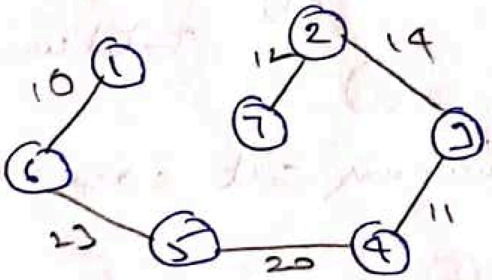


Total weight = 0+10+11+12+14+20
= 67

Step 8:

In this step, the next minimum cost edge from 7 to 5. It makes a closed cycle, Hence not selected

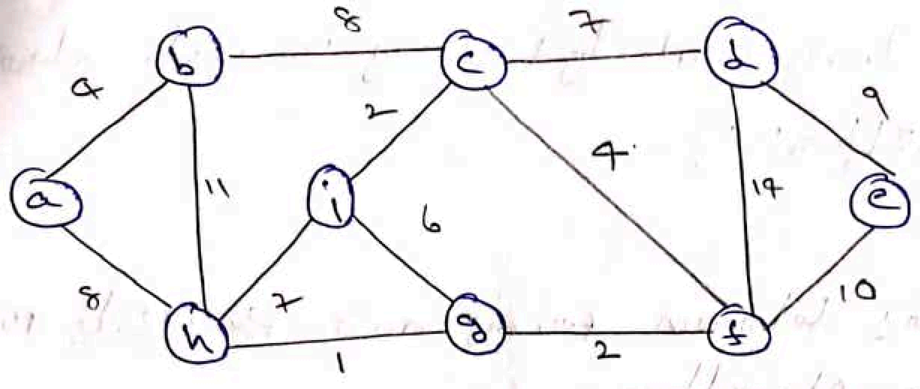
Step 9:- Identify next mini cost edge from the graph



Total weight = 0+10+11+12+14+20+23 = 90

Hence the minimum cost spanning tree is 90.

A



Ans :- 37

Algorithm :-

Algorithm Kruskal (E, cost, n, t)

// E is the set of edges in G.

// n is the no. of vertices

// cost (u,v) is the cost of edge (u,v).

// t is the set of edges in the minimum cost spanning tree.

```
{
    i = 0 ;
    min cost = 0 ;
    while (i ≤ n-1)
    {
        Delete a minimum cost edge (u,v) ;
        set_J := find(u) ;
        set_K := find(v) ;
        if (J ≠ K) then (so in different trees)
        {
            i = i+1 ;
            t[i,1] = u ;
            t[i,2] = v ;
            mincost = min cost + cost(u,v) ;
            union (J,K) ;
        }
    }
}
```

Note:- The time complexity of greedy Kruskal's algorithm is $O(|E|\log|E|)$ — Simple

Q Differences between Prim's and Kruskal's Minimum Spanning Algorithm.

solu

Prim's	Kruskal's
① It works by choosing the adjacent vertices from the selected set of vertices.	① It works by choosing the least weight edges, it organizes the edges by their weights.
② The generation of minimum spanning tree in this algorithm is based on the selection of graph vertices and it initiates with vertex	② In this algorithm the generation of minimum spanning tree depends on the edges and initiates with an edge.
③ This algorithm always generates MST with connected components	③ In this algorithm MST may not have connected components. (i.e. minimum spanning forest)
④ Prim's algorithm performs better in the dense graph	④ Kruskal's algorithm performs better in the sparse graph
⑤ The time complexity is $O(n^2)$	⑤ The time complexity is $O(|E|\log|E|)$

Single - Source Shortest Paths (Dijkstra's Algorithm)

→ In this problem the given graph is a weighted and directed graph.

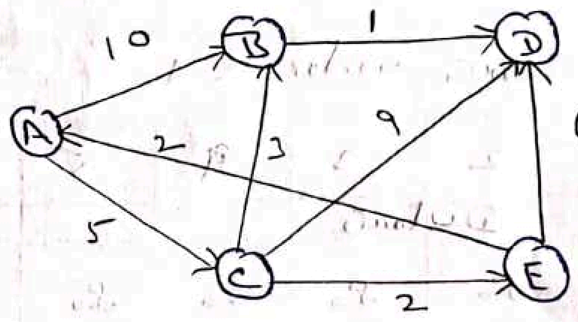
→ Dijkstra's algorithm is used to represent the distance between two cities

→ In single source shortest path problem, the shortest distance from a single vertex is called source and the last vertex is called destination

→ Finally, we can written the shortest path and minimum distance.

<u>Note</u>:- "It is Assumed that all the weights are positive"

Ex:- ①



Let source vertex = A

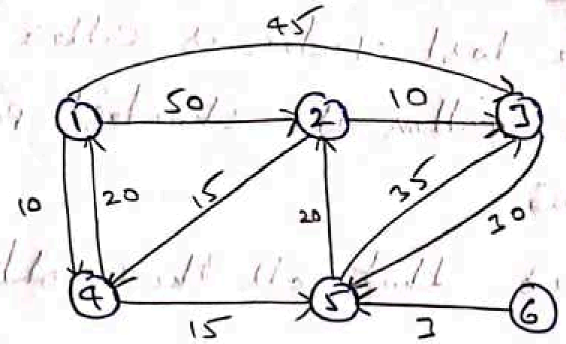
Selected vertex	A	B	C	D	E
		Distance			
A	$\boxed{0}$	∞	∞	∞	∞
C		10	$\boxed{5}$	∞	∞
E		8		14	$\boxed{7}$
B		$\boxed{8}$		13	
D				$\boxed{9}$	

if $d(u) + c(u,v) < d(v)$ then $d(v) = d(u) + c(u,v)$

Hence single source shortest path from each vertex is summarized below

source vertex	Minimum distance	Path
A → B	8	A → C → B
A → C	5	A → C
A → D	9	A → C → B → D
A → E	7	A → C → E

②



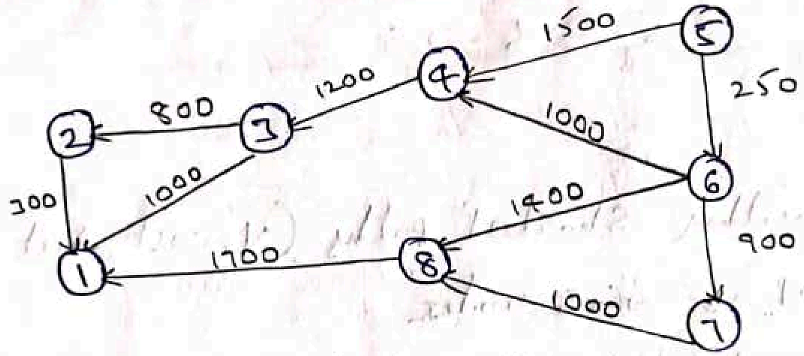
Let source vertex = 1

selected vertex	1	2	3	4	5	6
	Distance					
1	$\boxed{0}$	∞	∞	∞	∞	∞
4		50	45	$\boxed{10}$	∞	∞
5		50	45		$\boxed{25}$	∞
2		$\boxed{45}$	45			∞
3			$\boxed{45}$			∞
6						$\boxed{\infty}$

Hence single source shortest path from each vertex is summarized below.

Vertex	Minimum distance	Path
2	45	1 → 4 → 5 → 2
3	45	1 → 3
4	10	1 → 4
5	25	1 → 4 → 5
6	∞	NO

(3)



Let source vertex = 5

selected vertex	1	2	3	4	5	6	7	8
	Distance							
5	∞	∞	∞	∞	$\boxed{0}$	∞	∞	∞
6	∞	∞	∞	1500		$\boxed{250}$	∞	∞
7	∞	∞	∞	1250			$\boxed{1150}$	1650
4	∞	∞	∞	$\boxed{1250}$				1650
8	∞	∞	2450					$\boxed{1650}$
3	3350	∞	$\boxed{2450}$					
2	3350	$\boxed{3250}$						
1	$\boxed{3350}$							

Hence single source shortest path from each vertex is summarized below.

vertex	Minimum distance	Path
1	3350	5 → 6 → 8 → 1
2	3250	5 → 6 → 4 → 3 → 2
3	2450	5 → 6 → 4 → 3
4	1250	5 → 6 → 4
6	250	5 → 6
7	1150	5 → 6 → 7
8	1650	5 → 6 → 8

Note :-

Algorithm :-

```
Algorithm shortest paths (v, cost, dist, n)
// v is source vertex
// cost (u,v) is the cost of edge (u,v).
// dist [j], 1≤j≤n, is set to the length of the
// shortest path from vertex v to vertex j in a
// diagraph G with n-vertices.
{
    for i := 1 to n do
    {
        s[i]:= False ; dist [i] : = cost(v,i);
    }
    s[v]: = True ; dist [v] : = 0.0;
    for i := 2 to n do
    {
        choose a vertex u such that s[u]=False
        and dist [u] is minimum;
            s [u] := true;
        for each w adjacent to u with s[w] = false do
        if (dist [w] > dist [u] + cost [u,w]) then
            dist [w] = dist [u] + cost [u,w];
    }
}
```

Note:- The Time Complexity of Greedy single source shortest path is $O(n^2)$.

OPTIMAL Merge Patterns:

In Merging already the given list of elements are in sorted order

Ex:- list A contains 4 elements 3 5 7 9

list B contains 4 elements 2 4 8 11

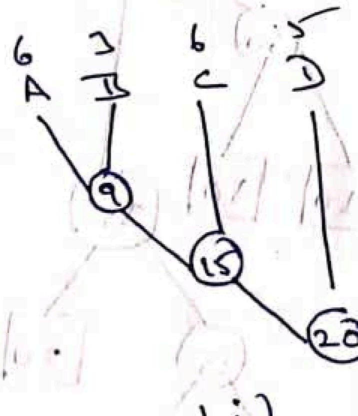
Now Merge A and B

A	B	C
3	2	2
5	4	3
7	8	4
9	11	5
		7
		8
		9
		11

=> the size of C list is 4+4=8

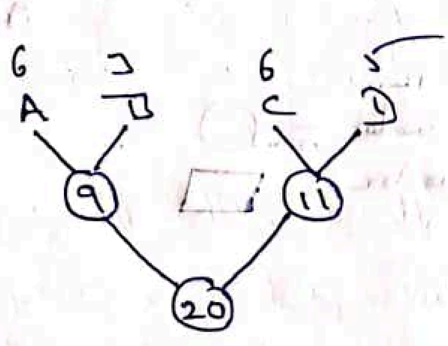
=> In two-way merging merge two list for each and every time.

Ex:- the given lists are A, B, C, D and its corresponding sizes are 6, 3, 6, 5

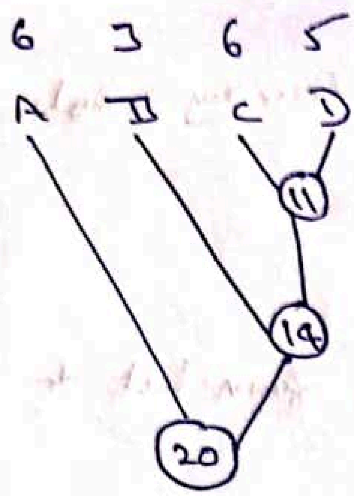


(i)

=> $9+15+20 = 44$



=> $9+11+20 = 40$



$\Rightarrow 11+14+20 = 45$

Hence the minimum cost optimal solution is 40

→ To Generate optimal solution the greedy method introduce the dynamic problem is called "optimal Merge pattern".

Ex:- Construct optimal Binary merge tree for the following files $(x_1, x_2, x_3, x_4, x_5) = (20, 30, 10, 5, 30)$.

Soln Given $(x_1, x_2, x_3, x_4, x_5) = (20, 30, 10, 5, 30)$

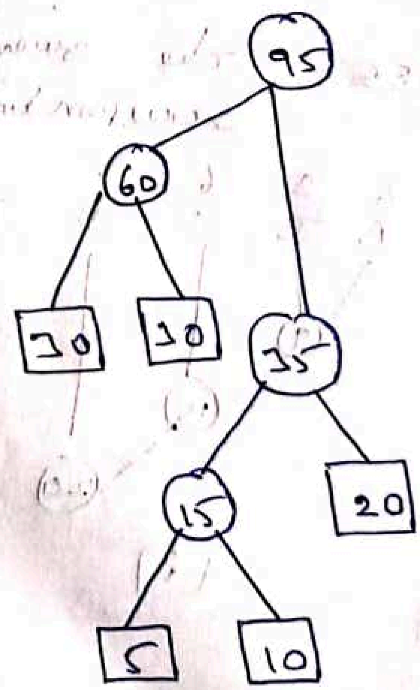
Arrange the files in Ascending order

$\underbrace{5 \quad 10}_{15} \quad 20 \quad 30 \quad 30$

$\underbrace{15 \quad 20}_{35} \quad 30 \quad 30$

$\underbrace{30 \quad 30}_{60} \quad 35$

$\underbrace{35 \quad 60}_{95}$



Parent vertex = ○

child vertex = □

Binary merge tree

Total no. of Merges = Sum of Internal Nodes

$= 15 + 35 + 60 + 95$

$= 205$

Also we have Total no. of Merges $= \sum d_i x_i$

where d_i – distance from root node to leaf node

x_i – leaf nodes

$= 3\times5 + 3\times10 + 2\times20 + 2\times30 + 2\times30$

$= 15 + 30 + 40 + 60 + 60$

$= 205$

Ex:- Construct optimal Binary Merge tree for the following files $(x_1, x_2, x_3, x_4, x_5, x_6) = (2, 3, 5, 7, 9, 13)$

Sol: Given $(x_1, x_2, x_3, x_4, x_5, x_6) = (2, 3, 5, 7, 9, 13)$

Arrange the files in Ascending order

$\underbrace{2 \quad 3}_{5} \quad 5 \quad 7 \quad 9 \quad 13$

$\underbrace{5 \quad 5}_{10} \quad 7 \quad 9 \quad 13$

$\underbrace{7 \quad 9}_{16} \quad 10 \quad 13$

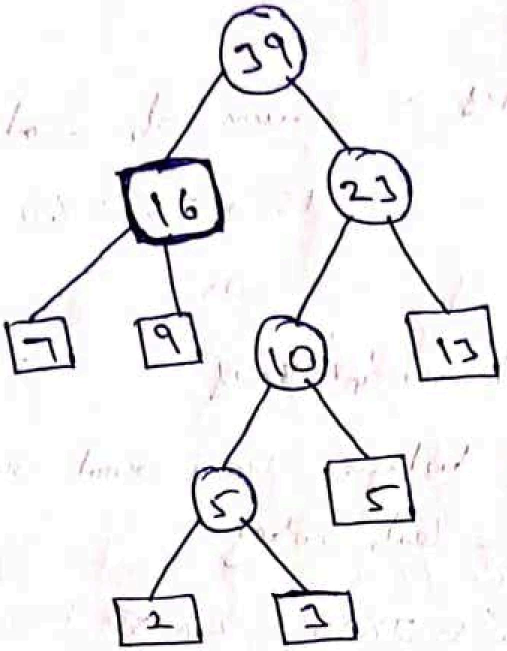
$\underbrace{10 \quad 13}_{23} \quad 16$

$\underbrace{16 \quad 23}_{39}$

Total no. of Merges = Sum of Internal nodes

$= 5 + 10 + 16 + 23 + 39$

$= 93$



Binary Merge tree

Also total no. of Merges $= \sum d_i x_i$

where d_i – Distance from root node to leaf node

x_i – leaf nodes.

$= 4\times2 + 4\times3 + 3\times5 + 2\times7 + 2\times9 + 2\times13$

$= 8 + 12 + 15 + 14 + 18 + 26$

$= 93.$

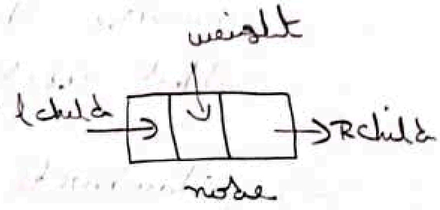
Ex:- Construct optimal Binary merge tree for the following files of sizes (28, 32, 12, 5, 84, 53, 91, 35, 3, 11)

Ans 1004

note:- the time complexity of Greedy optimal merge patterns is $O(n^2)$

Algorithm :-

```
Tree node = record
{
    Tree node * lchild;
    Tree node * rchild;
    Integer weight;
}
```



n – no. of files

```
Algorithm optimal Merge Tree (n)
{
    for i := 1 to n-1 do
    {
        pt = new Tree node;
        (pt -> lchild) := Least (list)
        (pt -> rchild) := Least (list)
        (pt -> weight) := ((pt -> lchild) -> weight)
                        + ((pt -> rchild) -> weight)
        Insert (list, pt);
    }
    return least (list);
}
```

Huffman codes

-> Data Compression is useful when we are Communicati.. over a low-bandwidth chanel and we wish to minimize the time needed to transmit the data. The method used for data Compression is "Huffman coding"

-> standard encoding schemes such as ASCII and unicode schemes use fixed length binary strings to encode characters (8 bits in ASCII, 16 bits in unicode). where as Huffman code uses variable length encoding.

-> In Huffman coding, the optimization is based on character frequencies for each character, count the number of times character appears in the given text

-> Huffman codes are widely used and a very effective technique for compressing data saving of 20% to 90%. depending on the characteristic of the data being compressed.

-> In this code, more frequently occuring letters have short codes, less frequently occuring letters have Large codes

-> Huffman coding used in FAX Machines, Computer Networks, High definition TV, Modems.

Ex:- Information transmitted over the internet contains the following characters with their associated frequencies

character	A	E	L	N	O	S	T
Frequency	45	65	13	45	18	22	53

(i) Build Huffman code tree for the message and find the codeword for each character

(ii) what is the total number of bits to be transmitted?

Arrange the character frequencies in Ascending order

L	O	S	N	A	T	E
13	18	22	45	45	53	65

13 18 → 31

22 31 45 45 53 65

22 31 → 53

45 45 53 53 65

45 45 → 90

53 53 65 90

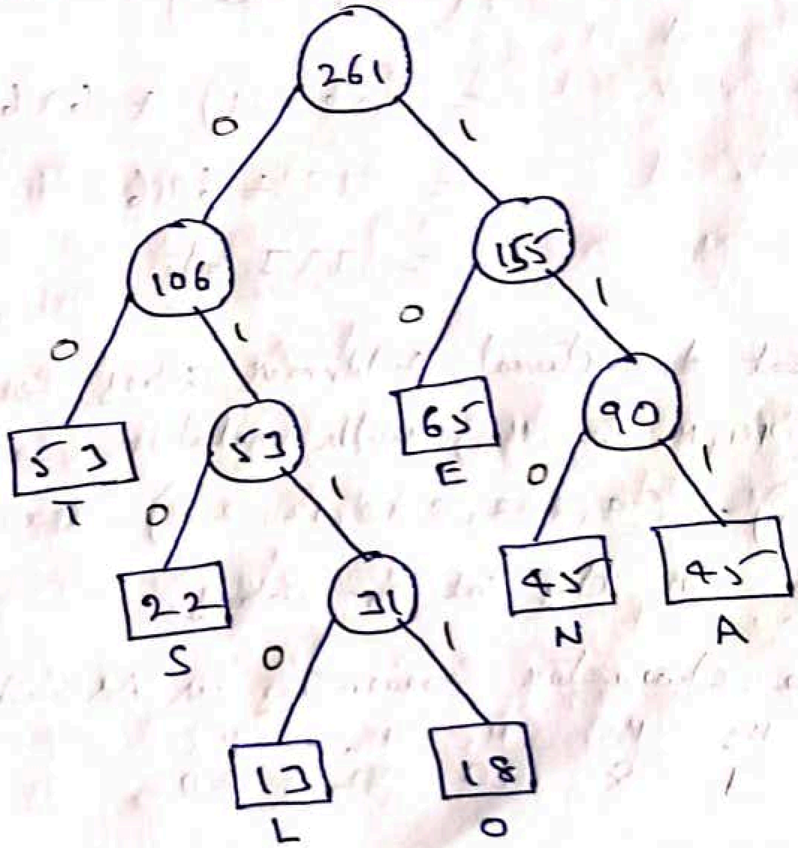
53 53 → 106

65 90 106

65 90 → 155

106 155

106 155 → 261



Huffman Code Tree

Note: Total no. of Merges = sum of internal nodes

$= 31 + 53 + 90 + 106 + 155 + 261 = 696$

character	code word	Count / frequency	Message size
A	111	45	3×45 = 135
E	10	65	2×65 = 130
L	0110	13	4×13 = 52
N	110	45	3×45 = 135
O	0111	18	4×18 = 72
S	010	22	3×22 = 66
T	00	53	2×53 = 106
7×8 bit = 56 bits	21 bits	261×8 = 2088	696 bits

Hence, total number bits to be transmitted

$$= (\text{Tree / Table}) + \text{Message}$$

$$= (56+21) + 696$$

$$= 77 + 696$$

$$= 773 \text{ bits.}$$

Ex:- obtain a set of optimal Huffman codes for the messages $(M_1, M_2, \dots, M_7)$ with relative frequencies $(q_1, q_2, \dots, q_7) = (4, 5, 7, 8, 10, 12, 20)$. Draw the decode tree for this set of codes..

soln Arrange the character frequencies in Ascending order

M_1	M_2	M_3	M_4	M_5	M_6	M_7
4	5	7	8	10	12	20

$\underbrace{4 \quad 5}_{9}$

$\underbrace{7 \quad 8}_{15} \quad 9 \quad 10 \quad 12 \quad 20$

$\underbrace{9 \quad 10}_{19} \quad 12 \quad 15 \quad 20$

12 15 19 20

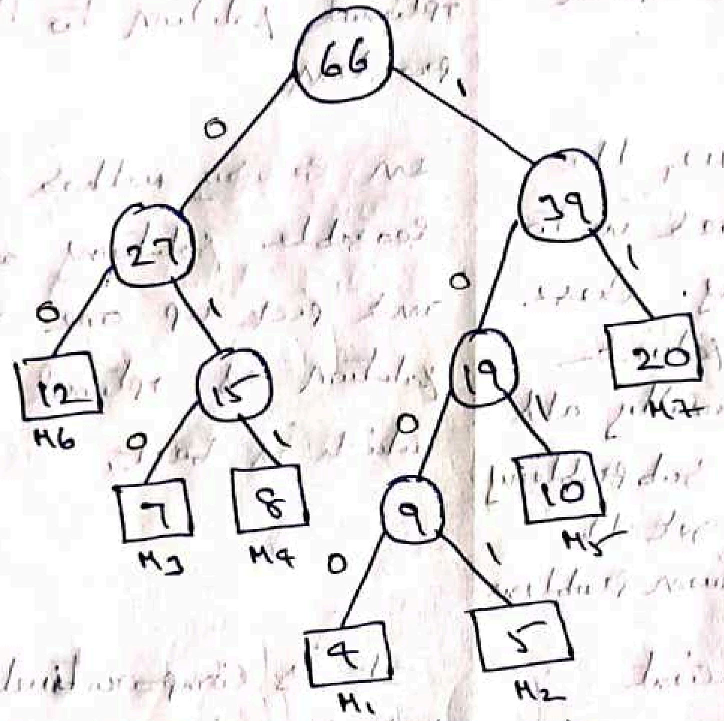
27

19 20 27

39

27 39

66



Huffman Code tree

character	code word	frequencies	Message size
M1	1000	4	4×4 = 16
M2	1001	5	4×5 = 20
M3	010	7	3×7 = 21
M4	011	8	3×8 = 24
M5	101	10	3×10 = 30
M6	00	12	2×12 = 24
M7	11	20	2×20 = 40
	21 bits	66×8=528	175 bits

Q. Differences between Greedy method and divide and Conquer.

Solu:

	Divide and Conquer	Greedy method
1	It is one of the Algorithm design technique.	It is also one of the algorithm design technique.
2	It is used to obtain a solution to the given problem	It is used to obtain a optimal solution to the given problem
3	In this technique, the problem is divided in to small subproblems. these subproblems are solved independently. Finally all the solutions of subproblems are collected to get the solution to the given problem	In Greedy method, a set of feasible solutions are generated and pick up one feasible solution as optimal solution which is best.
4	It is less efficient because of rework on solutions.	It is Comparatively efficient but there is no as such guarantee of getting optimal solution.
5	In this method, duplicate solutions may be obtained.	In this method, the optimum selection is without revising previously Generated solutions.
6	Examples: Merge sort, Quick sort, Binary search etc.	Examples:- Knapsack problem, minimum cost spanning trees, optimal merge patterns etc.