DDA

```
Algorithm DAndC(P) → Problem can be solved
{                    name of Algorithm,
                     boolean function.   Solution of Problem.
  if Small(P) then return S(P);
  else
  {
     divide P into smaller instances
          P1, P2 .... Pk, k ≥ 1;
     Apply D And C to each of these subproblems;
  return combine (D And C(P1), D and C(P2) ..... DA
                  ↳ function.
```

(CPPS) :-

3

Divide - and - conquer

n inputs

$1 \le k \le n$

control Abstraction.

computing time on n inputs

$$T(n) = \begin{cases} g(n) & n \text{ is small} \\ T(n_1) + T(n_2) + \cdots T(n_k) + f(n) & \text{Otherwise.} \end{cases}$$

Computing time or Comparing.

$$T(n) = \begin{cases} T(1) & n = 1. \\ aT(n/b) + f(n) & n > 1. \quad n = b^k \end{cases}$$ General formula

1) $a = 2, b = 2, T(1) = 2, f(n) = n.$

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

$T(n/2) : 2(2T(n/4)) + n.$

$= 4T(n/4) + 2n$

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

$= 8T(n/8) + 3n \rightarrow 8(2T(n/16) + n/8) + 3n.$

$= 16T(n/16) + 4n.$

$= 16[2T(n/32) + n/16) + 4n.$

$= 32T(n/32) + 5n.$

$= 2^k T(n/2^k) + kn.$

$= 2^k T(1) + k2^k$

$= nT(1) + kn$

$= 2n + kn.$

$= 2n + n\log_2 n.$

$\therefore n = 2^k$

$k = 5$

$n = 32$

$n = 25.$

$n = 2^k$: take log both side

$\log n = k \log 2$

$k = \log_2 n$

$$\boxed{T(n) = 2n + n\log_2 n}$$

Substitution method :

DDA

Binary Search :- (Straight)

```
Algorithm Binsearch (a, n, x)        search element
{
    low = 1, high = n.
    while (low ≤ high) do
    {
        mid := ⌊(low + high)/2⌋ ;
        if (x < a[mid]) then
            high := mid - 1;
```

1.+14/2 = 7.5

7.5 → 7 by using floor function.

```
else if (x > a[mid]) then
    low = mid + 1
else
    return mid ;
}
return 0
}
```

(8) Algorithm BinSearch (a, i, l, x) (Recursive)

```
{
  if (l = i) then
  {
    if (x = a[i]) then
      return i ;
    else
      return 0 ;
  }
  else
  {
    mid := [(i + l)/2] ;
    if (x = a[mid]) then return mid ;
    else if (x < a[mid]) then
      return BinSearch (a, i, mid - 1, x) ;
    else
      return BinSearch (a, mid + 1, l, x) ;
  }
}
```

Ex :- -15, 6, 10, 17, 19, 23, 54, 82, 101, 112, 125, 131, 142

To find :

x = 151, x = -14, x = 9

1) x = 151

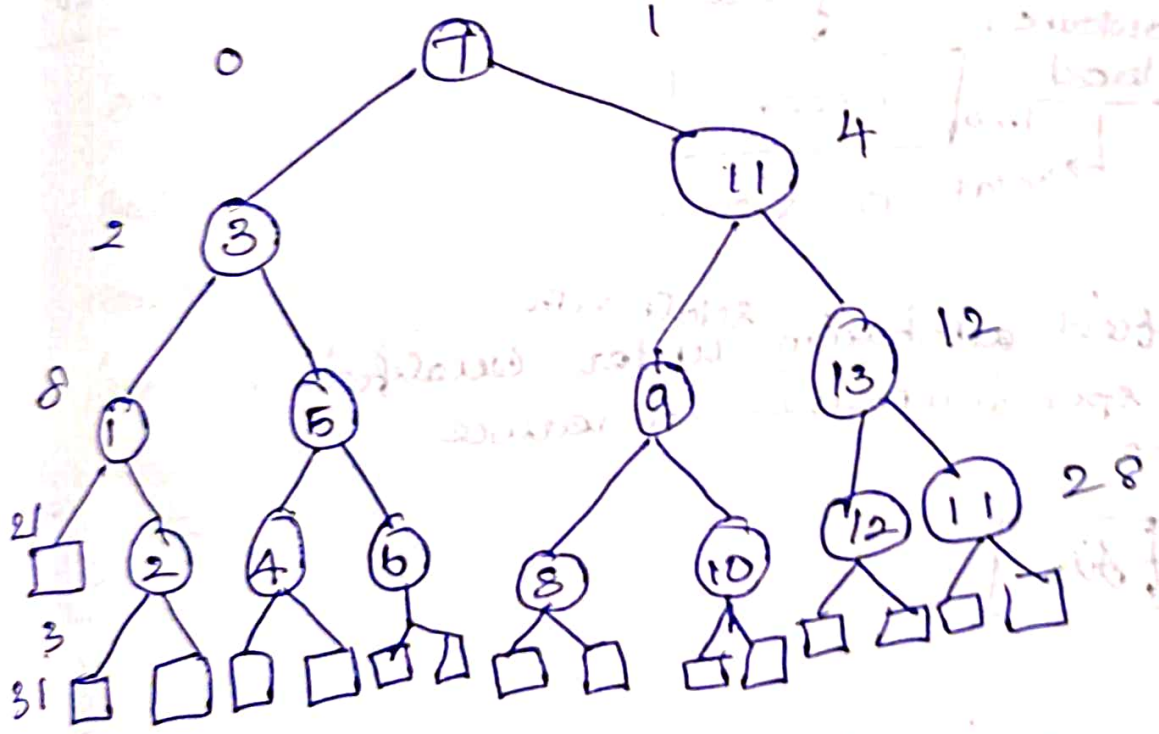
$mid = \frac{Max - 1}{2}$

	low	high	mid
1	1	14	7
2	8	14	11
3	12	14	13
4	14	14	14.

x = -14

	low	high	mid
		14	7
1	1		3
2	1	6	1
3	1	2	2
4	2	2	2.
5	2	1	

	low	high	mid
			X=9
		14	7
2	1	6	3
3	4	6	5



Complicity of Binary Algo - O(log n)

E = I + 2n

E = 59

I = 31.

A(n) = E/(n+1)

DAA

Straight forward maximum and minimum

```
Algorithm StraightMaxMin (a,n,max,min):
{
  max := min := a[1];
  for i := 2 to n do
  {
    if (a[i] > max) then max := a[i];
    if (a[i] < min) then min := a[i];
  }
}
```

Algorithm MaxMin (i, j, max, min) (Recursive Alg divide and conquer)

```
{
  if (i = j) then max := min := a[i];
  else if (i = j-1) then
  {
    if (a[i] < a[j]) then
    {
      max := a[j];
      min := a[i];
    }
    else
    {
      max := a[i];
      min := a[j];
    }
  }
  else
  {
    mid := ⌊(i+j)/2⌋;
    MaxMin (i, mid, max, min);
    MaxMin (mid+1, j, max1, min1);
    if (max < max1) then
      max := max1;
    if (min > min1) then
      min := min1;
  }
}
```

bestcase = n-1
worst case = 3(n/2) - 2
Avg case = 3n/2 - 1

i value less than j.
↳ $1 \le i \le j \le n$

→ floor function

Comparision
2(n-1)
2(9-1)
2(8)
= 16.

Eg:-

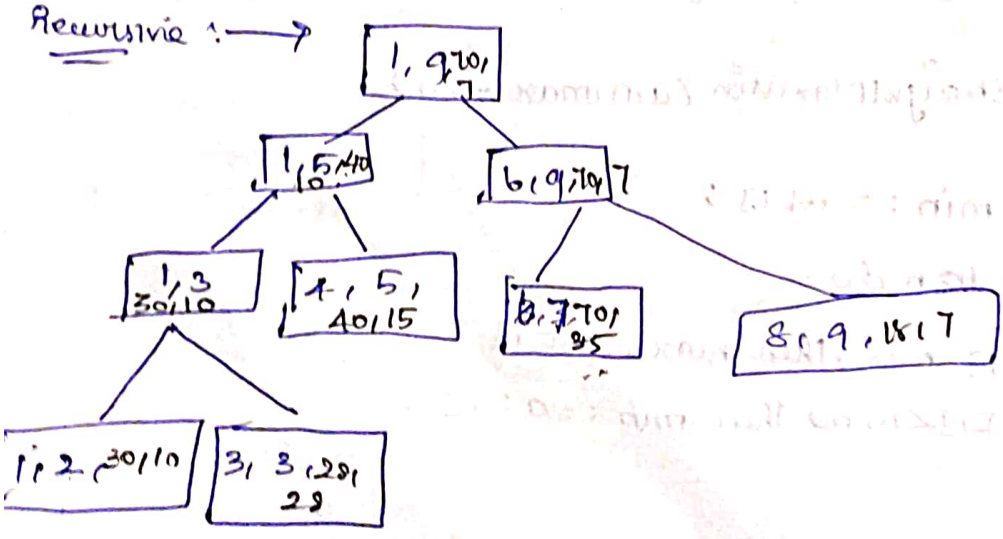
10, 30, 28, 15, 40, 35, 70, 25, 18 n = 9.

n = 9

(Straight method with out divide and conque method)

max = 10 30 40 70
min = 10 7

Recursive :→



$$T(n) : \begin{cases} 2T(n/2) \text{ or } T(n/2)+T(n/2)+1 & n > 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$

n must be pow of 2

$$T(n) = 2T(n/2) + 2$$
$$= 2(2T(n/4) + 2) + 2$$
$$= 4T(n/4) + 4 + 2$$
$$= 4(2T(n/8) + 2) + 4 + 2$$
$$= 8T(n/8) + 8 + 4 + 2$$
$$= 8(2T(n/16) + 2) + 8 + 4 + 2$$
$$= 16T(n/16) + 16 + 8 + 4 + 2$$
$$\vdots$$
$$= 2^{k-1} T(2) + \sum_{1 \le i \le k-1} 2^i$$

$\therefore$ assume $n = 2^k$

$$n = \frac{2^k}{2^{k-1}} = 2$$

$$= 2^{k-1} + 2^k - 2$$
$$= n/2 + n - 2$$
$$= \frac{3n}{2} - 2$$

$\therefore n = 2^k = 2^1$

$$2^n T(n) + \le 2^n$$

$$+ C \cdot \frac{n}{2} + n > 2$$

$$2n \quad \frac{3n}{2}$$

$$T(n) = \frac{3n}{2} - 2$$

Data Algorithm .—:

Merge sort :-

```
Algorithm Merge (low, high)
{
  if (low < high) then
  {
    mid := ⌊(low+high)/2⌋;
    MergeSort (low, mid);
    Merge Sort (low, mid);
     Merge (low, mid high);
  }
}
Algorithm Merge (low mid, high)
 h := low; j := low; j := mid+1;
while (ch ≤ mid) and (j ≤ high) do
 { if (a[h] ≤ a[j]) then
   {
     b[i] := a[h];
     h := h+1;
   }
   else
    {
      b[i] := a[i];
      j := j+1;
    }
   i = i+1;
}
if (h > mid) then
   for k := j to high do
   {
     b[i] = a[k];
      i := i+1;
    }
else
    for k := h to mid do
      {
        b[i] := a[k];
         i := i+1;
       }
```

for k := low to high do a[k] := b[k];

3

Eg: (25, 10, 60, 15, 70, 55, 28, 90, 40, 12)

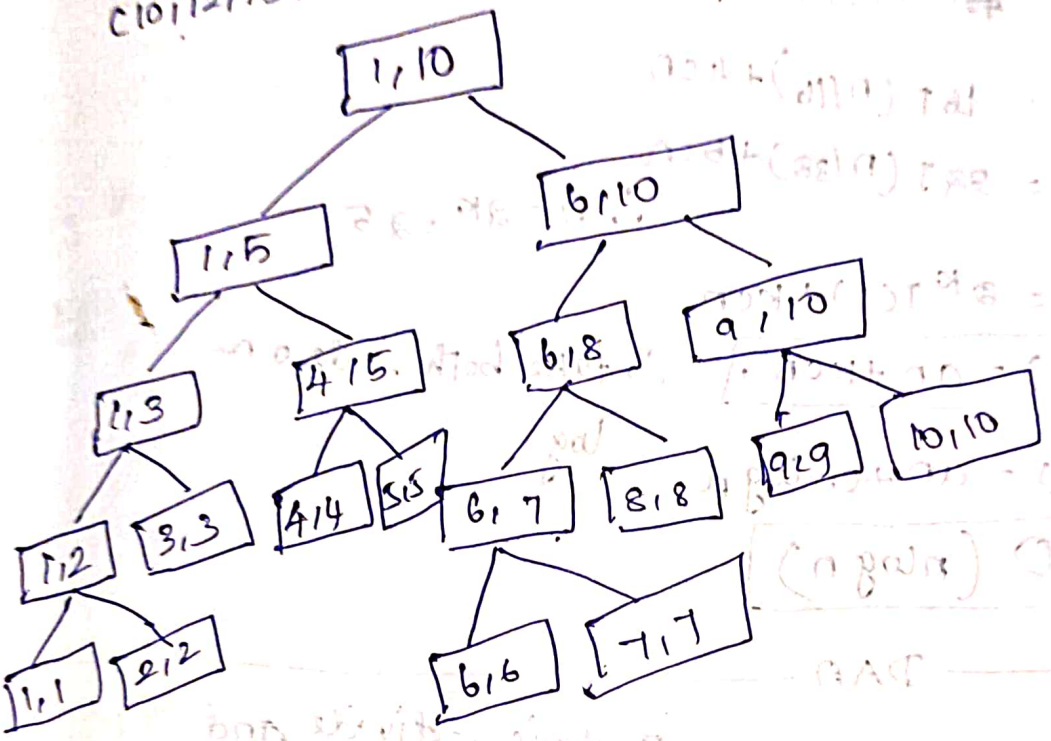
(25), (10), (60), (15), (70), (55), (28), (90), (40), (12)

(10, 25) (15, 60) (55, 70) (28, 90) (12, 40)

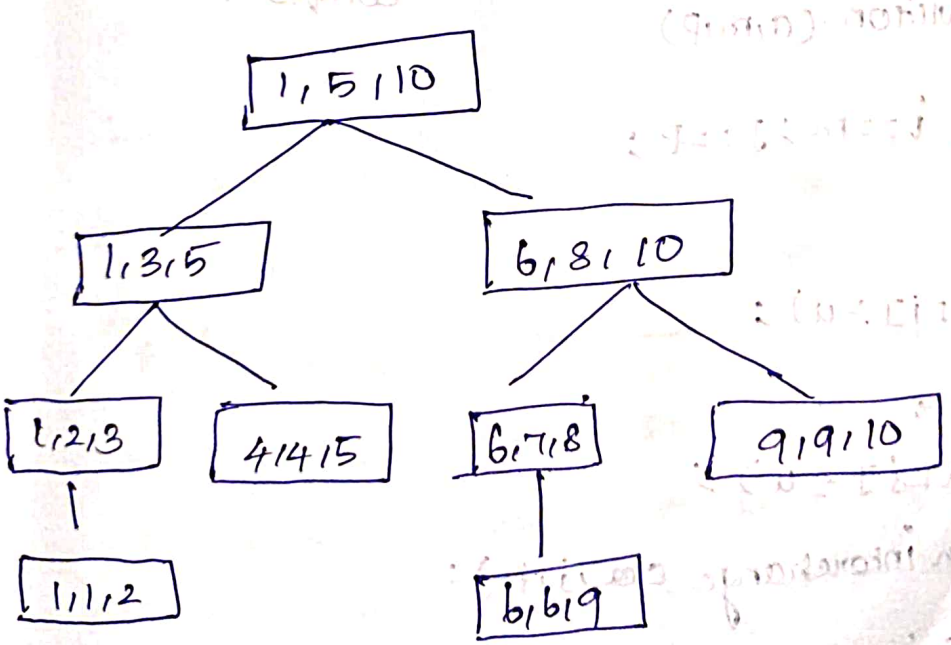
(10, 15, 25, 60) (28, 50, 70, 90) (12, 40)

(10, 15, 25, 28, 50, 60, 70, 90) (12, 40)

(10, 12, 15, 25, 28, 40, 50, 60, 70, 90).



1.1. Merge sort.



1.2 Merge.

$$T(n) = \begin{cases} a \text{ (constant)} & n = 1 \\ 2T(n/2) + cn \text{ (constant)} & n > 1 \end{cases}$$

$\therefore n = 2^k$

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

$= 2(2T(n/4) + cn/2) + cn$

$= 4T(n/4) + 2cn$

$= 2^3(2T(n/8) + cn/4) + 2cn$

$= 8T(n/8) + 3cn$

$= 16T(n/16) + 4cn.$

$= 32T(n/32) + 5cn.$

$\therefore n = 2^k = 2^5$

$= 2^k T(1) + kcn.$

$\boxed{T(n) = an + kcn.}$ $\therefore$ take both side o n log.

$T(n) = an + cn \log n$

$\boxed{T(n) = O(n \log n)}$

DAA

11/10/22

Technic :- divide And conque.

```
Algorithm Partition (a,m,p)
{
  u:=a[m]; i:=m; j:=p;
  Repeat
     i:=i+1;
     until (a[i] ≥ u);
  Repeat
     j:=j-1;
     until a[j] ≤ u);
  if (i<j) then interchange (a,i,n);
  }
  until (i ≥ j);
a[m] := a[j]; a[i]:= u;
  Return i;
}
```

Algorithm interchange (a, i, m), h)

{ p := a[i];
a[i] := a[j];
a[j] := p;
}

Algorithm QuickSort (p, q)

{ if (p < q) then
{ j := Partition (a, p, q+1)
QuickSort (p, j-1);
QuickSort (j+1, q);
}
}

C code theory.

Two types of sorting:

⇒ Internal sorting.
⇒ External sorting.

⇒ Similar to merge sort. But divide un equal.

⇒ when the element got greaterthan then interchange.

Interchange.

Ex:

44, 33, 11, 55, 77, 90, 40, 60, 99, 22, 88, 66.

22, 33, 11, 55, 77, 90, 40, 60, 99, 44, 88, 66.

22, 33, 11, 44, 77, 90, 40, 60, 99, 55, 88, 66.

22 33 11 40 77 90 44 60 99 55 88 66.

22 33 11 40 44 90 77 60 99 55 88 66 J=5

Sub array

Sub array.

11 33 22 40 44 90 77 60 99 55 88 66.

11 22 33 40 44 90 77 60 99 55 88 66.

11 22 33 40 44 66 77 60 99 55 88 90.

11 22 33 40 44 66 77 60 90 55 88 99

11 22 33 40 44 66, 77, 60, 88, 55 90 99

11 22 33 40 44 55, 77, 60, 88, 66 90 99

11 22 33 40 44 55, 66, 60, 88, 77 90 99.

11 22 33 40 44 55, 60, 66, 88, 77 90 99

11 22 33 40 44 55 60 66 77 88 90 99.

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

Strassen's Matrix Multiplication

$$C(i,j) = \sum_{1 \le k \le n} A(i,k)\, B(k,j)$$

- 2×2 matrix
- n^3 → multiplication
- n^2 – Addition

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

General method.

then

$C_{11} = A_{11} B_{11} + A_{12} B_{21}$

$C_{12} = A_{11} B_{12} + A_{12} B_{22}$

$C_{21} = A_{21} B_{11} + A_{22} B_{11}$

$C_{22} = A_{21} B_{12} + A_{22} B_{22}$

divide & conquer method = n is equal to 2^n

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

divide & conquer method time complexity.

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

18 - Sub & Addition.
7. Mul.

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

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

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

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

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

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

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

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

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

$C_{12} = R + T$

$C_{21} = Q + S$

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

$T = O(n^{2.81})$ — Time complexity for strassen's matrix multipication.

Greedy method :- * (most straight forwordion to solve any problem). * (maximize the product and minimize the cost)

Algorithm Greedy (a, n) — a is array with n inputs.

* Fesible sol → Sol one but we handle the different type method solution.

Optimal sol → Sol not give most values from the feasible solutions,

* n inputs & subset.

```
{
  Solution := φ;   (empty)
  for i := 1 to n do
  {
    x := Select(a);   (function)
    if Feasible (solution, x) then
      Solution := union (solution, x);
  }
  return solution;
}
```

DDA

Knapsack Problem :- (n object in a Bag – knapsack).

Maximize $\sum_{1 \le i \le n} P_i x_i$.

Subject to $\sum_{1 \le i \le n} w_i x_i \le m$ → capacity of knapsack

and $0 \le x_i \le 1, 1 \le i \le n$.

$n = 3, m = 20, (P_1, P_2, P_3) = (25, 24, 15)$ $(w_1, w_2, w_3) = (18, 15, 10)$

(x_1, x_2, x_3)	$\sum w_i x_i$	$\sum P_i x_i$
$(1/2, 1/3, 1/4)$	16.5	24.25
$(1, 2/15, 0)$	20	28.2
$(0, 2/3, 1)$	20	31
$(0, 1, 1/2)$	20	31.5

Algorithm Greedy knapsack (m, n) (capacity of knapsack, no. of objects)

{

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];

}.

① Increasing order (weight)

② Profit Increasing, Decreasing order (Profit)

③ p_i/w_i

⇒ Profit order ② method

$x_1 = 1$, $x_2 = 2/15$, $x_3 = 0$.

$x = 1$ 25

$x = 2/15$ $\frac{2}{15} \times 24 = \frac{16}{5} = 3.2$.

⇒ weight order. ① st method

$x_3 = 1$, $x_2 = 2/3$, $x_1 = 0$. ⇒ 20.

x_3 24×2 & 15×1 ⇒ 31

⇒ First obj. (feasible solution)

① $x_1 = 1, x_2 = 0, x_3 = 0$.   ② $x_1 = 0, x_2 = 1, x_3 = 0$.   ③ $x_1 = 0, x_2 = 0, x_3 = 1$

$x_1 = 25$ $x_2 = 24$ $x_3 = 15$

④ $x_1 = 1, x_2 = 0, x_3 = 2/10$.   ⑤ $x_1 = 5/9, x_2 = 0, x_3 = 1$

$x_1 = 25$

$x_2 = 0$

$x_3 = 15 \times \frac{2}{10} = 3 \times 2$

= 28

$x_3 = 25 \times 5$

$x_3 = \frac{125}{9}$ = 13.?

6) $x_1 = 1/2$, $x_2 = 1/3$, $x_3 = 1/2$.

x_1 ⇒ 12.5

x_2 ⇒ 8

x_3 : $\frac{7.5}{28}$

③ method ⇒ $\frac{p_i}{w_i}$ alias and decending order.

$\frac{25}{18}, \frac{24}{15}, \frac{15}{10}$ ⇒ 1.5, 1.6, 1.4

$x_1 = 0$ 0

$x_2 = 1$ 24

$x_3 = 1/2$ 7.5

31.5

⇒ 1.6, 1.5, 1.4

↳ L_2 & L_3 is we have place then fill it.

$\frac{p_i}{w_i} \geq \frac{p_{i+1}}{w_{i+1}}$

24 + 7.5 = 31.5

DAA - 14-Oct.

n=7, m=15.

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 5, 15, 7, 6, 18, 3)$

$(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (2, 3, 5, 7, 1, 4, 1)$

Sol:

	10	5	15	7	6	18	3
	2	3	5	7	1	4	1
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

(1) weight in ascending order.

m-w

P	w	m	
6	1	15-1 = 14.	$\Rightarrow x_5 = 1$.
3	1	14-1 = 13	$\Rightarrow x_7 = 1$
10	2	13-2 = 11	$\Rightarrow x_1 = 1$.
5	3	11-3 = 8	$\Rightarrow x_3 = 1$.
18	4	8-4 = 4	$\Rightarrow x_6 = 1$
15	5	4-5 = 0 X	$\Rightarrow x_3 = \frac{4}{5}$
7	7	0	$\Rightarrow x_4 = 0$.

w
2
3
5
7
1
4

	$P_i x_i$	$w_i x_i$
$x_1 = 1$	$1 \times 10 = 10$	$2 \times 1 = 2$
$x_2 = 1$	$1 \times 5 = 5$	$3 \times 1 = 3$
$x_3 = 4/5$	$\frac{4}{\cancel{5}_1} \times \cancel{15}^{3} = 12$	$4/\cancel{5} \times \cancel{5} = 4$
$x_4 = 0$	$0 \times 7 = 0$	$0 \times 7 = 0$
$x_5 = 1$	$1 \times 6 = 6$	$1 \times 1 = 1$
$x_6 = 1$	$1 \times 18 = 18$	$1 \times 4 = 4$
$x_7 = 1$	$1 \times 3 = 3$	$1 \times 1 = 1$
	$\sum p_i w_i = 54$	$\sum w_i x_i = 15$

(ii) Profit wise method :

	$P_i x_i$	$w_i x_i$
$x_1 = 1$	$1 \times 10 = 10$	$2 \times 1 = 2$
$x_2 = 0$	0	0
$x_3 = 1$	$1 \times 15 = 15$	$1 \times 5 = 5$
$x_4 = 4/7$	$4/7 \times 7 = 4$	$4/7 \times 7 = 4$
$x_5 = 0$	0	0
$x_6 = 1$	$1 \times 18 = 18$	$1 \times 4 = 0$
$x_7 = 0$	0	0

	P	w	m = W
x_6	18	4	$\Rightarrow 15 - 4 = 11$
x_3	15	5	$\Rightarrow 11 - 5 = 6$
x_1	10	2	$\Rightarrow 6 - 2 = 4$
x_4	7	7	$= 4/7 = 0.6/7$
x_5	6	1	0
x_2	5	3	0
x_7	3	1	0

(iii) P_i/w_i method :

P_i	10	5	15	7	6	18	3
w_i	2	3	5	7	1	4	1
P_i/w_i	5	1.6	3	1	6	4.5	3

(x_i) (1 ① 1 4/5 0 0 0)
x_1 x_2 x_3 x_4 x_5 x_6 x_7

P (7, 5, 3, 15, 18, 10, 6)

w (-1, 3, 1, 5, 4, 2, 1)

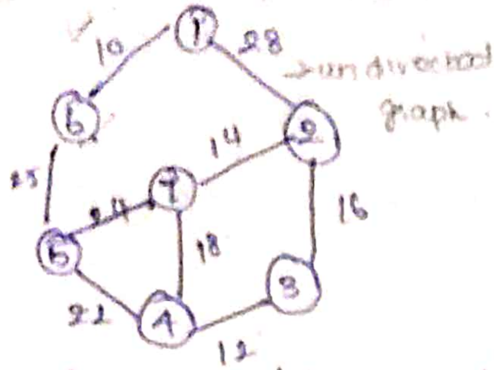
x_1 15 − 7 = 8
x_2 8 − 3 = 5
x_3 5 − 1 = 4
x_4 = 4/5
x_5 0

	$p_i x_i$	$w_i x_i$
$x_1 = 1$	$1 \times 7 = 7$	$1 \times 7 = 7$
$x_2 = 1$	$1 \times 5 = 5$	$1 \times 3 = 3$
$x_3 = 1$	$1 \times 3 = 3$	$1 \times 1 = 1$
$x_4 = 1/5$	$1/5 \times 15 = 12$	$1/5 \times 5 = 4$
$x_5 = 0$	0	0
$x_6 = 0$	0	0
$x_7 = 0$	0	0
	$\sum p_i x_i = 27$	$\sum w_i x_i = 15$

Min-cost Spanning tree prim's Algorithm :-

```
Algorithm Prim (E, cost, n, t)   // E → Set of edge in given graph; t → Spanning tree of given graph; n → number of vertices
{
 let (k,l) be an edge of minimum cost in E;
 mincost := cost [k,l];
 t[1,1] := k; t[1,2] = l;
 for i = 1 to n do.   // cost → boolean adjacency matrix of given graph.
  if (cost[i,1] < cost[i,k]) then.
      near[i] = l;
  else
    near[i] := k;
 near[k] := near[i] := 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,j] = j;
  t[i:2] := near[j];
  min cost := min cost + cost[j, near[j]];
  near[j] := 0;
  for k := 1 to do
    for ((near[k] ≠ 0) and (cost[k, near[k]] >
 } then near[k] := j; cost[k,j])
  return min cost;
```



g - edge t [1:n-1, 1:2]

7 - vertices = n

cost

	1	2	3	4	5	6	7
1	∞	28	∞	∞	∞	10	∞
2	28	∞	16	∞	∞	∞	14
3	∞	16	∞	12	∞	∞	∞
4	∞	∞	12	∞	22	∞	18
5	∞	∞	∞	22	∞	25	24
6	10	∞	∞	∞	25	∞	∞
7	∞	14	∞	18	24	∞	∞

let k = 1, l = 6

(1) adjuncy (6)

[2 6 / 28 10] [1 5 / 10 25]

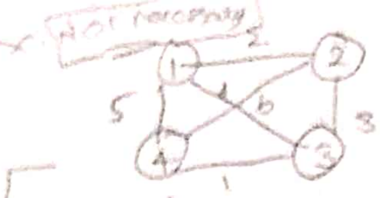
which is minimum distance that can be j so that,
j = 25 → connect anna வருதை

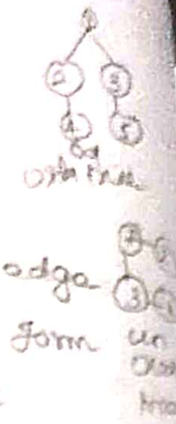
(5) [4 6 7 / 22 25 24] j = 22

(4) [3 5 7 / 12 22 18] j = 12

(3) [2 4 / 16 12] j = 16

(2) [1 3 7 / 25 16 14]

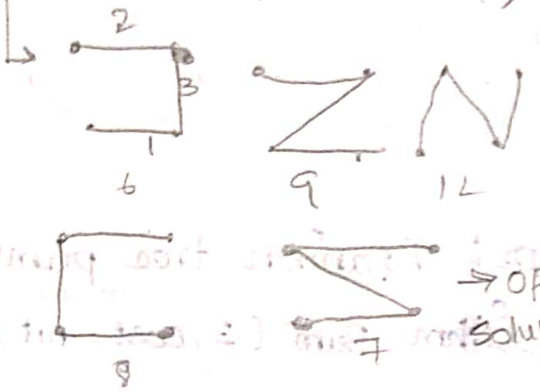




G = <V, E>

Spanning tree -ல ஒரு edge connect பண்ணுனா cycle form ஆகும். 4 node edge இருக்கணும்
n-1 edge

T is subset of G (T ⊆ G)



minimum spanning tree
(Kruskal)

Graph - n vertices இருந்தால் n-1 edge இருக்கும்.

(1) 10 t [1,1] = 1, t [1,2] = 6

(2) 10+25 = 35 ⇒ t [2,1] = 6, t [2,2] = 5

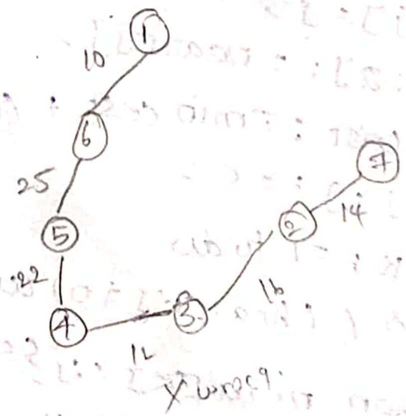
(3) = 57 ⇒ t [3,1] = 5, t [3,2] = 4

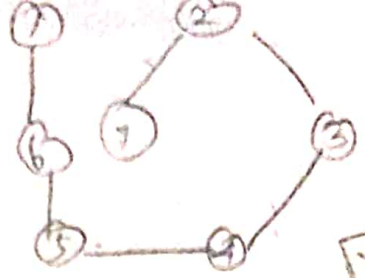
(4) = 57+12 = 69 ⇒ t [4,1] = 4, t [4,2] = 3

(5) = 69+16 = 85 t [5,1] = 3, t [5,2] = 2

(6) = 85+14 = 99 t [6,1] = 2, t [6,2] = 7

min-cost = 99





7 - edge node vertex

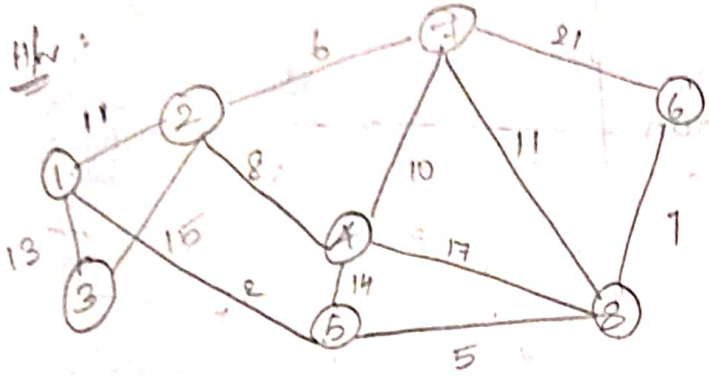
6 - edge.

V = 7

een - 1

Pg.No : 229. HW

HW :



n(vertices) = 8.

edges = 18.

t[1:n-1, 1:2]

cost:	1	2	3	4	5	6	7	8
1	∞	11	13	∞	2	∞	∞	∞
2	11	∞	15	8	∞	∞	6	∞
3	13	15	∞	∞	∞	∞	∞	∞
4	∞	8	∞	∞	14	∞	10	17
5	2	∞	∞	14	∞	∞	∞	5
6	∞	∞	∞	∞	∞	∞	21	7
7	∞	6	∞	10	∞	21	∞	11
8	∞	∞	∞	17	5	7	11	∞

let K = 1, l = 5.

①			⑤	
2	3	5	4	8
11	13	7	14	5

i = 2 j = 5

① t[1,1] = 1, t[2,1] = 5

②

Kruskal's Algorithm :-

t = ϕ.

while ((#t has less than n-1 edgs) and (E ≠ ϕ)) do

{

choose an edge (v,w) from E of leaost cost ;

Delete (v,w) from E;

if (v,w) does not create a cycle in t

then add(v,w) to t ;

else

discard(v,w);

}

Set of Edges.

Algorithm Kruskal (E, cost, n, t).

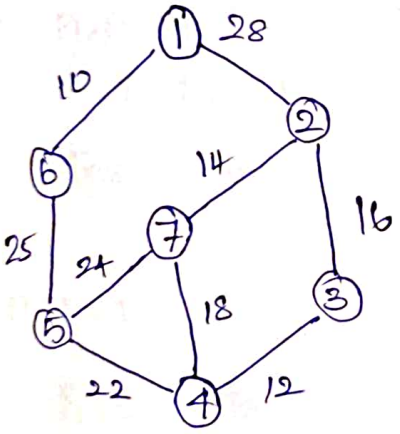
{

Construct a heap out of the edge costs using Heapify ;

for i = 1 to n do.

Parent [i] = -1;

```
i := 0;
mincost := 0.0;
while ((i < n-1) and (heap not empty)) do
{
  Delete minimum cost edge (u,v) from the heap
  and reheapify using Adjust;
    j := Find (u);
    k := Find (v);
    if (j ≠ k) then
    {
      i := i+1;
      t[i,1] := u
      t[i,2] := v
      mincost := mincost + cost[u,v];
      Union(j, k);
    }
}
if (i ≠ n-1) then write ("No Spanning tree");
else
  return mincost;
}
```



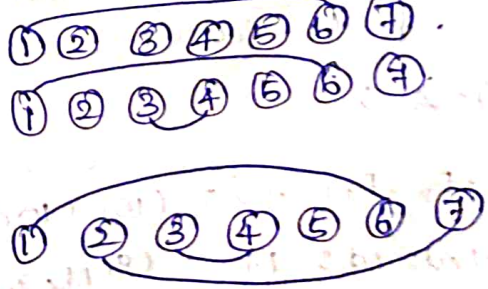
→ weighted and undirected graph.

(a)

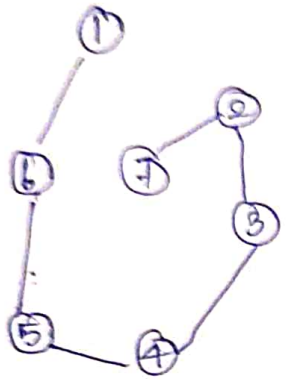
Edge	Cost	A/R
(1,6)	10	Accept
(3,4)	12	Accept
(2,7)	14	Accept

Spanning forest:

1 2 3 4 5 6 7



Edge	cost	A/R	Spanning forest
(3,5)	11	Accept	① ② ③ ④ ⑤ ⑥ ⑦
(4,7)	18	Reject	① ② ③ ④ ⑤ ⑥ ⑦
(1,5)	22	Accept	① ② ③ ④ ⑤ ⑥ ⑦
(5,7)	24	Reject	① ② ③ ④ ⑤ ⑥ ⑦
(5,6)	26	Accept	① ② ③ ④ ⑤ ⑥ ⑦



t[1,1] = 6 (n-1) steps

t[1,2] = 6.

E log n

Job sequencing with Deadlines :-

Algorithm Greedy_Job (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};
}
}.
```

Job - can be divided the Job as multiple no of Process each Process can be divide different one.

n = 4 ;

$(P_1, P_2, P_3, P_4) = (100, 10, 15, 23)$

$(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$

Schedule a profit on same time
Solution.

$1 \le i \le n$
$d_i > 0$
$p_i > 0$.

Feasible Sol	Processing Sequence	value
(1,3)	(1,3)	115
(2,1)	(2,1)	110
(2,3)	(2,3)	25
(3,1)	(3,1)	115
(4,1)	(4,1)	127
(4,3)	(4,3)	42
(2)	(2)	100
(1)	(1)	100
(3)	(3)	15
(4)	(4)	27

$n=5$

1) 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

Feasible Sol	Processing Sequence	value

2) $n=7$

$P_1\ P_2\ P_3\ P_4\ P_5, P_6, P_7$: 3 5 20 18 1 6 30.

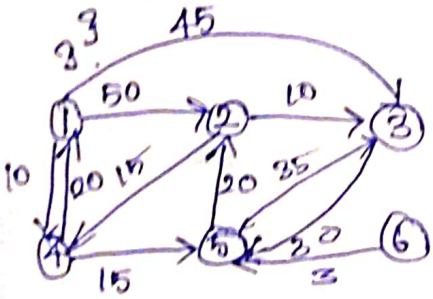
$d_1\ d_2\ d_3\ d_4\ d_5$: 1 3 4 3 2 1 2
d_6, d_7

DDA

Single Source shortest Path (or) Dijkstra's Algorithm.

Algorithm shortest path (v, cost, dist, n) → source vertex.

```
{ for i:=1 to n do
  {
   S[i]:= false ;
   dist[i] := cost(v,i)
  }
S[v] := true ; dist[v] := 0.0
for num := 0 to n-1 do
 {
  choose u from among those vertices not in S
  such that dist[u] is minimum ;
    S[u] := true ;
  for (each w adjacent to u with S[w] = false)
   if (dist[w] > dist[u] - cost[u,w]) ) then .
    dist[w] = dist[u] + cost[u,w] ;
```



→ undirected graph.
→ Not connected as sync, designation vertices.

let 1 is source vertices, so that

V = 1.

	1	2	3	4	5	6
1	0	50	45	10	∞	∞
2	∞	0	10	15	∞	∞
3	∞	∞	0	∞	30	∞
4	20	∞	∞	0	15	∞
5	∞	20	35	∞	0	∞
6	∞	∞	∞	∞	3	0

→ shortest path is 0.

Dist (0 50 45 10 ∞ ∞)
45 25

We consider source = 1. so that we take 1 row of matrix.

S (0 0 0 0 0 0)
1 1 → S[4] is true so, that we make it

Choose u from among those vertices not in S such that dist[u] is minimum, so that u = 4.

Each adjacent row with s(w).

w = 1, 5.

∞ > 10+15.

num = 3.

u = 5.

S[5] = true,

w = 2, 3

d(2) = 50,

① 50 > 25 + 20

d(2) = 45

num = 4.

u = 2.

s(2) = 1

w = 3, 4.

d(3) = 45 > 45 + 10

num = 5

v = 3

w = 5

it is one that we cant change.

Final col:

Dist	1	2	3	4	5	b
Dist	0	45	45	10	25	20
S	1	1	0	1	1	0

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

Optimal Storage on Tapes :-

Algorithm Store(n,m)

```
{
  j = 0;
  for i = 1 to n do.
  {
    write("append program", i, "to permutation
          for tape", j);
    j := (j+1) mod m ;
  }
}
```

There are n programs that store in a tape length.

$1 \le i \le n$. → l_i - [length of the Right program], clearly all program to store, sum of the program must be all.

l_i - retrieve from the ith program.

Let

$n=3$, $(l_1, l_2, l_3) = (5, 10, 3)$.

1, 2, 3.

Ordering (I)	length d(I)	
1, 2, 3	5+(10+5)+(5+(10+3) = 38	
1, 3, 2	5 + 8 + 18	= 31
2 1 3	10 + 15 + 18	= 43
2 3 1	10 + 13 + 18	= 41
3 1 2	3 + 8 + 18	= 29
3 2 1	3 + 13 + 15	= 24.

Feasible solution

minimum retrieve time

$l_1 \le l_2 \le l_3 \ldots$.

II

n-Program m-tape.

6, 10, 4, 5, 7, 25, 30, 15, 20, 8.

4, 5, 6, 7, 8, 10, 15, 20, 25, 30.

tape 1 $4 + 31 + 26 + 56$. (4+7) (11+15) $= 97$

tape 2 $5 + 13 + 33$ $= 51$

tape 3 $6 + 16 + 41$ $= 63$

$$\text{Avg. time} = \frac{211}{3} = 70.33$$

Multistage graphs

Algorithm FGraph (G, k, n, P)

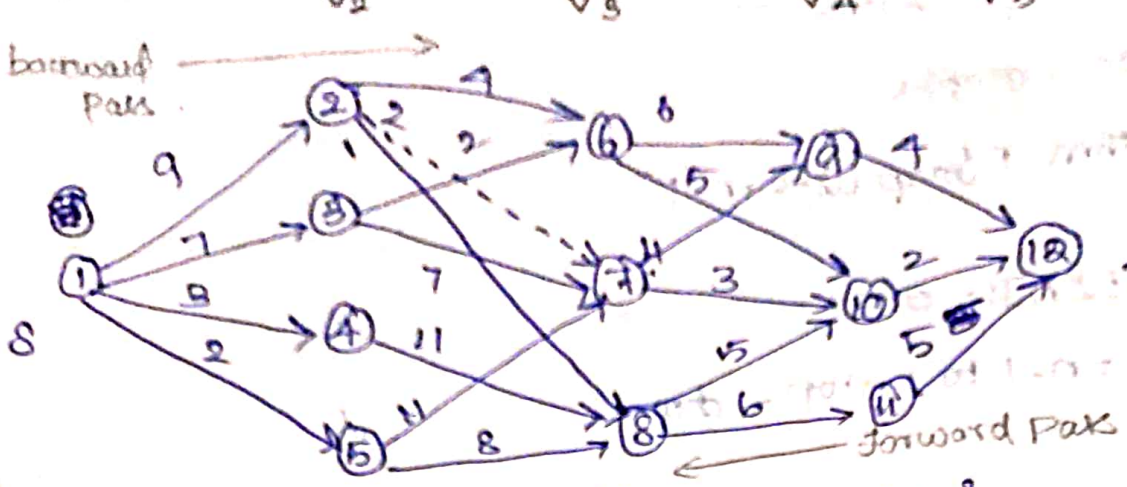
{
cost[n] := 0.0;
for j := n-1 to 1 step -1 do
{
let r be a vertex such that (j, r) is an edge of
G and c[j, r] + cost[r] is minimum;
cost[j] = c[j, r] + cost[r];
d[j] := r;
}
P[1] := 1, P[k] := n;
for j := 2 to k-1 do P[j] = d[P[j-1]];
}

Formula:-

$$bcost(i,j) = \min_{\substack{l \in n-1 \\ (i,l) \in E}} \{ bcost(i-1, l) + c(l, j) \}$$

Algorithm BGraph (G, k, n, P)

{
bcost[1] = 0.0;
for j = 2 to n do
{
let r be such that (r, j) is an edge of G and c[j, r]
+ bcost[r] + c[r, j] is minimum;
bcost[j] := bcost[r] + c[r, j];
d[j] := r;
}
P[1] := 1; P[k] := n;
for j := k-1 to 2 do P[j] := d[P[j+1]];



$$cost(i,j) = \min \{ c(j,l) + cost(i+1,l) \}$$

$l \in V_{i+1}$

$(j,l) \in E$

$|V_5| = 1$ $|V_3| = 3$ $|V_2| = 4$

$|V_4| = 3$ $|V_1| = 1$

$G = (V, E)$ $k \geq 2$.

k = Stage of Graph

$k = 5$

l = vertices

Formula :- (forward Pass).

$$\rightarrow cost(i,j) = \min \{ (j,l) + cost(i+1,l) \}$$

(1) $cost(k-1,j) = c(j,k)$ for $k = 5$

$cost(5,12) = 0$.

$cost(5-1,j) = c(j,k)$

$cost(4,9) = 4$

$cost(4,10) = 2$

$cost(4,11) = 5$.

(2)

$$cost(3,6) = \left\{ \begin{array}{l} c(6,9) + cost(4,9) \\ 6 + 4 = 10 \\ c(6,10) + cost(4,10) \\ 5 + 2 = 7 \end{array} \right\} = 7.$$

$$cost(3,7) = \left\{ \begin{array}{l} c(7,9) + cost(4,9) \\ 4 + 4 = 8 \\ cost(7,10) + cost(4,10) \\ 3 + 2 = 5 \end{array} \right\} = 5.$$

$$cost(3,8) = \left\{ \begin{array}{c} c(8,10) + cost(4,10) \\ 5+2=7 \\ c(8,11) + cost(4,11) \\ 6+5=11 \end{array} \right\} = 7.$$

Choose, vertices.

$$cost(2,2) = \left\{ \begin{array}{l} c(2,6) + cost(3,6) \quad cost(3,7) \quad c(2,8) \\ + cost(3,8) \quad 2 \quad + 5 \\ 4+7=11 \quad 7 \quad 1+7 \\ 8. \end{array} \right\}$$

$$= 7.$$

$$cost(2,3) = \left\{ \begin{array}{l} c(3,6) + cost(3,6), \; c(3,7) + cost(3,7) \\ 2+7=9 \quad 7+5=11 \end{array} \right\} = 9$$

$$cost(2,4) = \left\{ \begin{array}{c} c(4,8) + cost(3,8) \\ 11+7 \end{array} \right\} = 18.$$

$$cost(2,5) = \left\{ \begin{array}{l} c(5,7) + cost(3,7), \; c(5,8) + cost(3,8) \\ 11+5=16 \quad 8+7 \end{array} \right\} = 15$$

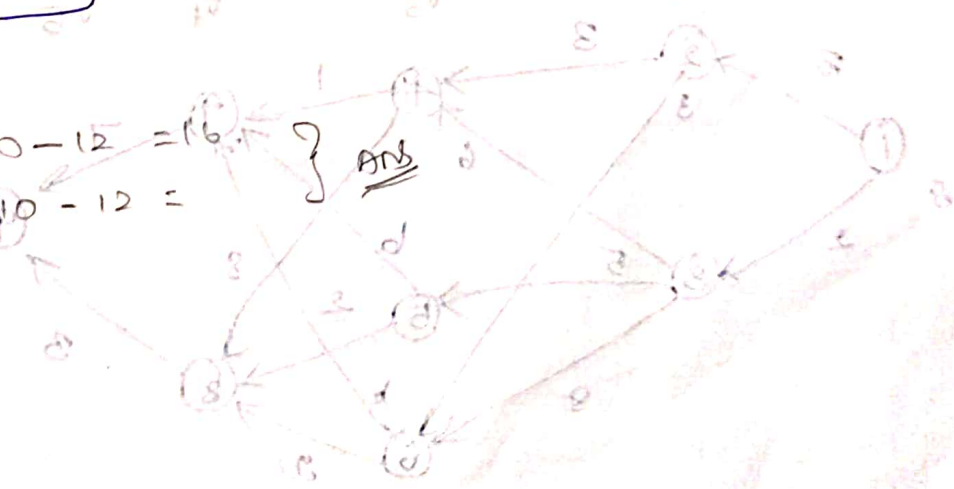
$$cost(1,1) = \left\{ \begin{array}{l} c(1,2) + cost(2,8), \; c(1,3) + cost(2,3), \\ 9+7=16 \quad 7+9=16 \\ 2 \\ c(1,4) + cost(2,4), \; c(1,5) + cost(2,5) \\ 3+18=21 \quad 2+15=17 \end{array} \right\}$$

$\boxed{cost(1,1) = 16.}$

Sol: (Algo)

1 – 2 – 7 – 10 – 12 = 16 } Ans

1 – 3 – 6 – 10 – 12 = }



Backward path:-

Bcost (8,8) = 9

Bcost (2,3) = 7

Bcost (8,4) = 3

Bcost (8,5) = 2

$$Bcost(3,6) = \left\{ \begin{array}{l} Bcost(8,8) + c(3,6) \\ 9+4 = 13 \\ Bcost(8,3) + c(3,6) \\ 7+2 = 9 \end{array} \right\} = 9.$$

$$Bcost(3,7) = \left\{ \begin{array}{l} Bcost(8,2) + c(2,7) \\ 9+2 = 11 \\ Bcost(2,3) + c(3,7) \\ 7+4 = 11 \end{array} \right\} = 11.$$

$$Bcost(8,8) = \{10, 14, 10\} = 10.$$

Bcost (4,9) = {15, 15} = 15

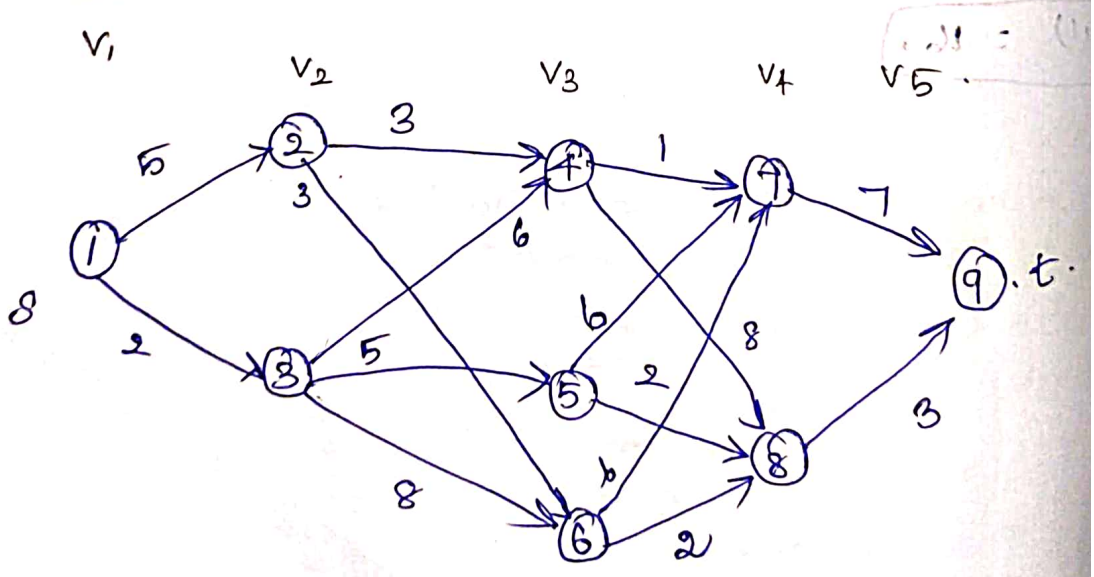
Bcost (4,10) = {14, 14, 15} = 14

Bcost (4,11) = {10+6} = 16.

Bcost (5,12) = 16 → min

Bcost (5,12) : 16

Sum :-



Forward pass :

cost (5,9) = 0

cost (4,7) = 7

cost (4,8) = 3

Flow shop scheduling :- $8m//$ (Multiple Job multiple Process)

Job

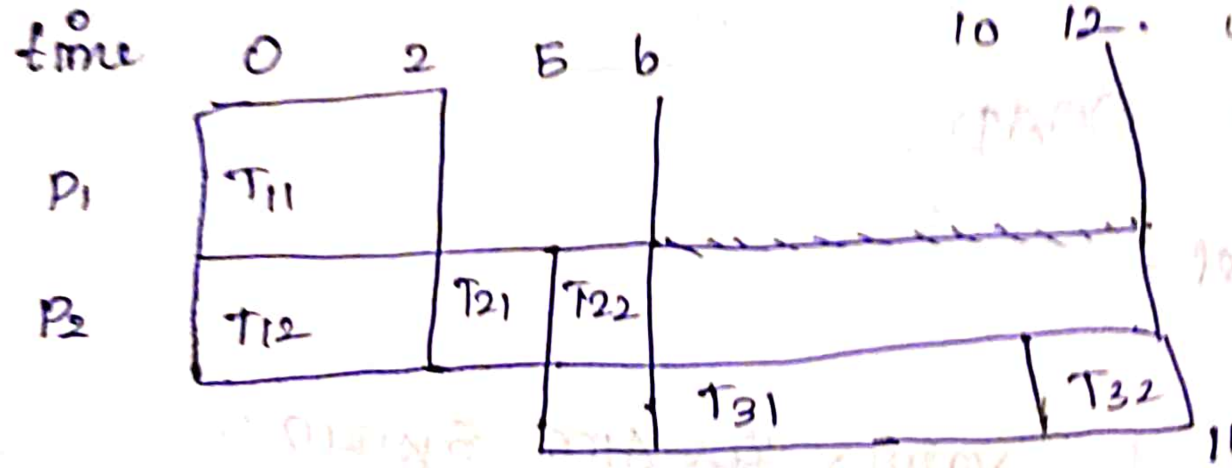
$$j = \begin{bmatrix} 2 & 0 \\ 8 & 3 \\ 5 & 2 \end{bmatrix} \text{ task.}$$

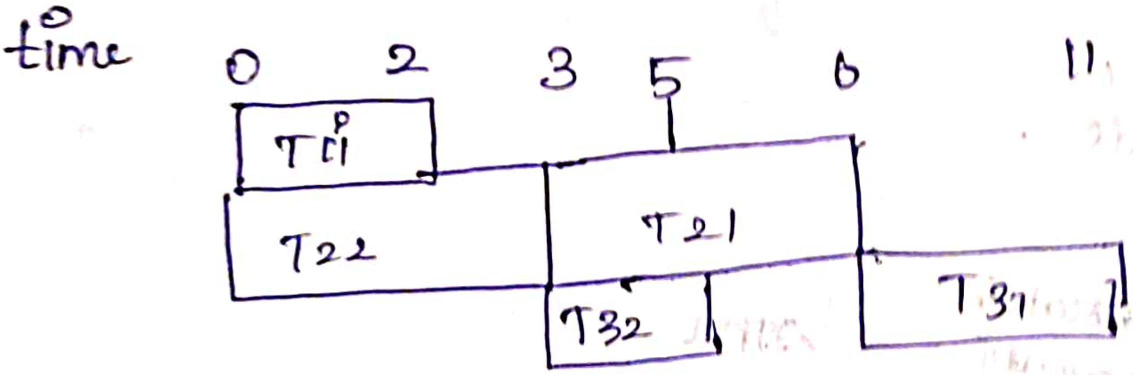
Job 1 = 10.

Job 2 = 12.

waiting J_2 = 1.

waiting J_1 = 0





Job 1 = 11.

waiting J_1 = 0.

Job 2 = 10.

waiting J_2 = 1

$$F(s) = \max_{1 \le i \le n} \{ t_i(s)^2 \}.$$

$f_1(s) = 10$, $f_1(s) = 11$

mean flow time.

$$MFT(s) = \frac{1}{n} \sum_{1 \le i \le n} f_i(s).$$

$f_2(s) = 12$, $f_2(s) =$

i = Job number. $= n$ - of Job require -

j : Job task $= m$ - task.

T_i : P_j → Processor.

↳ time required to complete task — E_{ij}.

OFT - Optimal finishing Time.

OMFT - Optimal mean finishing time.

POMFT - Premetive optimial mean flow time.