DAA HOME ASSIGNMENT-1

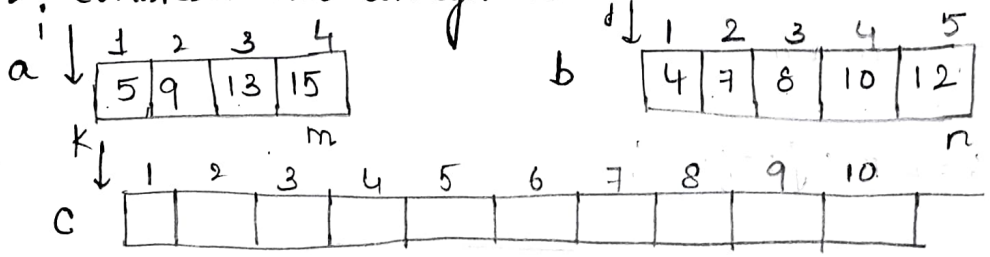
1. Explain the Divide-and-conquer strategy in which the division into two sub arrays is made so that the sorted sub arrays need to be merged later.

1. Divide and Conquer Strategy computes the Solution of the problem by computing the solution to the sub-problems and Unifying the Solutions of the sub problems as the solution of the given problem.
2. Solution of the sub-problem is computed in such a similar way till the base case is reached.
3. Merge Sort is an example.

Merging of the sorted sub arrays:

1. Combining of two sorted lists into one sorted list is called as merging.
2. Initialize an index variable to the index of the first element of each of the list.
3. Initialize an index variable to the index of the first element of the initially empty merged list.

Ex:- Consider two arrays a and b



1. Compare a[i] and b[i] and the smallest is copied into the current index location k of the merged list.
2. Increment the source and destination indices of the copy operation

3. c[k] := a[i]; i := i+1 & c[k] := b[j]; j := j+1;
4. k := k+1;

c

1	2	3	4	5	6	7	8	9
4	5	7	8	9	10	12	13	15

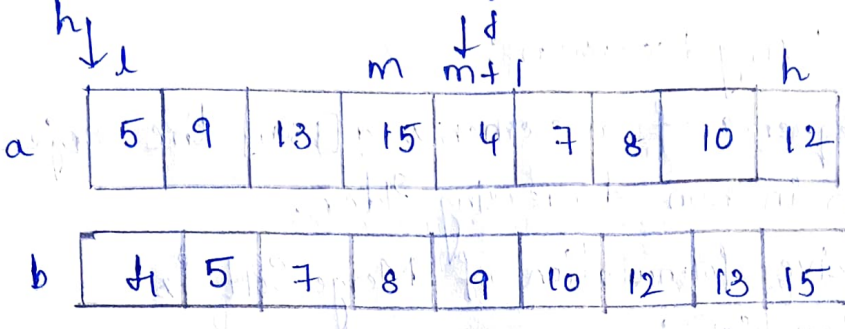
5. When we reach the end of one of the lists, elements are copied from the other list, till the end of the Source list is reached.
6. Copy the elements from i to m & j to n.

```
i:=1 ; j:=1; k:=1;
while (i<=m and j<=n) do
{
   if (a[i]<=b[j]) then
   {
      c[k]:=a[i]; i:=i+1;
   }
   else
   {
      c[k]:=b[j]; j:=j+1;
   }
   k:=k+1;
}
for p:=i to m do
{
   c[k]:=a[p]; k:=k+1;
}
for p:=j to n do
{
   c[k]:=b[p]; j:=j+1;
}
```

Merging in Merge Sat:

1. Merging in Merge Sat has a Small Variation.
2. The two lists to be merged are logical partitions in the Same array.
3. A temporary array is used to store the merged lists of the two logical partitions.
4. The merged list from the temporary array is copied back

a




```
Algorithm merge(low, mid, high)
// a[low:mid] and a[mid+1:high] are two sorted logical
// partitions of global array a
// These two partitions are merged to array b[low:high]
// These elements are copied from b[low:high] to a[low: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;
  }
  for k:=h to mid do
  {
    b[i]:=a[k]; i:=i+1;
  }
  for k:=j to high do
  {
    b[i]:=a[k]; i:=i+1;
  }
  for k:=low to high do
  {
    a[k]:=b[k];
  }
}
```

Merge Sort Using Recursion:

1. Given a Sequence of n elements a[1:n] the objective is to sort them in non-decreasing order.
2. The recursive formulation of Merge Sort is
   MergeSort(low, high):
3. A list with one element is Merge Sorl

2. Explain the divide and Conquer Strategy in which the division into sub arrays is made so that the sorted sub arrays do not need to be merged later.

1. the divide and Conquer Strategy in which the division into Subarrays and do not need to be merged later is Quick Sort.
2. Partitioning is static in the case of Merge Sort algorithm the list is divided into two near halves.
3. In QuickSort, partitioning is dynamic and the size of the two lists are not be the same. It depends on the elements present in the list.

Algorithm:

```
Algorithm QuickSort(a,p,q)
{
    if(p<q) then
    {
      j:= partition(a,p,q)
      Quicksort(a,p,j-1);
      Quicksort(a,j+1,q);
    }
}
```

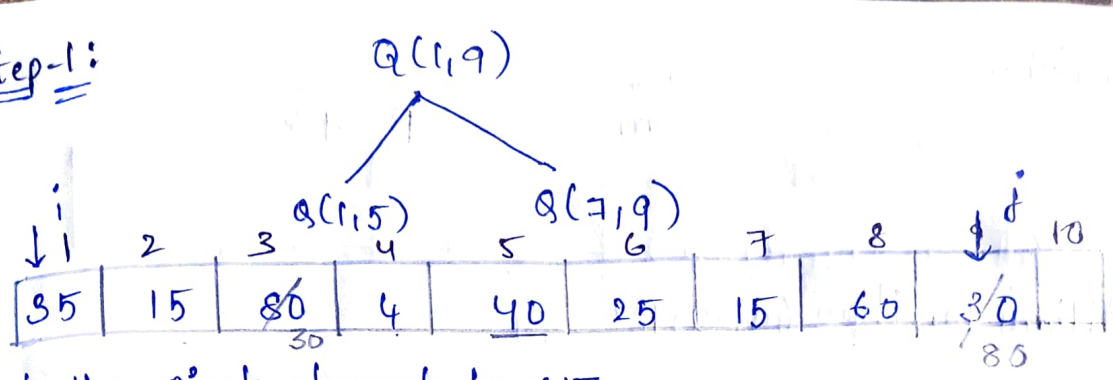
Example:

Quicksort(1,9)

Initially array be

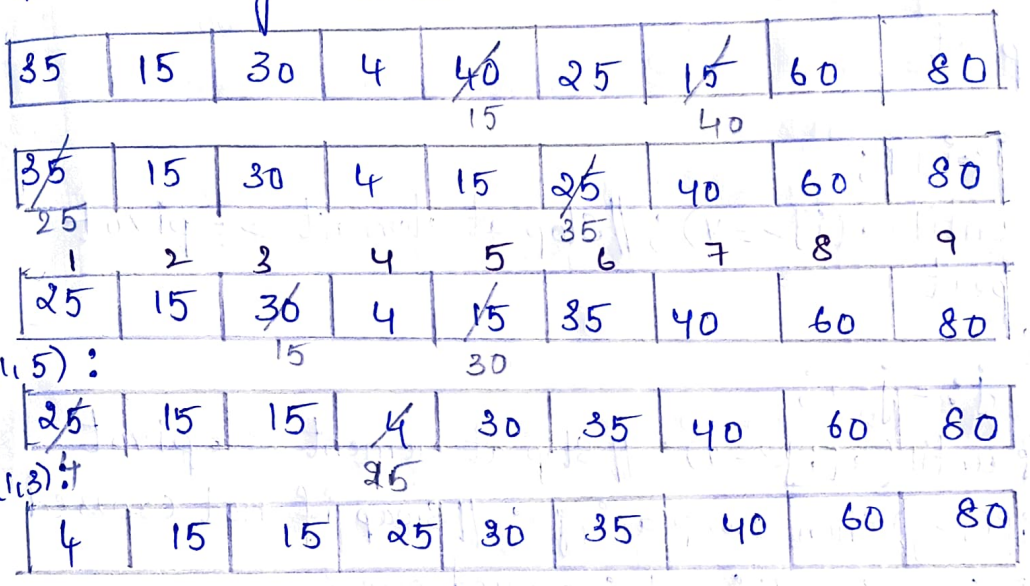
1	2	3	4	5	6	7	8	9
35	15	80	4	40	25	15	60	30

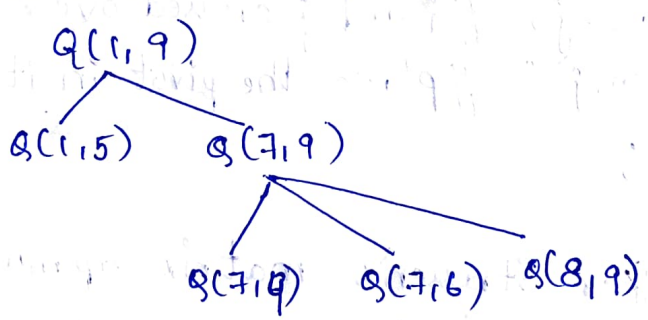
Step-1:



Let the pivot element be 35

See the element greater than 35 and lesser than 35 from the right side.



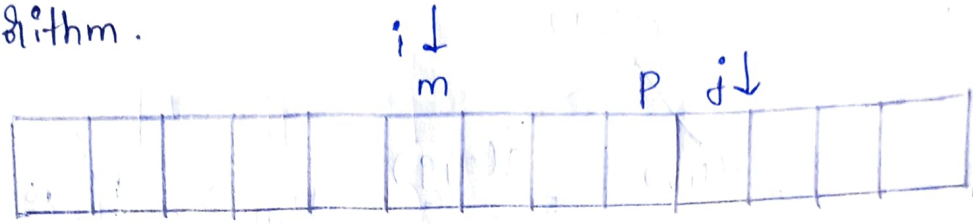



```
repeat
{
  i:i+1;
} until(a[i]>=v)
```

```
repeat
{
  j:j-1;
}. until(a[j]<=v)
```

```
if(i<j)then
  swap(a,i,j);
```

Algorithm.




```
Algorithm Partition (a, m, p)
{
   v := a[m] ; i := m; j := p+1;
   repeat
   {
      repeat
      {
         i := i+1;
      } until (a[i] >= v) ; // Stop at element >= pivot
      repeat
      {
         j := j-1;
      } until (a[i] <= v) ; // Stop at element <= pivot
      if(i<j) then swap(a, i, j) ; // Swap if not crossed
   }until (i >= j) // i and j crossed over
   swap (a, m, j) ; // Place the pivot in its position
   return j;
}
```

3. Discuss volker Strassen's matrix ~~ap~~ multiplication with the following example !

$$A = \begin{bmatrix} 2 & 3 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{bmatrix}$$

1. Given two matrices A and B.
2. Now, Compute AB of nxn matrices, we need 8 matrices multiplication of $\frac{n}{2} \times \frac{n}{2}$ matrices and 4 Sums of $\frac{n}{2} \times \frac{n}{2}$ matrices.
3. Sum of $\frac{n}{2} \times \frac{n}{2}$ matrices takes $\frac{n^2}{4}$ additions, 4 Sums take n^2 operations and hence is $\Theta(n^2)$.

A matrix can be Written as $\begin{bmatrix} 2 & 3 & 1 & 0 \\ 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

B matrix can be Written as $\begin{bmatrix} 1 & 2 & 3 & 0 \\ 3 & 2 & 1 & 0 \\ 2 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

these are 4x4 matrices.

Now, $\begin{bmatrix} \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 3 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 0 \\ 1 & 0 \end{bmatrix} \\ \begin{bmatrix} 2 & 1 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$

(A_{11}, A_{12}, B_{11}, B_{12}; A_{21}, A_{22}, B_{21}, B_{22})

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

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

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

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

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

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

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

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

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

$C_{12} = R + T$

$C_{21} = Q + S$

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

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

$$\left[\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right]\left[\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}\right]$$

$$= \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 4 & 2 \\ 3 & 2 \end{bmatrix}$$

$\rightarrow P = \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 4 & 2 \\ 3 & 2 \end{bmatrix}$

$P = (a_{11}+a_{22})(b_{11}+b_{22})$

$= (2+2)(4+2) = 4 \times 6 = 24$

$Q = (a_{21}+a_{22})b_{11}$

$= (1+2)(4) = 3 \times 4 = 12$

$R = a_{11}(b_{12}-b_{22})$

$= 2(2-2) = 2 \times 0 = 0$

$S = a_{22}(b_{21}-b_{11})$

$= 2(3-4) = 2 \times -1 = -2$

$T = (a_{11}+a_{12})b_{22}$

$= (2+3)(2) = 5 \times 2 = 10.$

$U = (a_{21}-a_{11})(b_{11}+b_{12})$

$= (1-2)(4+2) = -1 \times 6 = -6.$

$V = (a_{12}-a_{22})(b_{21}+b_{22})$

$= (3-2)(3+2) = (1)(5) = 5.$

$C_{11} = P+S-T+V = 24+(-2)-10+5$

$= 17.$

$C_{12} = R+T = 0+10 = 10$

$C_{21} = Q+S = 12+(-2) = 10$

$C_{22} = P+R-Q+U = 24+0-12+(-6)$

$= 6.$

$\therefore P = \begin{bmatrix} 17 & 10 \\ 10 & 6 \end{bmatrix}$

$\rightarrow Q = [A_{21}+A_{22}]B_{11}$

$= \left[\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}+\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right]\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$

$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$

$P = (a_{11} + a_{22})(b_{11} + b_{22})$

$= (0+0)(1+2)$

$= 0$

$Q = (a_{21} + a_{22})(b_{11}) = (0+0)(1)$

$= 0$

$R = a_{11}(b_{12} - b_{22}) = 0$

$S = a_{22}(b_{21} - b_{11}) = 0$

$T = (a_{11} + a_{12}) b_{22} = 0(2)$

$= 0$

$U = (a_{21} - a_{11}) b_{22} = 0$

$V = (a_{12} - a_{22})(b_{21} + b_{22}) = 0$

$C_{11} = P + S - T + V = 0 + 0 - 0 + 0$

$= 0$

$C_{12} = R + T = 0 + 0 = 0$

$C_{21} = Q + S = 1 + 0 = 0$

$C_{22} = P + R - Q + U = 0 + 0 - 0 + 0$

$= 0$

$\therefore Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

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

$= \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \left(\begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix} - \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix} \right)$

$R = \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 2 \end{bmatrix}$

$P = (a_{11} + a_{22})(b_{11} + b_{22})$

$= (2+2)(0+0) = 0$

$Q = (a_{21} + a_{22})(b_{11})$

$= (1+2)(0) = 3 \times 2$

$= 6 \quad 0$

$R = a_{11}(b_{12} - b_{22})$

$= 2(2-0) = 0.$

$S = a_{22}(b_{21} - b_{11})$

$= 2(3 - (-2))$ $= 2(1-0)$

$= 2 \times 5 = 10$. $= 2$

$T = (a_{11} + a_{12})(b_{22})$

$= (2+3)(0)$

$= 0$.

$U = (a_{21} - a_{11})(b_{11} + b_{12})$

$= (1-2)(1+0) = 0$

$V = (a_{12} - a_{22})(b_{21} + b_{22})$

$= (3 - 2)(1+0)$

$= 1 \times 5 = 5 = 1$

$C_{11} = P + S - T + V = 0 + 12 - 10 + 1$

$= 3$

$C_{12} = R + T = 0 + 10 = 10$

$C_{21} = Q + S = -6 + 10 = 4 \quad 0 + 2 = 2$

$C_{22} = P + R - Q + U = 0 + 0 + (-6) + 0$

$= 6$.

$\therefore R = \begin{bmatrix} 5 & 4 \\ 4 & 6 \end{bmatrix}$

$\therefore R = \begin{bmatrix} 3 & 0 \\ 2 & 0 \end{bmatrix}$

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

$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\left(\begin{bmatrix} 2 & 1 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}\right)$

$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ -3 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

$P = 0 \qquad U = 0$

$Q = 0 \qquad V = 0$

$R = 0 \qquad C_{11} = 0$

$S = 0 \qquad C_{12} = 0$

$C_{21} = 0$

$T = 0 \qquad C_{22} = 0$

$\therefore S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

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

$$\left(\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 3 & 0 \end{bmatrix}\right)\begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 3 & 3 \\ 4 & 2 \end{bmatrix}\begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}$$

$P = (a_{11} + a_{22})(b_{11} + b_{22})$

$= (3+2)(3+0) = (5)(3) = 15$

$Q = (a_{21} + a_{22})b_{11}$

$= (4+2)(3) = (6)(3) = 18$

$R = a_{11}(b_{12} - b_{22})$

$= 3(0-0) = 0$

$S = a_{22}(b_{21} - b_{11})$

$= 2(0-3) = -6$

$T = (a_{11} + a_{12})b_{22}$

$= (3+3)(0) = 0$

$U = (a_{21} - a_{11})(b_{11} + b_{12})$

$= (4-3)(3+0) = 1 \times 3 = 3$

$V = (a_{12} - a_{22})(b_{21} + b_{22})$

$= (3-2)(0+0) = 0$

$c_{11} = P + S - T + V = 15 - 6 - 0 + 0 = 9$

$c_{12} = 0 + 0 = 0$

$c_{21} = Q + S = 18 - 6 = 12$

$c_{22} = P + R - Q + U = 15 + 0 - 18 + 3 = 0$

$\therefore T = \begin{bmatrix} 9 & 0 \\ 12 & 0 \end{bmatrix}$

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

$$= \left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\right)\left(\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 0 \\ 1 & 0 \end{bmatrix}\right)$$

$$u = \begin{bmatrix} -2 & -3 \\ -1 & -2 \end{bmatrix}\begin{bmatrix} +4 & 2 \\ 4 & 2 \end{bmatrix}$$

$P = (a_{11} + a_{22})(b_{11} + b_{22})$

$= (-2-2)(4+2) = (-4)(6) = -24$

$Q = (a_{21} + a_{22}) b_{11} = (-1-2)(4) = (-3)(4) = -12$

$R = (a_{11})(b_{12} - b_{22}) = (-2)(2-2) = 0$

$S = a_{22}(b_{21} - b_{11}) = (-2)(4-4) = 0$

$T = (a_{11} + a_{12}) b_{22} = (-2-3)(2) = (-5)(2) = -10.$

$u = (a_{21} - a_{11})(b_{11} + b_{12}) = (-1+2)(4+2) = (1)(6) = 6.$

$V = (a_{12} - a_{22})(b_{21} + b_{22}) = (-3+2)(4+2) = (-1)(6) = -6.$

$c_{11} = P + S - T + V = -24 + 0 + 10 + 6 = -20$

$c_{12} = R + T = 0 - 10 = -10$

$c_{21} = Q + S = -12 + 0 = -12$

$c_{22} = P + R - Q + u = -24 + 0 + 12 + 6 = -6$

$$u = \begin{bmatrix} -20 & -10 \\ -12 & -6 \end{bmatrix}.$$

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

$$= \left(\begin{bmatrix} 1 & 0 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right)\left(\begin{bmatrix} 2 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 1 & 0 \\ 3 & 0 \end{bmatrix}\begin{bmatrix} 5 & 1 \\ 0 & 0 \end{bmatrix}$$

$P = (a_{11} + a_{22})(b_{11} + b_{22})$

$= (1+0)(5+0) = (1)(5) = 5.$

$Q = (a_{21} + a_{22}) b_{11} = (3+0)(5) = 15$

$R = (a_{11})(b_{12} - b_{22}) = (1)(1-0) = 1$

$S = a_{22}(b_{21} - b_{11}) = 0(0-5) = 0$

$T = (a_{11}+a_{12})b_{22} = (1+0)(0) = 0$

$U = (a_{21}-a_{11})(b_{11}+b_{12}) = (3-1)(5+1) = (2)(6) = 12$

$V = (a_{12}-a_{22})(b_{21}+b_{22}) = (0-0)(0+0) = 0$

$C_{11} = P+S-T+V = 5+0-0+0 = 5$

$C_{12} = R+T = 1+0 = 1$

$C_{21} = Q+S = 15+0 = 15$

$C_{22} = P+R-Q+U = 5+1-15+12 = 3$

$\therefore C = \begin{bmatrix} 5 & 1 \\ 15 & 3 \end{bmatrix}$

4. What is fractional knapsack problem ? Find the Solution to the fractional knapsack problem instance $n=4$, $m=15$, $(P_1, P_2, P_3, P_4) = (10, 10, 12, 18)$ and $(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$.

1. Given n objects and a knapsack. object i has weight w_i and profit p_i and the knapsack has a capacity m.
2. The objective is to maximize the total profit by selecting objects (fraction of an object) without exceeding the total capacity of the knapsack.
3. * maximize $\sum_{1 \le i \le n} p_i x_i$
   * Subject to $\sum_{1 \le i \le n} w_i x_i \le m$
   * $0 \le x_i \le 1$ and $1 \le i \le n$
   * A feasible Solution is any set $\{x_1, x_2, x_3, \ldots x_n\}$ which Satisfy (2) and (3)
   * An optimum Solution is a feasible Solution that satisfies (1)

i	1	2	3	4
P_i	10	10	12	18
w_i	2	4	6	9
P_i/w_i	5	2.5	2	2

Decreasing order of P_i/w_i ratio is

$i_1 > i_2 > i_3 > i_4$

$5 > 2.5 > 2 > 2$

$m = 15$

i	P_i	w_i	Rem Weight
1	10	2	15 − 2 = 13
2	10	4	13 − 4 = 9
4	18	9	9 − 9 = 0
3	12	6	0

The Solution vector is $(1, 1, 0, 1) = x_i$

Maximum profit is

$\sum P_i x_i = 10(1) + 10(1) + 18(1)$

$= 38.$

	1	2	3	4
x	1	1	0	1

Algorithm GreedyKnapSack(m, n)

// p[1..n] contains the profits and w[1..n] contains the weigh of n objects ordered such that p[i]/w[i] >= p[i+1]/w[i+1]. job[1..n] Contains the IDs of jobs in p and w. m is the capac // x[1..n] is the Solution Vector.

```
{ for i:=1 to n do x[i]:=0;
  u:=m;
  for i:=1 to n do
  { if (w[job[i]] > u) then break;
    x[job[i]]:=1.0; u:=u-w[job[i]];
    profit:=profit+p[job[i]];
  }
  if (i<=n) then
  { x[job[i]]:=u/w[job[i]];
    profit:=profit+p[job[i]]*u/w[job[i]];
  }
```

5. State the Job Sequencing with deadlines problem. Find Solution generated by job Sequencing problem with deadlines for 7 jobs given profits 3, 5, 20, 18, 1, 6, 30 and deadlines 1, 3, 4, 3, 2, 1, 2 respectively.

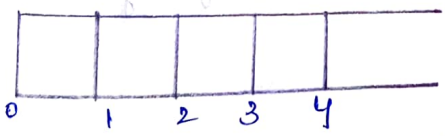
1. N jobs 1, 2, ..., n are given where each job needs 1 unit of time to Complete
2. Each job i has a profit P_i, which is awarded if the job can be Completed before its deadline d_i.
3. The objective is to earn maximum profit when only one job can be Scheduled & processed at any given time.

	1	2	3	4	5	6	7
p	3	5	20	18	1	6	30
d	1	3	4	3	2	1	2

Sort the job in descending order of their profit value

JOB	7	3	4	6	2	1	5
PROFIT	30	20	18	6	5	3	1
Deadline	2	4	3	1	3	1	2

Select maximum deadline $D_{max} = 4$



Select the job to be entered which has highest profit.

Step-1:

TS	1	2	3	4
	0	J_7	0	0

Profit = 30

Step-2:

TS	1	2	3	4
	0	J_7	~~J_0~~	~~J_3~~ J_3

Profit = 30 + 20 = ~~50~~ 50.

Step-3

TS

1	2	3	4
0	J_7	J_4	J_3

Profit = 50 + 18
= 68

TS

1	2	3	4
J_6	J_7	J_4	J_3

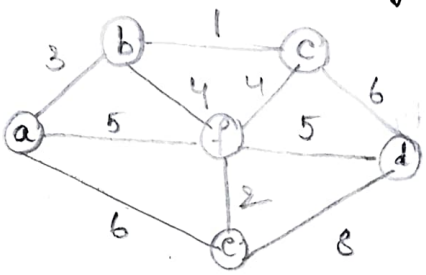
Profit = 68 + 6
= 74

Job Sequence is $J_6 - J_7 - J_4 - J_3$.

Algorithm:

```
Algorithm JobSequencing(n,d,p)
// n is the number of jobs
// d[1..n] stores deadline of job i in d[i]
// p[1..n] stores profit of job i in p[i]
{
  Create a sorted array SJ where SJ[1..n] stores the
  job ids such that p[SJ[i]] <= p[SJ[j]], if i<j
  profit := 0;
  for i:=1 to n do
  {
    j:=SJ[i];
    for k := d[j] to step-1 do
    {
      if(TS[k] =0) then
        TS[k]:=j;
        profit:=profit+p[j];
        break;
    }
  }
  return profit;
```

6. What is Minimum Cost Spanning tree? Find Minimum Cost Spanning tree for the following graph using prim's algorithm.



1. the Subgraph of the given graph is called as Spanning tree.

2. Spanning tree follows 3 rules:
   1. All the vertices must be Considered.
   2. Edges must be (n-1).
   3. It should not have any cycle.

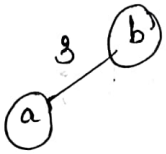
3. A Minimum Spanning Tree is a Subset of edges of a Connected weighted Undirected graph that Connects all the vertices together with the minimum possible total edge weight.

4. If there exist any duplicate weighted edges, the graph may have multiple minimum Spanning tree:

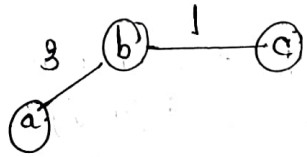
Ex:-

firstly, choose the minimum Cost edge.

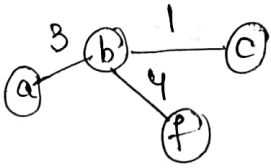
Step-1:



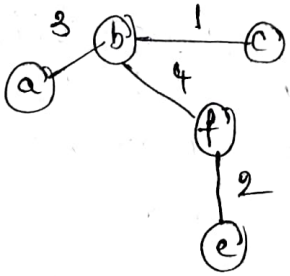
step-2:



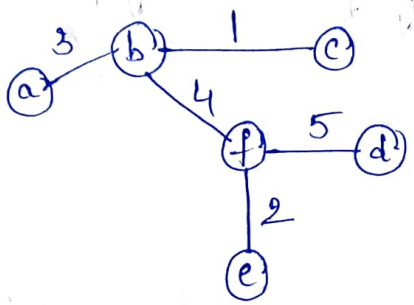
Step-3:



Step-4:



Step-5:



Minimum Cost = 1+4+5+2+3
= 15

t

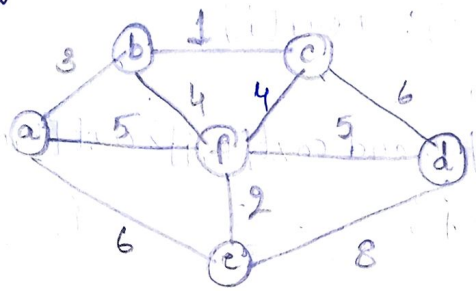
1	a	b
2	c	b
3	f	b
4	e	f
5	d	f

Algorithm Prim(E, Cost, n, t)
// E is the set of edges. N is the number of vertices
// Cost is 2-d array of size nxn such that cost[i,j]
is the cost of the edge between the vertices i and j
if exists else ∞
// t is a 2-d array of size (n-1)x2 . if (p,q) is th
ith edge included in the minimum cost spanning tre

```
{
    let (u,v) be the minimum cost edge in E;
    mincost := cost[u,v];
    t[1,1] := u; t[1,2] := v;
    for i := 1 to n do
        if(cost[i,u]<cost[i,v]) then near[i]:= u;
        else near[i] := v;
    near[u] := 0; near[v] := 0;
    for i := 2 to n do
    { choose j such that near[j] ≠ 0 and cost[j, near[j]]
```

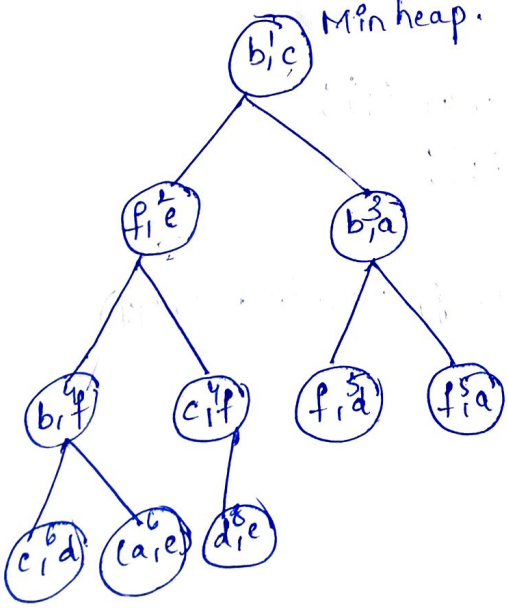
```
        the minimum;
        near[j] := 0;
        mincost := mincost + cost[j, near[j]];
        t[i,1] := j ; t[i, 2] := near[j];
        for k := 1 to n do
            if (near[k] ≠ 0 and cost[k,j] < cost[k,near[k]]) then
                near[k] := j ;
    }
    return mincost;
}
{
    let (k,l) be the minimum cost edge in E;   O(|E|)
    mincost := cost[k,l];
    t[1,1] := k; t[1,2] := l ;
    for i := 1 to n do   O(n)
        if (cost[i,k] < cost[i,l]) then near[i] := k;
        else near[i] := l;
    near[k] := 0 ; near[l] := 0 ;
    for i := 2 to n-1 do   O(n)
    {
        chose j such that near[j] ≠ 0 and cost[j, near[j]] is
        minimum;
        near[j] := 0 ;
        mincost := mincost + cost[j, near[j]];
        t[i,1] := j; t[i,2] := near[j];
        for k := 1 to n do   O(n)
            if (near[k] ≠ 0 and cost[k,j] < cost[k, near[k]])
                then near[k] := j ;
    }
}
return mincost;
```

7. What is Minimum Cost Spanning tree? Find Minimum Cost Spanning tree for the following graph using Krushkal's algorithm.



1. The Subgraph of the given graph is called as Spanning tree.
2. Spanning tree follows 3 rules:
   1. All the Vertices must be Considered in the Subgraph.
   2. Edges must be (n-1)
   3. It should not have any cycle.
3. A Minimum Spanning Tree is a Subset of edges of a Connected Weighted graph that connects all the Vertices together with the minimum possible total edge weight.
4. If there exists any duplicate weighted edges, the graph may have multiple minimum Spanning tree.

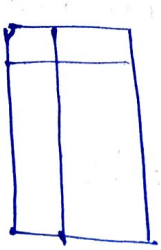
Ex:- Step-1: Consider the minimum cost edge first.



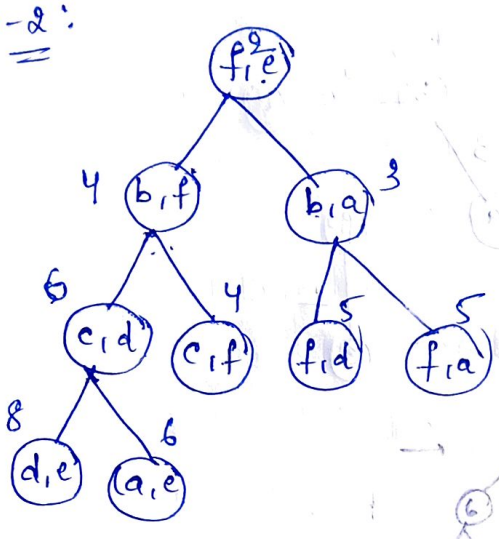
cost = 0

{a}, {b}, {c}, {d}, {e}, {f}.

t



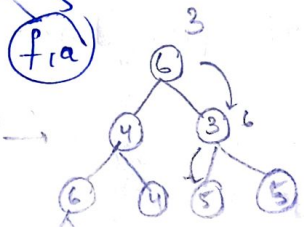
Step-2:



Edges

t

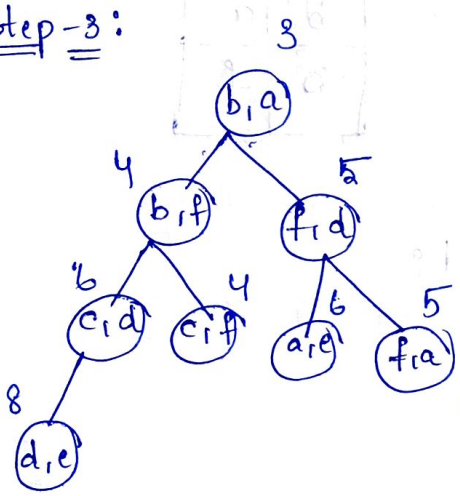
b	c



{ a, {b, c}, {d}, {e}, {f} }.

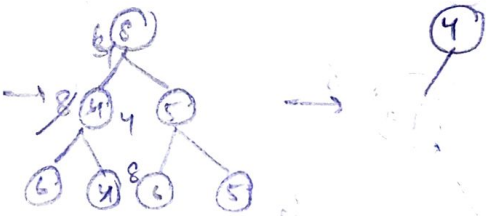
Cost = 1

Step-3:



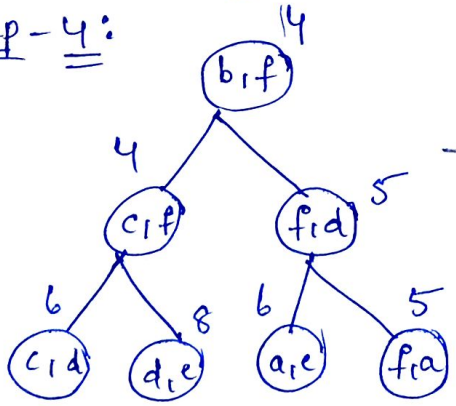
t

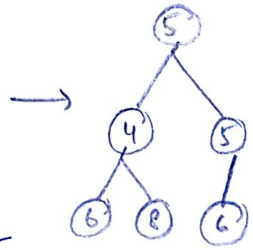
b	c
f	e



Cost = 1 + 2
= 3.

Step-4:

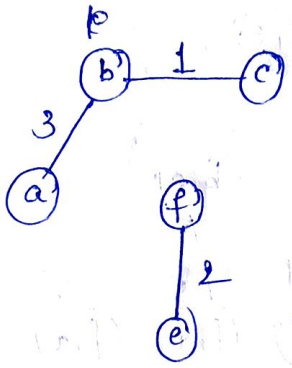




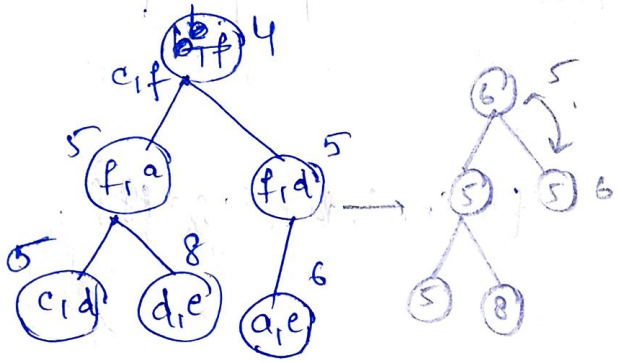
b	c
f	e
b	a

Cost = 6

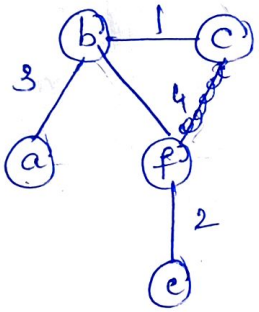
{a, b, c}, {d}, {e, f}



Step-5:



b	c
f	e
b	a
b	f

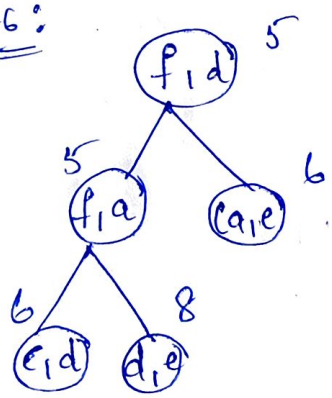


Cost = 6 + 4

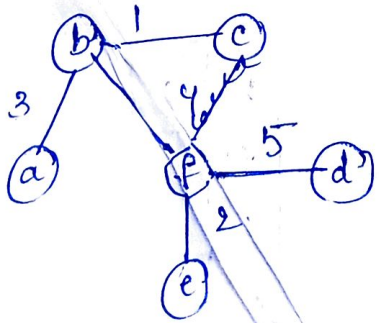
= 10

{a, b, c, e, f}, {d}

Step-6:



{a, b, c, d, e, f}, {d}



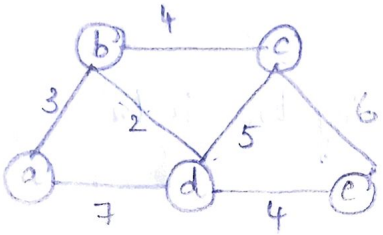
Cost = 1+2+3+4+5

= 15.

∴ Minimum Cost for the Spanning tree is 15.

```
Algorithm Kruskal(E, cost, n, t)
// E is the set of edges in G
// n is the number of vertices
// Cost is a matrix such that cost(u,v) is the cost of the
                                        edge(u,v)
// t is the array of edges included in the MST.
{
  Construct the min heap of edges in E based on cost;
  i := 1 ; mincost := 0 ;
  while (i<n and heap not empty) do
  {
    Delete minimum cost edge(u,v) from the heap
    j := Find (u); k := Find(v);
    if (j ≠ k) then
    {
      t[i,1] := u ; t[i,2] := v ;
      mincost := mincost + cost[u,v];
      Union(j, k);
      i := i+1;
    }
  }
  if (i ≠ n) then write ("No Spanning Tree");
  return mincost;
}
```

8. What is Single-Source shortest-paths problem? Find shortest paths to all other vertices from a.



1. Dijkstra algorithm can be used to find the shortest path from a source vertex to all other vertices.
2. It works for both directed and Undirected graphs.
3. Selected vertices whose shortest distance has been Computed.
4. Unselected vertices are those whose shortest distance is yet to be Computed.
5. Unselected vertices record the shortest distance from it to selected vertices.
6. Whenever a vertex is selected, Unselected vertices update their shortest distances.

Ex:-

cost	a	b	c	d	e
a	0	3		7	
b	3	0	4	2	
c		4	0	5	6
d	7	3		0	4
e			6	4	0

Step-1:

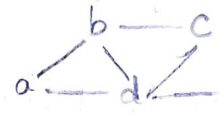
	a	b	c	d	e	f
Selected	f	f	f	f	f	f
dist	0	∞	∞	∞	∞	∞
prev	-	-	-	-	-	-

Nearest Selected vertex u = a.

Step-2:

	a	b	c	d	e
Selected	T	f	f	f	f
dist	0	3	∞	7	∞
Prev		a		a	

Nearest selected vertex u = b.



Step-3:

	a	b	c	d	e
Selected	T	T	f	f	f
dist	0	3	3+4 = 7	3+2 = 5	∞
Prev		a	b	b	

Consider the distance from 'a'.

Nearest selected vertex u = d.

Step-4:

	a	b	c	d	e
Selected	T	T	f	T	f
dist	0	3	7	5	5+4 =9
Prev		a	b	b	d

Nearest vertex = c

Step-5:

	a	b	c	d	e
Selected	T	T	T	T	f
dist	0	3	7	5	9
Prev.		a	b	b	d

```
Algorithm Shortestpath(v, cost, dist, n)
// n is the number of vertices in the Graph G.
// Cost is the adjacency matrix of G.
// v is the source vertex.
// dist[j]  1<=j<=n  records the shortest distance of
vertex j from source vertex v.
{
    for j:=1 to n do
        s[j]:=false;
    dist[v]:=0;
    for k:=1 to n do
    {
        choose u such that s[u]=false and dist[u] is the
        minimum;
        s[u]:=true;
        for each adjacent vertex w of u with s[w]=false
        do
        if (dist[w] > dist[u]+cost[u,w]) then
            dist[w] := dist[u]+cost[u,w]
    }
}
```