Algorithm Convension.

Algorithm Sum (a,b)
begin / {
C ← a+b ; c := a+b
return c
end / }

for (i=0 ; i<n ; i++)

① for (i ← 0 to n do step +1)

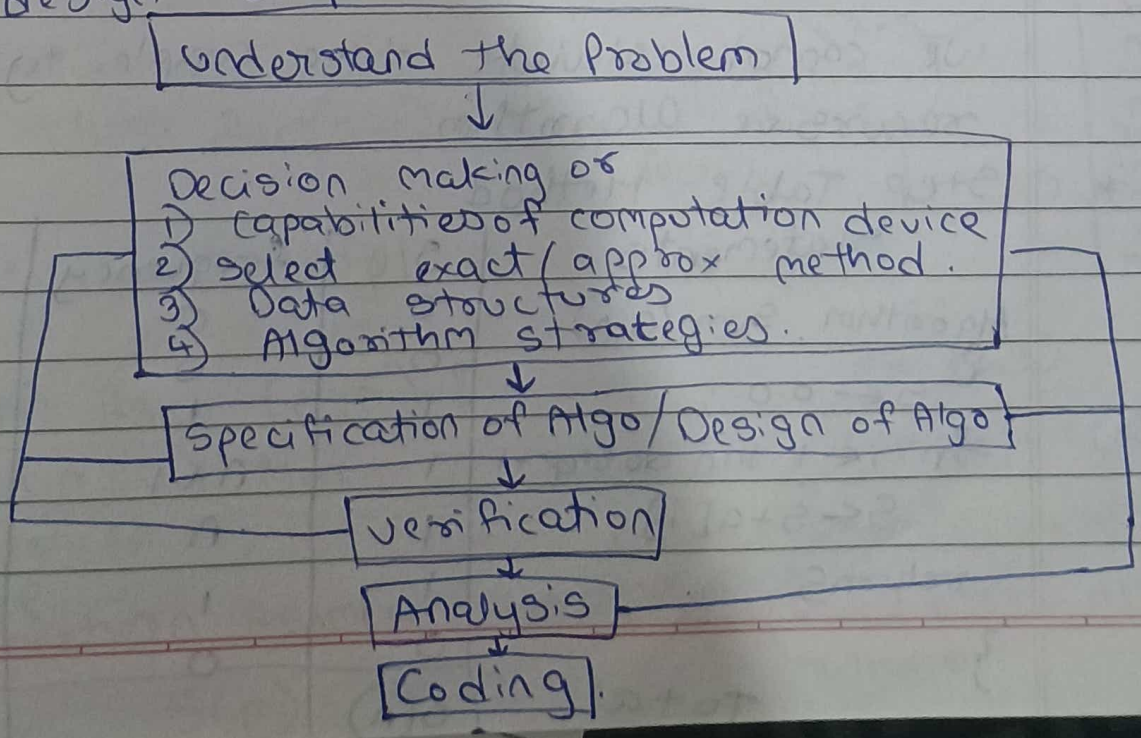
② while condition do while () {
}

③ repeat { do {
} }
until ( ) while()

④ if (n==2) then if (n=2) {
}

⑤ (a (AND) b) (a && b)
(a OR b) (a || b)
(NOT a) (!a)

* Algo design steps



Specification of Algo
- using natural language
- flow chart
- pseudo code

Properties / Characteristics of Algorithm

① input ② output ③ finiteness

④ Definiteness → clear & unambigious

⑤ Effectiveness → pen & paper (traceble)

Analysis

f(n) Time Complexity - order of growth

Space Complexity - Best, worst, average case.
Input size.

$\log_2 n$	n	n^2	n^3	2^n	3^n
0	1	1	1	2	3
1	2	4	8	4	9
1585	3	9	27	8	27
2	4	16	64	16	81
2322	5	25	125	32	243

We cannot calculate time complexity of recursive algorithm

* Step Table Method

Statements	S/e	frequency	total step
Algorithm Sum (a,n)	0	0	0
{	0	0	0
s ← 0.0;	1	1	1
for i ← 1 to n do step +1	1	n+1	n+1
s ← s+a[i];	1	n	n
return s	1	1	1
}	0	0	0
Total	O(n)		2n+3

Statement	s/e	frequency	T steps
Algorithm MatAdd(A,B,C,m,n)	0	0	0
{	0	0	0
for i ← 1 to m do step+1	1	m+1	m+1
for j ← 1 to n do step+1	1	m(n+1)	~~m(n+1)~~ mn+m
c[i,j] ← A[i,j] + B[i,j]	1	mn	mn
}	0	0	0

Time complexity - Total = 2mn+2m+1

* Space complexity

S(P) = fixed part + variable part.

① s ← 0;
for i ← 1 to n do step+1
s ← s + a[i];
return s

s, i, n, whole

4

n+4

Total space complexity = n+4.

② Algorithm fact(n)
{
if (n <= 0) then
return 1;
else
return fact(n-1) * n;
}

n+1 depth

fact(4) → 24
fact(3) → 6×4 = 4
fact(2) → 2×3 = 3
fact(1) → 1×2 = 2
fact(0) → 1×1 = 1

0	n=0 ret
1	n=1 ret
2	n=2 ret
3	n=3 ret
4	n=4 ret

Space complexity = 2(n+1) = 2n+2

③ Algorithm Rsum (a,n) {
if (n ≤ 0) then
return 0;
else
return Rsum(a, n-1) + a[n];
}

Rsum(a,3)
Rsum(a,2) + a[3]
Rsum(a,1) + a[2]
Rsum(a,0) + a[1]
return 0

S(P) = 3n+3

Q Find Time & space of following

```
Algorithm Demo (A, 1, n)
{
    create B[n];
    for i ← 1 to n do step+1
        B[i] ← A[i];
}
```

i	A	B	i	n
1	0	0	1	1

$2n+3$

$\boxed{\text{Space} = 2n + 3}$

Step Table method

	s/e	freq	Total step
Algorithm Demo (A, 1, n)	0	0	0
{	0	0	0
create B[n];	1	1	1
for i ← n to n do step+1	1	n+1	n+1
B[i] ← A[i];	1	n	n
}	0	0	0

$\boxed{\text{Time} = 2n + 2}$

* Backward substitution method.

1] $T(n) = T(n-1) + 1$ — ①

with stopping condition $T(0) = 1$

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

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

Subsitute in eq ①

$T(n) = [T(n-2) + 1] + 1$

$T(n) = T(n-2) + 2$ — ②

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

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

subsitute in ②

$T(n) = [T + (n-3) + 1] + 2$

$T(n) = T(n-3) + 3$ — ③

$T(0)$

$T(n) = T(n-k) + k$

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

$T(n) = T(0) + n$

$\boxed{T(n) = 1 + n}$

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

2] $T(n) = T(n-1) + 2$ with initial cond^n $T(0) = 2$

$T(n) = T(n-1) + 2$ — ①

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

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

Substitute in ①

$T(n) = [T(n-2) + 2] + 2$

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

$T(n) = T(n-2) + 2(2)$ — ②

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

$T(n-2) = T(n-3) + 2$

Substitute in ②

$T(n) = [T(n-3) + 2] + 2(2)$

$T(n) = T(n-3) + 6$

$T(n) = T(n-3) + 3(2)$ — ③

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

$T(n-3) = T(n-4) + 2$

substitute in ③

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

$T(n) = T(n-4) + 4(2)$ — ④

$T(n) = T(n-k) + k(2)$

Assume $n = k$

$T(n) = T(n-n) + n(2)$

$T(n) = T(0) + 2n$

$T(n) = 2 + 2n = \underline{\underline{2n+2}}$

Q $T(n) = T(n-1) + 3$ if $n > 0$

with initial condition $T(0) = 3$

$T(n) = T(n-1) + 3$ — (1)

$T(n-1) = T(n-1-1) + 3$

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

substitute in (1)

$T(n) = [T(n-2) + 3] + 3$

$T(n) = T(n-2) + \cancel{3(3)}$ — (2)

$2(3)$

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

$T(n-2) = T(n-3) + 3$

substitute in (2)

$T(n) = [T(n-3) + 3] + 2(3)$

$T(n) = T(n-3) + 3(3)$ — (3)

$T(n) = T(n-k) + k(3)$

Assume $n = k$

$T(n) = T(n-n) + n(3)$

$T(n) = T(0) + 3n$

$T(n) = 3n + 3$

$T(n) = 3(n+1)$

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

with initial condition $T(1) = 1$

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

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

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

substitute in ①

$T(n) = 2\left[2T\left(\frac{n}{4}\right) + C\right] + C$

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

$T(n) = 4T(n/4) + 3C$ — ②

$T(n/4) = 2T\left(\frac{n/4}{2}\right) + C$

$T(n/4) = 2T(n/8) + C$

substitute in ②

$T(n) = 4\left[2T(n/8) + C\right] + 3C$

$T(n) = 8T(n/8) + 4C + 3C$

$T(n) = 8T(n/8) + 7C$ — ③

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

$= 2^k T\left(\frac{n}{2^k}\right) + (2^k - 1)C$

Assume $2^k = n$

$T(n) = n\,T\left(\frac{n}{n}\right) + (n-1)C$

$= n\,T(1) + (n-1)C$

$T(n) = n + (n-1)C \quad \because T(1) = 1$

Q $T(n) = T\left(\frac{n}{3}\right) + C$ — ① $T(1) = 1$

$T(n/3) = T\left(\frac{n/3}{3}\right) + C$

$T(n/3) = T\left(\frac{n}{9}\right) + C$

$T(n) = \left[T\left(\frac{n}{9}\right) + C\right] + C$

$T(n) = T\left(\frac{n}{9}\right) + 2C$ — ②

$T\left(\frac{n}{9}\right) = T\left(\frac{n/3}{9}\right) + C$

$T\left(\frac{n}{9}\right) = T\left(\frac{n}{27}\right) + C$

$T(n) = \left[T\left(\frac{n}{27}\right) + C\right] + 2C$

$T(n) = T\left[\frac{n}{27}\right] + 3C$ — ③

$T(n) = T\left(\frac{n}{3^k}\right) + kC$

Assume $n = 3^k$ $\therefore k = \log_3 n$

~~$T(n) = T\left(\frac{n}{3n}\right) + 3C$~~

$T(n) = T\left(\frac{n}{n}\right) + \log_3 n\, C$

$T(n) = 1 + \log_3 n\, C$

Q $T(n) = T(n-1) + 1$ $n > 1$

$n = 1$

Binary Search

```
Algorithm BinSearch (a, i, l, x)
{
// Given an array a[i, l] of element in non
   decending order.
// 1 ≤ i ≤ l, determine whether x is
present or not & if so
// return j such that x = a[j] ; else return 0.

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

declare variable in global.

Q

1	2	3	4	5	6	7
10	-48	36	28	42	68	11

Find max & min from above array using DAC approach. (divide & conquer)

1	2	3	4	5	6	7
10	-48	36	28	42	68	11

i, j, mid

1	2	3	4
10	-48	36	28

i, j, mid

5	6	7
42	68	11

i, j, mid

1	2
10	-48

max = 10
min = -48

3	4
36	28

max1 = 36
min1 = 28

5	6
42	68

max = 68
min = 42

7
11

max1 = 11
min1 = 11

max = 36
min = -48

max = 68
min = 11

maxmin (1, 7, max, min)
mid = 4

max = 68
min = -48

maxmin (1, 4, max, min)
mid = 2

(maxmin 5, 7, max1, min1)
mid = 6

maxmin (1, 2, max, min)
max = 10
min = -48

maxmin (3, 4, max1, min1)
max1 = 36
min1 = 28

maxmin (5, 6, max, min)
max = 68
min = 42

maxmin (7, 7, max1, min1)
max1 = 11
min1 = 11

max = 36
min = -48 → max = 68
min = -48 ← max1 = 68
min1 = 42

* Recurrence Relⁿ for Binary search

$T_{Best}(n) = \Omega(1)$

$T(n) = T_{worst}\left(\frac{n}{2}\right) + 1$ for $n > 1$

$T(1) = 0$ if $n = 1$

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

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

$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 1$

$T(n) = \left[T\left(\frac{n}{4}\right) + 1\right] + 1$

$T(n) = T\left(\frac{n}{4}\right) + 2$ — (2)

$T\left(\frac{n}{4}\right) = T\left(\frac{n}{2}/4\right) + 1$

$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1$

$T(n) = \left[T\left(\frac{n}{8}\right) + 1\right] + 2$ —

$T(n) = T\left(\frac{n}{8}\right) + 3$ — (3)

$\cancel{K =}$ $T(n) = T\left(\frac{n}{2^k}\right) + k$

Assume $\frac{n}{2^k} = 1$ $\therefore k = \log_2 n$

$T(n) = (1) + \log_2 n$

$T(n) = 0 + \log_2 n$

$T(n) = \log_2 n$

Strassen's Matrix Multiplication.

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

$A \quad 2\times 2 \qquad B \quad 2\times 2$

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

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

$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21} \qquad O(n^3)$

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

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

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

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

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

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

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

$O(n^{2.81})$ $\quad V = (A_{12} - A_{22}) (B_{21} + B_{22})$

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

$C_{12} = R + T$

$C_{21} = Q + S$

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

$T(n) = 7T\left(\frac{n}{2}\right) + 18n^2$

$\underset{Q}{B_{11}} \quad \underset{R}{B_{11}} \quad \underset{S}{A_{22}} \quad \underset{T}{B_{22}}$

$\xrightarrow{T \oplus}$

$$S\ominus \uparrow \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \downarrow R\ominus \qquad \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$\xrightarrow{Q \oplus}$

Q $A \begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$ $B \begin{bmatrix} 6 & 5 \\ 8 & 4 \end{bmatrix}$ B_{11} A_{11} A_{22} B_{22}

$$C = \begin{bmatrix} 36 & 22 \\ 32 & 24 \end{bmatrix}$$

$P = (2+1)(6+4) = 3 \times 10 = 30$

$Q = B_{11}(4+1) = 6(4+1) = 30$

$R = A_{11}(5-4) = 2(5-4) = 2$

$S = A_{22}(8-6) = 1(8-6) = 2$

$T = B_{22}(2+3) = 4(2+3) = 20$

$U = (4-2)(6+5) = 22$

$V = (3-1)(8+4) = 2 \times 12 = 24$

$C_{11} = P + S - T + V = 30 + 2 - 20 + 24 = 36$

$C_{12} = R + T = 22$

$C_{21} = Q + S = 32$

$C_{22} = P + R - Q + U = 30 + 2 - 30 + 22 = 24$

$$C = \begin{bmatrix} 36 & 22 \\ 32 & 24 \end{bmatrix}$$

```
Algorithm MM (A, B, n)
{
  if (n ≤ 2) then
  {
      C11 ← a11 × b11 + a12 * b21;
      C12 ← a11 × b12 + a12 * b22;
      C21 ← a21 × b11 + a22 × b21;
      C22 ← a21 × b12 + a22 × b22;
  }
```

```
else
{
Divide A & B with order n×n into 4
subset each with order
MM (A11, B11, n/2) + MM (A12, B21, n/2);
MM (A11, B12, n/2) + MM (A12, B22, n/2);
MM (A21, B11, n/2) + MM (A22, B21, n/2);
MM (A21, B12, n/2) + MM (A22, B22, n/2);
}
}
```

* Algorithm Mergesort (low, high)

```
{
  if (low < high)
  {
    mid ← (low + high)/2;
    Mergesort (low, mid);
    Mergesort (mid+1, high);
    Merge (low, mid, high);
  }
}
```

Algorithm Merge (low, mid, high)

```
{
  i ← low
  j ← mid+1
  k ← low
  while (i < mid AND j <= high)
{
  if (a[i] < a[j]) then
    c[k++] ← a[i++];
  else
    c[k++] ← a[j++];
}
```

```
if (i < mid) then
{
  while (i <= mid)
  {
    c[k++] ← a[i++]
  }
}
else
{
  while (j <= high)
  {
    c[k++] ← a[j++];
  }
}
for k ← low to high step +1 do
{
  a[k] ← c[k] ;
}
}
```

* Recurrence Relation

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

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

$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$

$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)\right] + n$

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

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

$$T\left(\frac{n}{4}\right) = 2T\left(n/4/2\right) + \frac{n}{4}$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}$$

$$T(n) = 2^2\left[2T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)\right] + 2n$$

$$T(n) = 2^3\, T\left(\frac{n}{2^3}\right) + 3n$$

$$= 2^k\, T\left(\frac{n}{2^k}\right) + kn$$

$$\frac{n}{2^k} = 1 \quad \therefore \quad n = 2^k, \quad k = \log_2 n$$

$$T(n) = 2^k\, T(1) + k \cdot n$$

$$T(n) = n * 1 + \log_2 n \times n$$

$$= n + n \log_2 n \times n$$

$$T(n) = O(n \log_2 n)$$

* Quick Sort

Best = $O(n \log_2 n)$

worst = $O(n^2)$

1	2	3	4	5	6
5	17	-8	9	6	13
low	i				j high

j ← partition (1, 6)

pivot = 5

i = 2

j = 6

5	-8	17	9	6	13

-8	5	17	9	6	13

pivot = 17

i = 4

j = 6

3	4	5	6	7
17	9	6	13	∞
pivot	i		j	

3	4	5	6
13	9	6	17

pivot = 13
i = 4
j = 5

3	4	5
13	9	6

low ↑ high
j

3	4	5
6	9	13

pivot = 6
i = 4
j = 4

3	4
6	9

```
do{
while(a[i] <= pivot)
    i++;
while (a[j] > pivot
    j--
if (i < j)
    swap a[i], a[j]
while (i <= j)
    swap a[low], a[j]
}
```

Q worst case for quick sort time complexity

$T(n) = T(n-1) + n \qquad T(1) = 1$

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

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

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

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

$T(n-2) = T(n-3) + (n-2)$

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

$\vdots$

K times

$T(n) = T(n-(K+1)) + (n-K) + \dots + (n-2) + (n-1) + n$

$n-(K+1) = 1 \qquad \therefore K = n+2$

$n-K-1 = 1$

$= T(1) + 2 + 3 + \dots + n$

$= \frac{n(n+1)}{2} = \frac{n^2+n}{2} = O(n^2)$

```
Algorithm RQuicksort (p,q)
{
 // sorts the element a[p], ... a[q] which
  reside in the global array a[1:n] in
 // ascending order a[n+1] is considered
 to be defined & must be ≥ all the
  elements in a[1:n]

 if (p < q) then
 {
   if ((q-p) > 5) then
      interchange (a, Random()mod(q-p+1)+p, p);
   j ← partition (a, p, q+1);
    // j is the position of partitioning element
   RQuicksort (p, j-1);
   RQuicksort ((j+1), q);
 }
}
```

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

* Greedy Approach

```
Algorithm Greedy (a,n)
{
 // a[1:n] contains n inputs
  Solution ← ϕ   // initialize the solution
  for i ← 1 to n do
  {
   x ← select (a);
   if feasible (solution, x) then
      Solution ← union (Solution, x);
  }
  return solution;
}
```

* Fractional Knapsack Problem

maximize $\sum_{1 \le i \le n} p_i x_i$

Subject to $\sum_{1 \le i \le n} w_i x_i \le m$

& $0 \le x_i \le 1, \; 1 \le i \le n$

when completely adding a product in cart the $x_i = 1$
when half adding a product in cart then $x_i = 0.5$

p_i = profit of individual
w_i = complete weight
m = bag size

Problem :

$n = 3$ $m = 20$
$(p_1, p_2, p_3) = (25, 24, 15)$
$(w_1, w_2, w_3) = (18, 15, 10)$

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	28.02
0, 2/3, 1	20	31
0, 1, 1/2	20	31.5

Greedy

knapsack

```
Algorithm ~~Greedy~~ (m, n)
{
  //m is capacity of knapsack
  //n is total object
  //p[1:n] is the weight of each object
  //Arrange objects such that p[i]/w[i] > p[i+1]/w[i+1]
  // x[1:n] is the solution vector
  for i ← 1 to n do step +1
                    //initialize x[i] to 0.0
      x[i] ← 0.0 ;
        U ← m ;
    for i ← 1 to n do step+1
    {
        if (w[i] > U) then break ;
            x[i] ← 1.0 ;
            U ← U - w[i] ;
    }
    if (i ≤ n) then
        x[i] ← U / w[i] ;
```

Q

$n = 3$ $\qquad U = m$

$m = 20$ $\qquad U = 20$

$(p_1, p_2, p_3) = (25, 24, 15)$

$(w_1, w_2, w_3) = (18, 15, 10)$

$\frac{25}{18} \quad \frac{24}{15} \quad \frac{15}{10}$

$1.38, 1.6, 1.5$

Profit weight Ratio

After Sorting

$10 > 5$

		20
7.5	5	35
24	15	
		31.5

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

$(w_1 \quad w_2 \quad w_3) = (15, 10, 18)$

$i = 2$

$2 \leq 3$

$x[2] = 5/10 = 1/2 = 0.5$

$i \quad U \quad w[2]$

1.0	0.5	0.0

Max $\Sigma P_i x_i$

① $24 \times 1 = 24$

② $15 \times 0.5 = 7.5$

③ $25 \times 0.0 = 0$

31.5

Subject To $\Sigma w_i x_i \leq m$

① $15 \times 1 = 15$

② $10 \times 0.5 = 5$

③ $18 \times 0 = 0$

20

Q] $n = 5$ • $m = 30$

$(P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5) = (30, 45, 32, 15, 16)$

$(w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5) = (10, 18, 12, 21, 11)$

P_i	30	32	45	16	15
w_i	10	12	18	11	21

$x_i = \boxed{0.0 \mid 0.0 \mid 0.0 \mid 0.0 \mid 0.0}$

$U = 30$

$i=1 \quad 10 > 30 \quad U = 20 \quad x_1 = 1.0$

$i=2 \quad 12 > 20 \quad U = 8 \quad x_2 = 1.0$

$i=3 \quad 18 > 8 \quad U = 0 \quad x_3 = U/w_i = 8/18 = 0.44$

$x_i = \boxed{1.0 \mid 1.0 \mid 0.44 \mid 0.0 \mid 0.0}$

Total $= P_i x_i = 30 + 32 + 20 = 82$ //

Q] $n = 5$ $m = 25$ 1.6 2.75 1.33 1 1.5

$(p_1\ p_2\ p_3\ p_4\ p_5) = (16, 22, 12, 13, 9)$

$(w_1\ w_2\ w_3\ w_4\ w_5) = (10, 8, 9, 13, 6)$

p_i	22	16	9	12	13
w_i	8	10	6	9	13

$x_i = \boxed{0.0 \mid 0.0 \mid 0.0 \mid 0.0 \mid 0.0}$

$U = 25$

$i=1$ $8 > 25$ $x_i = 1.0$ $U = 17$

$i=2$ $10 > 17$ $x_i = 1.0$ $U = 7$

$i=3$ $6 > 7$ $x_i = 1.0$ $U = 1$

$i=4$ $9 > 1$ $x_i = U/w_i = 1/9 = 0.11$

$x_i = \boxed{1.0 \mid 1.0 \mid 1.0 \mid 0.11 \mid 0.0}$

$p_i w_i = 22 + 16 + 9 + 1.32 = 48.32$ //

* Kruskal

Heapify $(i = \frac{n}{2}; i >= 1; i--)$

leafchild = $j = i * 2$;

$a[j] > a[j+1]$

$j++$

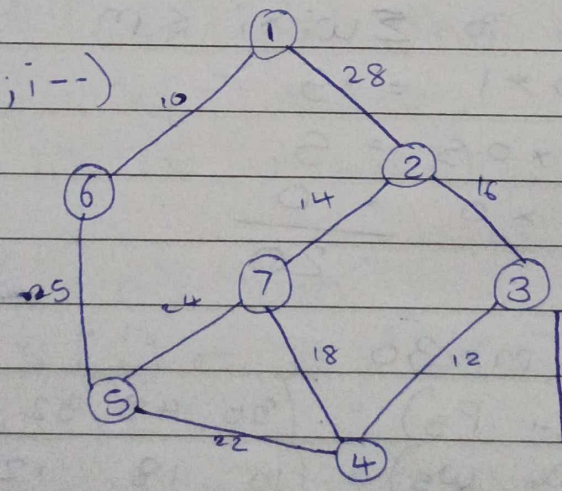
item $\leftarrow a[i]$

$a[j/2] \leftarrow a[j]$

$j = \text{find}(u)$ $u = 1$

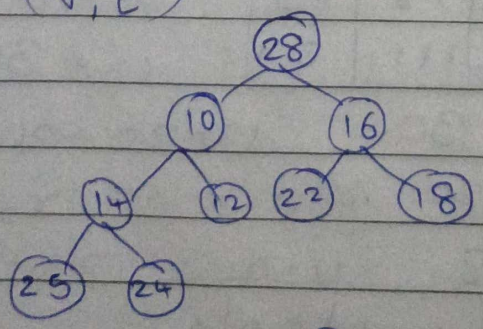
$k = \text{find}(v)$ $v = 6$

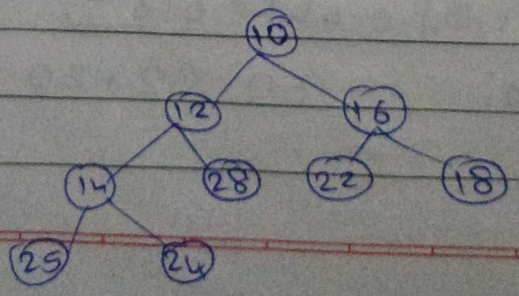
$G'(V, E')$



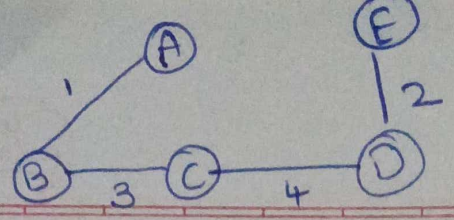
No cycle

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



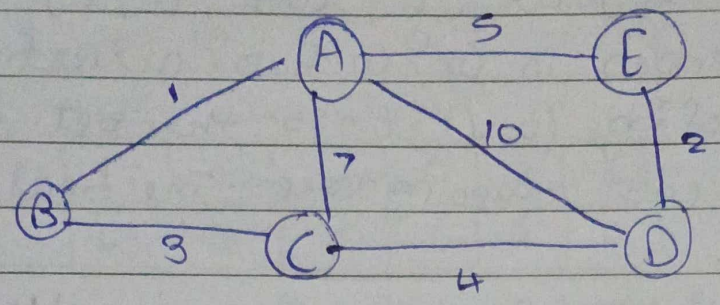


```
Algorithm Kruskal (E, cost, n, t)
// E is the edges in G. G has n vertices. cost[u,v]
// is the cost of edge (u,v). t is the set of edges
in the minimum-cost spanning tree. The final cost is returned
{
  Construct a heap out of edge costs using Heapify;
  for i ← 1 to n do parent[i] ← -1;
  // Each vertex is in a different set.
   i ← 0 ; mincost ← 0.0;
  while ((i < n-1) and (heap not empty)) do
   {
     Delete a minimum cost edge (u,v) from the heap &
     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;
Algorithm SimpleUnion(i,j){
     p[i] ← j;
}
Algorithm SimpleFind(i){
    while (p[i] ≥ 0) do i ← p[i];
      return i;
}
```



	1	2
1	A	B
2	D	E
3	B	C
4	C	D

①



u	v	cost
A	B	1
A	C	7
A	D	10
A	E	5
B	C	3
C	D	4
D	E	2

E [ ] [ ]

edge [10]

struct edge {

char v_1

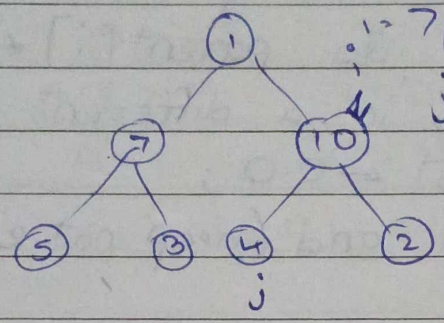
char v_2

int ~~edge~~ cost

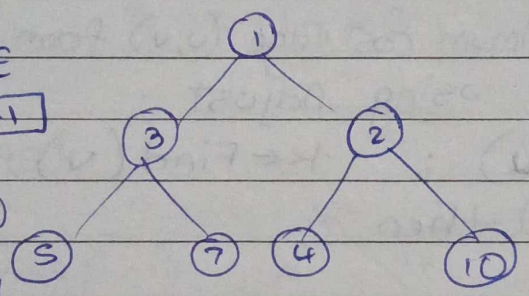
}

$i = 7/2 = 3$

$j = 2 * i$



Mean Heap Tree



A	B	C	D	E
-1	-1	-1	-1	-1

parent [ ]

B	-1	-1	-1	-1

B	-1	-1	E	-1

mincost = 0.0

B	C	-1	E	-1

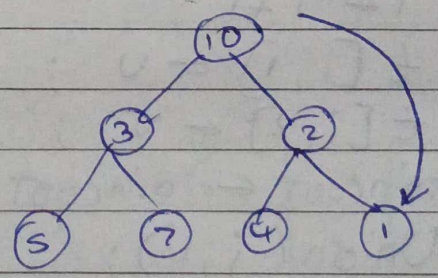
i = 0

B	C	E	E	-1

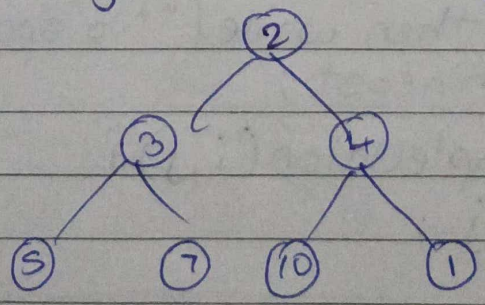
0 < 4

edge = 1

u = A v = B



Again Min Heap



j = find (A) = A

k = find (B) = B

union (A, B)

parent [A] ← B

find (i) {

while (parent[i] > 0)

i ← parent [i];

return i;

}

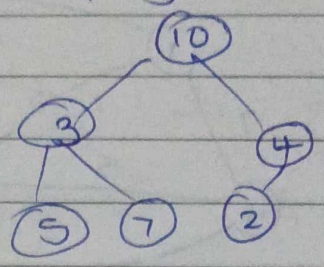
i = 1

1 < 4

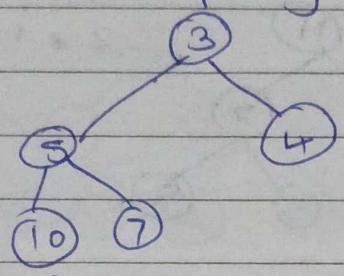
edge = 2

u = D , v = E

~~Reheapify~~



Reheapify



j = find (D) = D

K = find (E) = E

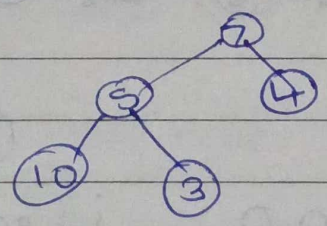
union (D, E)

i = 2 mincost = 3

2 < 4

edge = 3.

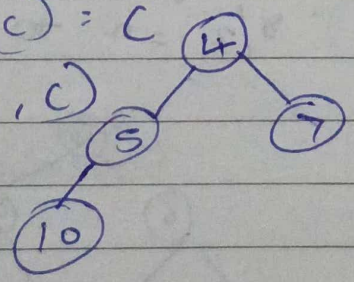
u = B v = C



j = find (B) = B

K = find (C) = C

union (B, C)



i = 3 mincost = 6

3 < 4

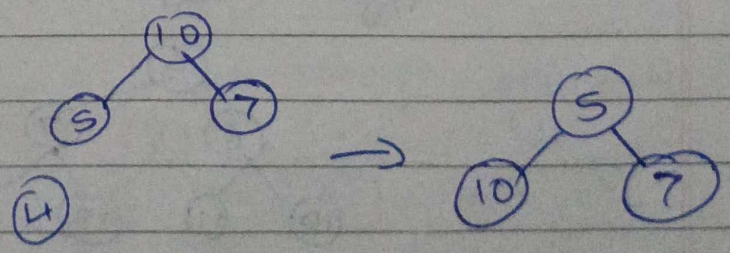
edge = 4

u = C v = D

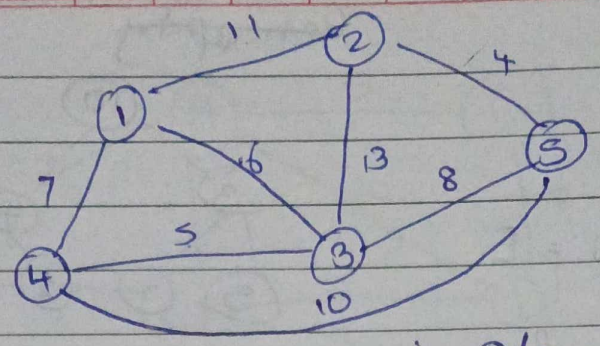
j = find (C) = C

K = find (D) = E

union (C, E)



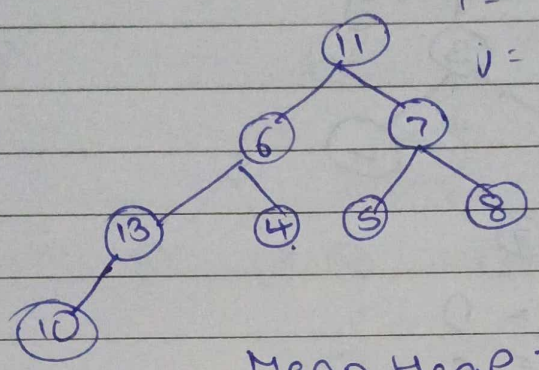
Q



U	V	cost
1	2	11
1	3	6
1	4	7
2	3	13
2	5	4
3	4	5
3	5	8
4	5	10

$i = 8/2 = 4$

$j = 2 * i$

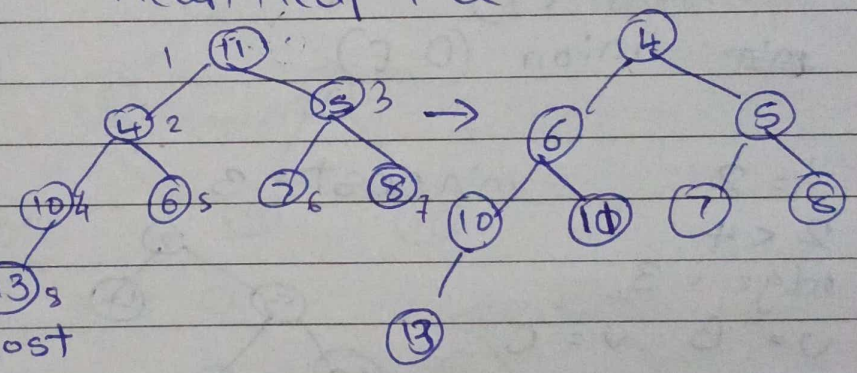


Mean Heap Tree

1	2	3	4	5
-1	-1	-1	-1	-1

-1	5	-1	-1	-1

-1	5	4	-1	-1



mincost = edge + mincost

4	5	4	-1	-1

mincost = 0.0

4	5	4	5	-1

$i = 0$

$0 < 4$ $(n-1)$

edge = 4

u = 2 v = 5

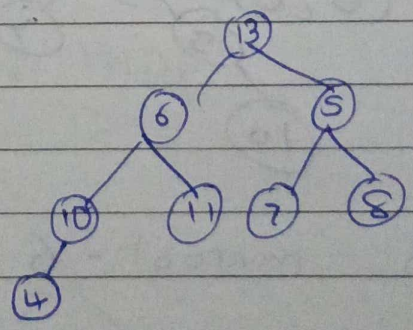
j = find(2) = 2

K = find(5) = 5

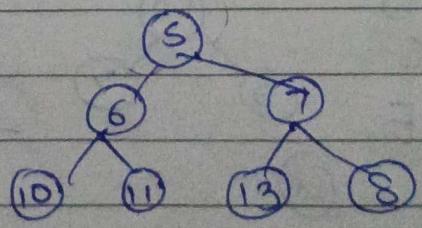
Union(2, 5)

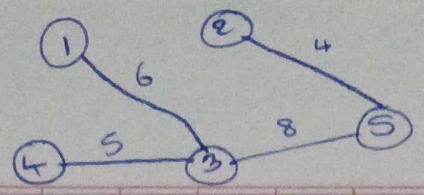
$i = \text{edge}/2$ to 1

$8/2 = 4$ to 1



Again Mean Heap





U	V
2	5
3	4
1	3
3	5

i = 1 mincost = 4

1 < 4

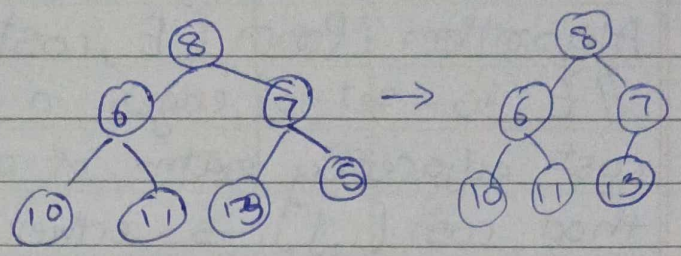
edge = 5

U = 3 V = 4

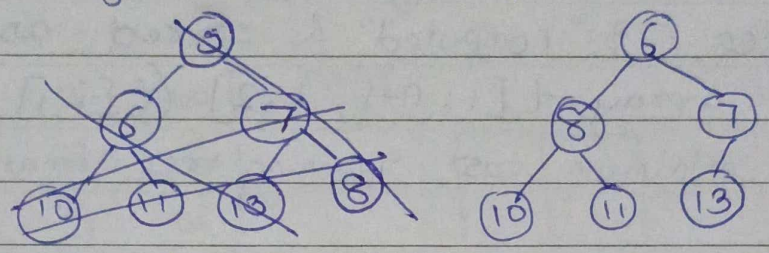
j = find(3) = 3

K = find(4) = 4

union (3,4)



Again Mean Heap



i = 2 mincost = 9

2 < 4

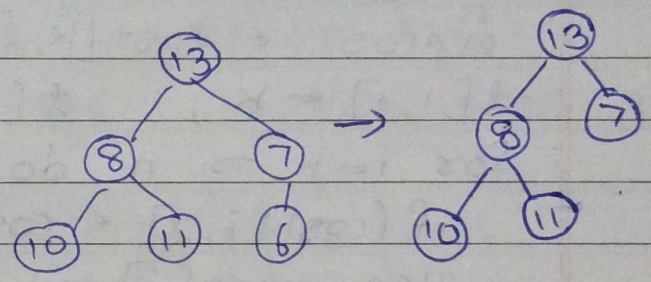
edge = 6

U = 1 V = 3

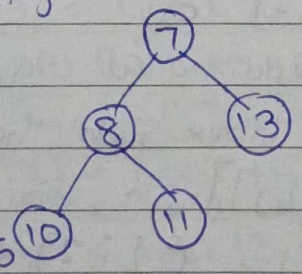
j = find(1) = 1

K = find(3) = 4

union (1,4)



Again Mean Heap



i = 3 mincost = 15

3 < 4

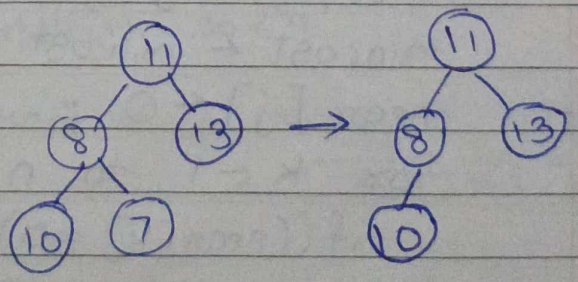
edge = 8

U = 3 V = 5

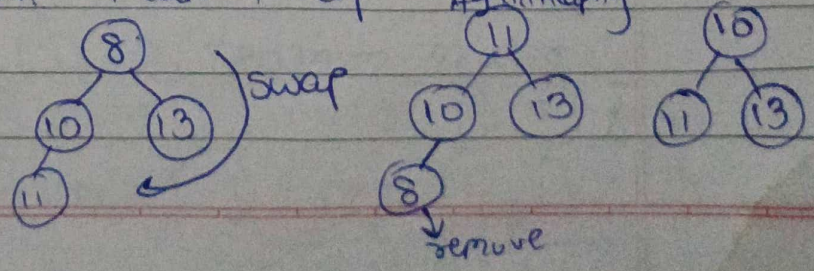
j = find(3) = 4

K = find(5) = 5

union (4,5)



Again Mean Heap



mincost = 23

4/9 DAA

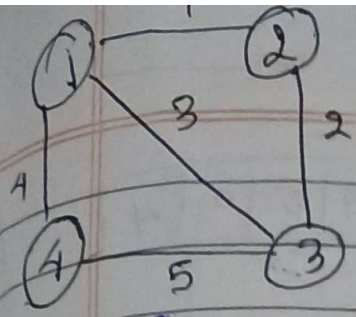
PRIMS

Algorithm Prims (E, cost, n, t)
{
// E is the set of edges in G, cost [1:n, 1:n] is the cost
// adjacency matrix of an n vertex graph such that
cost [i, j] is either a positive real number or ∞ if no
edge(i, j) exists. A minimum spanning tree is computed
& stored as a set of edges in the array t[1:n-1, 1:2]
(t[i,1], t[i,2]) is an edge in the minimum-cost spanning
tree. The final cost is returned

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 // Initialize near
    if (cost[i, l] < cost[i, k]) then
        near[i] ← l;
    else
        near[i] ← k;
near[k] ← near[l] ← 0;

for i ← 2 to n-1 do
{ // find n-2 additional edges for t
  Let j be an index such that near[j] ≠ 0 &
  cost[j, near[j]] is minimum
  t[i, 1] ← j ; t[i, 2] ← near[j];
  mincost ← mincost + cost[j, near[j]];
  near[j] ← 0;
  for k ← 1 to n do // Update near[]
    if ((near[k] ≠ 0) AND (cost[k, near[k]] > cost[k, j]))
        then near[k] ← j;
}
return mincost;

Page No.



$n = 4$

u	v	cost
1	2	1
1	3	3
1	4	4
2	3	2
3	4	5

v_1	v_2
1	2
2	3
4	3

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

2

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & \infty & 1 & 3 & 4 \\ 2 & 1 & \infty & 2 & \infty \\ 3 & 3 & 2 & \infty & 5 \\ 4 & 4 & \infty & 5 & \infty \end{array}$$

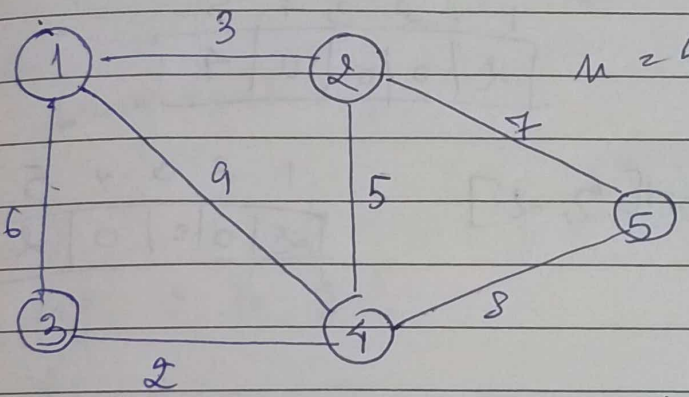
1	2	3	4
2			

near[ ]

(storing nearest vertex for each)

	1	2	3	4
near	0	0	2	1
near	0	0	0	3

1 1 2 2 3



$n = 5$

u	v	cost
1	2	3
1	3	6
1	4	9
2	4	5
2	5	7
3	4	2
4	5	8

cost matrix [ ][ ]

$4 - 3 \Rightarrow 2$ → min edge cost

mincost = 2

t[ ][ ]

i	4	3
	2	4
	1	2
	5	2

for loop - i = 1 to n (1 ke pas 3 h or 4)

$i = 1 \Rightarrow$ cost [1, 3] < cost [1, 4]

6 < 9 ✓

$i = 2 \Rightarrow$ [2, 3] < [2, 4]

∞ < 4 ✗

	1	2	3	4	5
j	3	4	4	3	4

near[ ]

↓

3	4	0	0	4

$\Sigma\, 2(y_i - mx_i - \bar{y} + m\bar{x})^2 * (0 - x_i + \bar{x})$

Page No.

$i = 2$ to 4

2	4	0	0	4

Cost [1, 3] = 6
Cost [2, 4] = 5 ← min
Cost [5, 4] = 8

1	2	3	4	5
3	0	0	0	4

near

$j = 2$

minist = 5 = 7

for k ← 1 to n → update near to consider new edge

k = 1 near [1] ≠ 0

6 > 3 ✓

1	2	3	4	5
2	0	0	0	4

update near as new vertex is being considered

Cost [5, 4] > cost [5, 2]
8 > 7

1	2	3	4	5
2	0	0	0	2

$i = 3$

$j = 1$ Cost [1, 2] = 3
j near [j]

1	2	3	4	5
0	0	0	0	2

j = 2
= 3 ✗ = 0
= 4

Cost [5, 2] = 7

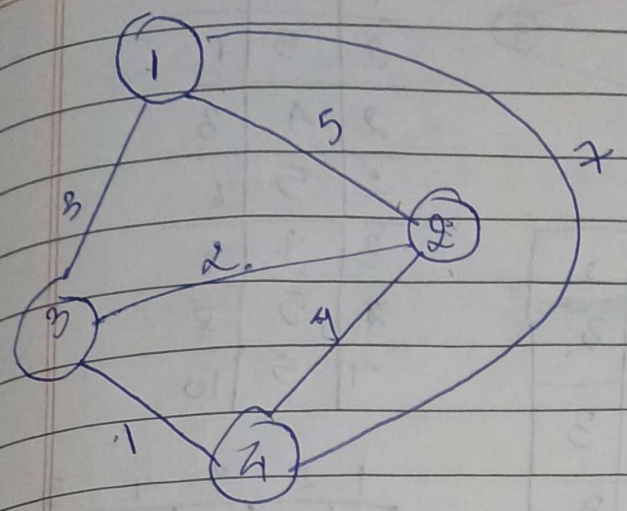
3 < 7

Cost [5, 2] > cost [5, 1]
7 > ∞ ✗

$i = 4$

~~new cost~~ = 5 + 10
= 17

Page No.



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

t[][]

3	4
2	3
2	3

edge (3, 4)
l k

1	2	3	4
3	3	4	3

↓

mincost = 1

1	2	3	4
3	3	0	0

for 2 to 3

i = 2,

edge (2, 3)
l k

j = 1
2
3 >
4 >

1	2	3	4
3	0	0	0

mincost = 1 + 2 = 3

k = 1 >
2 ✓

1	2	3	4
0	0	0	0

i = 3 edge (2, 3)

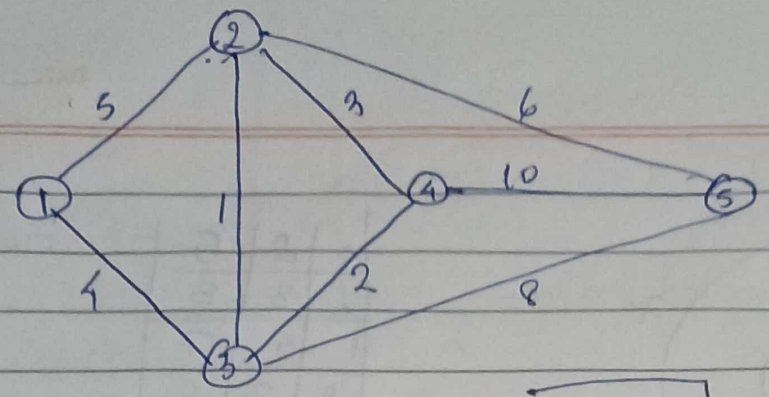
j = 1 2 3 4 >

k = 1 2 3 4 ×

1	2	3	4
0	0	0	0

mincost = 3 + 3
= 6

8



u	v	cost
1	2	5
1	3	4
2	3	1
2	4	3
2	5	6
3	4	2
3	5	8
4	5	10

$n = 5$

	1	2	3	4	5
1	∞	5	4	∞	∞
2	5	∞	1	3	6
3	4	1	∞	2	8
4	∞	5	2	∞	10
5	∞	6	8	10	∞

cost[ ][ ]

X	1	2
1	2	3
2	4	3
3	1	3
4	5	2

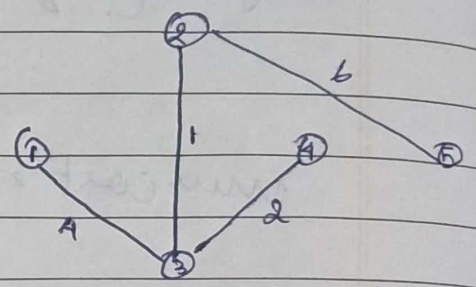
t[ ][ ]

1	2	3	4	5

near[ ]

cost (k, l) ← minimum

cost [2, 3] = 1



t[1, 1] = k = 2 , t[1, 2] = l = 3

minCost = 1;

for i = 1 to n

```
    if (cost[i, k] < cost[i, l])
        near[i] ← k;
    else
        near[i] ← l;
```

i = 1, cost [1, 2] < cost [1, 3]

5 < 4 ✗

near[1] = 3;

1	2	3	4	5
3				

i = 2, cost [2, 2] < cost [2, 3]

∞ < 1 ✗

near [2] = 3

1	2	3	4	5
3	3			

i = 3, cost [3, 2] < cost [3, 3]

1 < ∞ ✓

near [3] = 2

1	2	3	4	5
3	3	2		

$i = 4$, $cost[4, 2] < cost[4, 3]$

$3 < 2$ ✗

1	2	3	4	5
3	3	2	3	

$near[4] = 3$

$i = 5$, $cost[5, 2] < cost[5, 3]$

$6 < 8$ ✓

1	2	3	4	5
3	3	2	3	2

$near[5] = 2$

$near[k] = 0$, $near[l] = 0$

1	2	3	4	5
3	0	0	3	2

for $i = 2$ to $n-1$

for $j = 1$ to n, find minimum $Cost[j, near[j]]$

$j = 1$, $cost[1, near[1]] = 4$

$j = 4$, $Cost[4, near[4]] = 2$ ← minimum

$j = 5$, $cost[5, near[5]] = 6$

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

$= 4$ $= 3$

~~mincost~~ $= 1 + 2 = 3$

$near[j] = 0$;

$near[4] = 0$;

1	2	3	4	5
3	0	0	0	2

for $k = 1$ to n

if ($near[k] \neq 0$ AND $cost[k, j] < cost[k, near[k]]$)

$near[k] = j$;

$k = 1$, $cost[1, 4] < cost[1, 3]$

$\infty < 4$ ✗

$k = 2$, $k = 3$, $k = 4$ ✗ ($near[k] = 0$)

$k = 5$, $cost[5, 4] < cost[5, 2]$

$10 < 6$ ✗

1	2	3	4	5
3	0	0	0	2

$i = 3$,

$j = 1$, $cost[1, near[1]] = 4$ ← minimum

$j = 5$, $cost[5, near[5]] = 0$

$t[i, 1] = \underset{1}{j}$, $t[i, 2] = near[\underset{3}{j}]$

$\underline{mincost = 3 + 4 = 7}$

$near[j] = 0$, $near[1] = 0$

1	2	3	4	5
0	0	0	0	2

for $k = 1$ to n

if ($near[k] \neq 0$ AND $cost[k, j] < cost[k, near[k]]$)

$near[k] = j$

$k = 1$, $k = 2$, $k = 3$, $k = 4$ – ✗ ($near[k] = 0$)

$k = 5$, $cost[5, 1] < cost[5, 2]$

$\infty < 6$ ✗

1	2	3	4	5
0	0	0	0	2

$i = 4$,

$j = 5$, $cost[5, near[5]] = 6$ ← min

$t[i, 1] = \underset{5}{j}$, $t[i, 2] = near[\underset{2}{j}]$

$mincost = 7 + 6 = 13$

$near[j] = near[5] = 0$

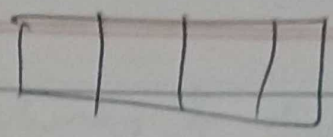
1	2	3	4	5
0	0	0	0	0

for $k = 1$ to n

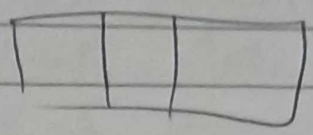
1,2,3,4,5 ✗ $near[k] = 0$

$i = 5$ ✗

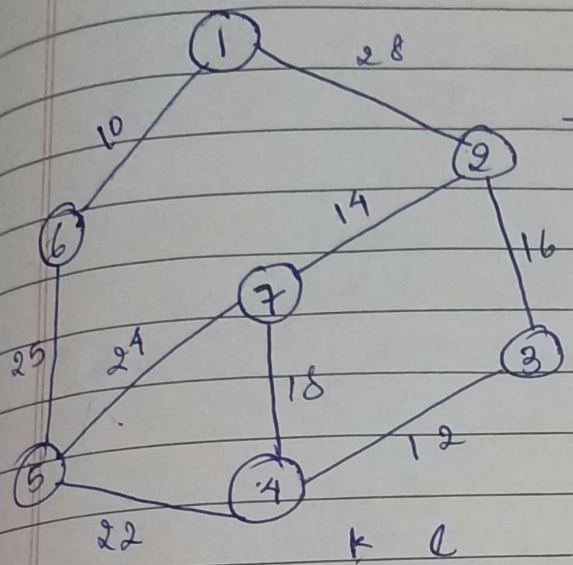
k, l)



j = 1



k = 1 16 > 8



n = 7

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

t[][]

1	6
5	6

cost (1, 6) = 10 ← min

min cost = 10

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

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

near[k] = near[l] = 0

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

for i = 2 to n − 1
6

i = 2, j = 5

mincost = 10 + 25 = 35

t[i, 1] = t[2, 1] = 5

t[i, 2] = t[2, 2] = 6

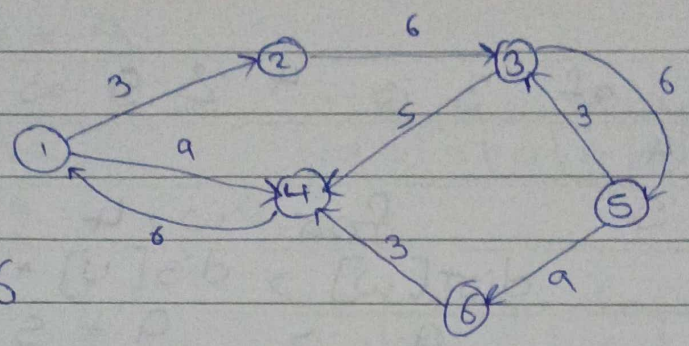
near[j] = 0, near[5] = 0

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

Page No.

Greedy Method

```
Algorithm ShortestPaths (v, cost, distance, n)
// dist[j], 1 ≤ j ≤ n, is set to the length
of shortest path from vertex v to vertex j
in a digraph G with n vertices. dist[v]
is set to zero. G is represented
by its cost adjacency matrix cost[1:n, 1:n]
{
    for i -> 1 to n do
    {   // Initializes
        S[i] ← false;
        dist[i] ← cost[v, i];
    }
    S[v] ← true ; dist[v] ← 0.0; //Put v in S
    for num ← 2 to n do
    {
        // Determine n-1 paths from v
        Choose u ~~that~~ from among those vertices not
        is S such that dist[u] is minimum;
        S[u] ← true ; // Put u in S
        for (each w adjacent to u with S[w]=false) do
            // update distances.
            if (dist[w] > dist[u] + cost[u,w]) then
                dist[w] ← dist[u] + cost[u,w];
        }
}
```



$n = 6$

	1	2	3	4	5	6
1	0	3	∞	9	∞	∞
2	∞	0	6	∞	∞	∞
3	∞	∞	0	5	6	∞
4	6	∞	∞	0	∞	∞
5	∞	∞	3	∞	0	9
6	∞	∞	∞	3	∞	0

cost [ ] [ ]

6	∞
5	∞
4	9
3	∞
2	3
1	0

dist [ ]

	1	2	3	4	5	6
S[ ]	0	0	0	0	0	0

$v = 1$

$i = 1$ to 6

dist [i] ← cost [v] [i] ;

cost [1] [1]

cost [1] [2]

⋮

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

dist [ ]

$u = 2$ ✓

$u = 4$

select the minimum - u

S[u] = true

adjacent

$w = 3$

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

6	∞
5	∞
4	9
3	9
2	3
1	0

dist[w] > dis[u] + cost[u,w]

∞ > 3 + 6

∞ > 9

Now $u = 3$

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

Adjacent of 3 is 4 & 5 as w

w = 4

w = 5

for w = 4

dist[w] > dis[u] + cost[u, w]

9 > 9 + 5

× 9 > 14

for w = 5

dist[w] > dis[u] + cost[u, w]

∞ > 9 + 6

∞ > 15

6	∞
5	15
4	9
3	9
2	3
1	0

dist[]

u = 4

w = 1

1	1	1	1	0	0

but is is already true so we can't take that

Now u = 5

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

w = 6 ✓

w = 3 × it is true

for w = 6

dist[w] > dis[u] + cost[u, w]

∞ > 15 + 9

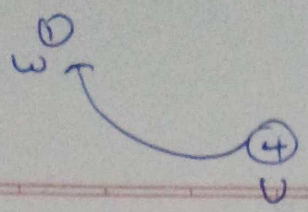
∞ > 24

6	24
5	15
4	9
3	9
2	3
1	0

dist[]

u = 6

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



Q $N = 3$

6	∞
5	6
4	5
3	0
2	∞
1	∞

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

S[]

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

$u = 4$

$w = 1$

$dist[1] > dis[4] + cost[1,4]$

$\infty > 5 + 6$

$\infty > 11$

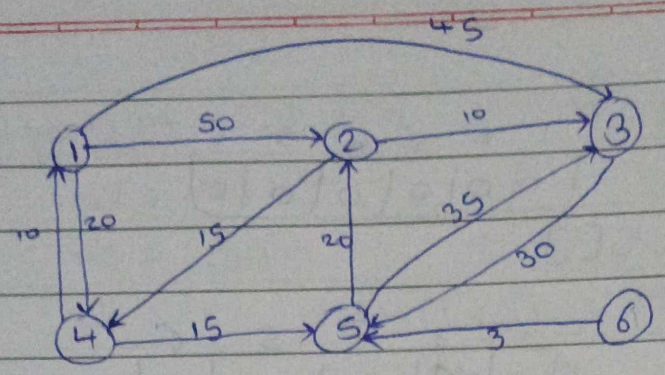
6	∞
5	6
4	5
3	0
2	∞
1	11

0	0	1	1	1	0

$u = 5$

15
6
5
0
14
11

dist[]



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

cost [][]

<u>v = 5</u> i = 1 to 6

dist [i] ← cost [v][i]

6	∞
5	0 ×
4	∞
3	35
2	20 ×
1	∞

S[]

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

S[]

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

n = 2 to n

v = 2

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

w = 3

w = 4

6	∞
5	0
4	∞
3	30
2	20
1	∞

6	∞
5	0
4	35
3	30
2	20
1	∞

for w = 3

35 > 20 + 10

35 > 30

for w = 4

∞ > 20 + 15

∞ > 35

num = 3

v = 3

w = 5

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

S[] is true so we will not consider

num = 4

v = 4

w = 1

w = 5 × cannot consider this

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

For w = 1

∞ > 35 + 10

∞ > 45

6	∞
5	0
4	35
3	30
2	20
1	45

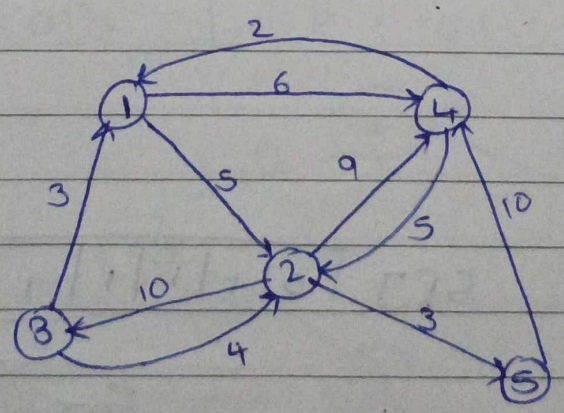
num = 5

v = 1

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

num = 6

Q



V = 2

1	0	5	∞	6	∞
2	∞	0	10	9	3
3	3	4	0	∞	∞
4	2	5	∞	0	∞
5	∞	∞	∞	10	0

cost[][]

V = 2

5	3
4	9
3	10
2	0
1	∞

dist[]

	1	2	3	4	5
S[]	0	0	0	0	0

	1	2	3	4	5
S[]	0	1	0	0	0

num = 2 to n

u = 5

w = 4

	1	2	3	4	5
S[]	0	1	0	0	1

for w = 4

9 > 3 + 10

9 > 13

num = 3

u = 4

w = 1

w = 2 x

	1	2	3	4	5
S[]	0	1	0	1	1

For w = 1

∞ > 9 + 2

∞ > 11

5	3
4	9
3	10
2	0
1	11

num = 4

u = 3

w = 1 x

w = 2 x

	1	2	3	4	5
S[]	0	1	1	1	1

For w = 1

11 > 10 + 3

11 > 13

num = 5

u = 1

	1	2	3	4	5
S[]	1	1	1	1	1

Huffman code tree

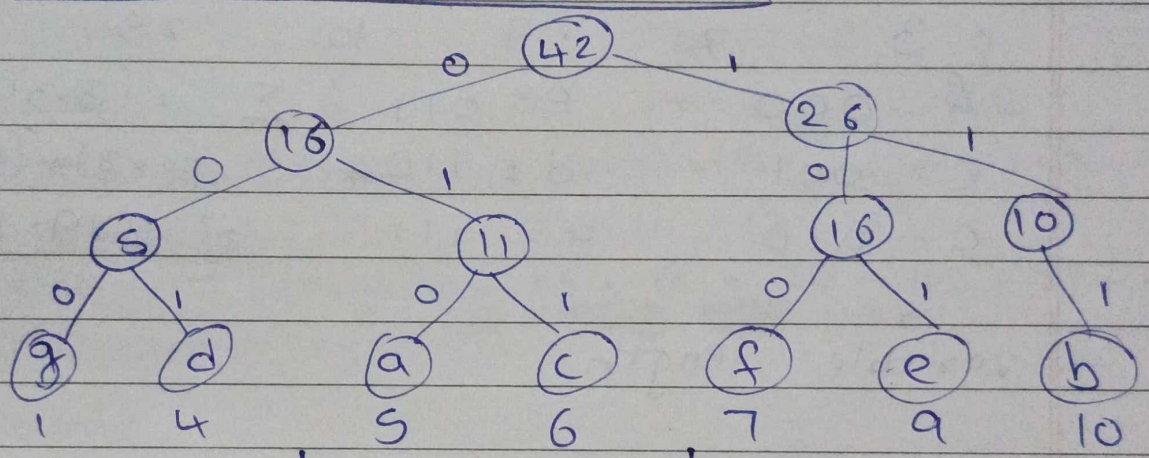
4 Binary value

fixed length encoding — variable length encoding

a	b	c	d	e	f	g
5	10	6	4	9	7	1

4×5	4×10	4×6	4×4	4×9	4×7	4×1	
20	40	24	16	36	28	4	= 168

g	d	a	c	f	e	b
1	4	5	6	7	9	10

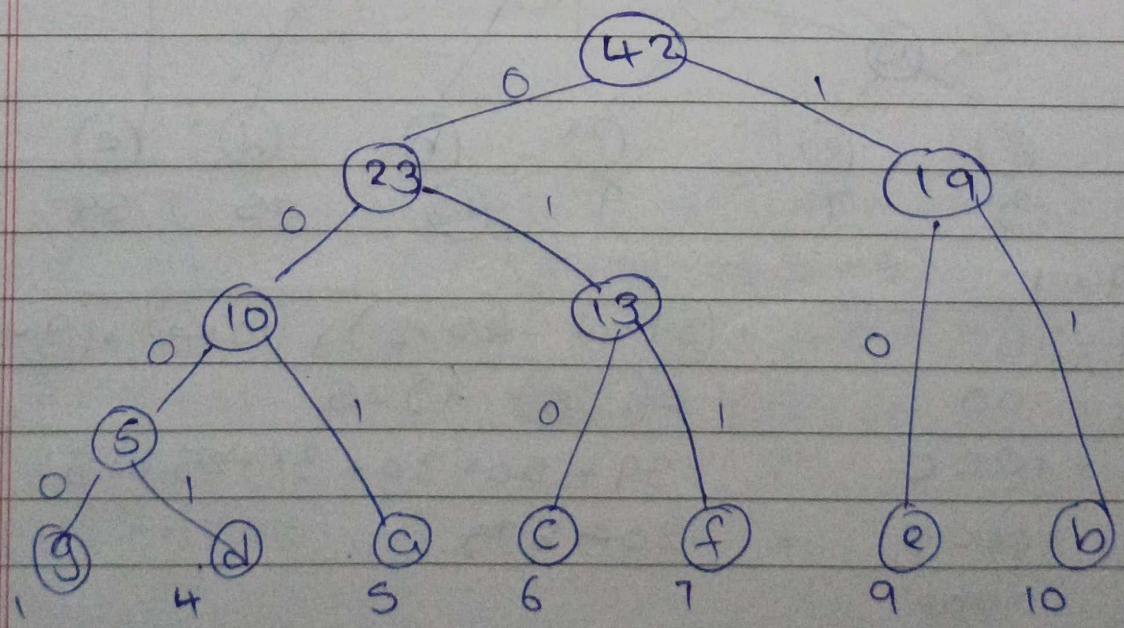


g = 000
d = 001
a = 010

c = 011
f = 100
e = 101

b = 111

$\sum$ freq × no of bits = 126

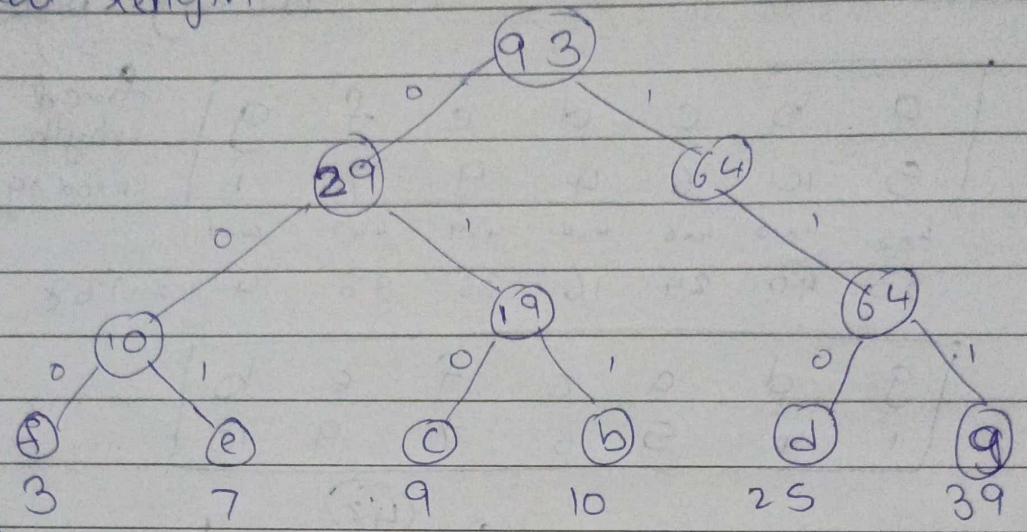


g = 0000 a = 001 f = 011 b = 11
d = 0001 c = 010 e = 10 $\sum$ freq × no of bits = 112

Q

a	b	c	d	e	f	→	f	e	c	b	d	a
39	10	9	25	7	3		3	7	9	10	25	39

fixed length

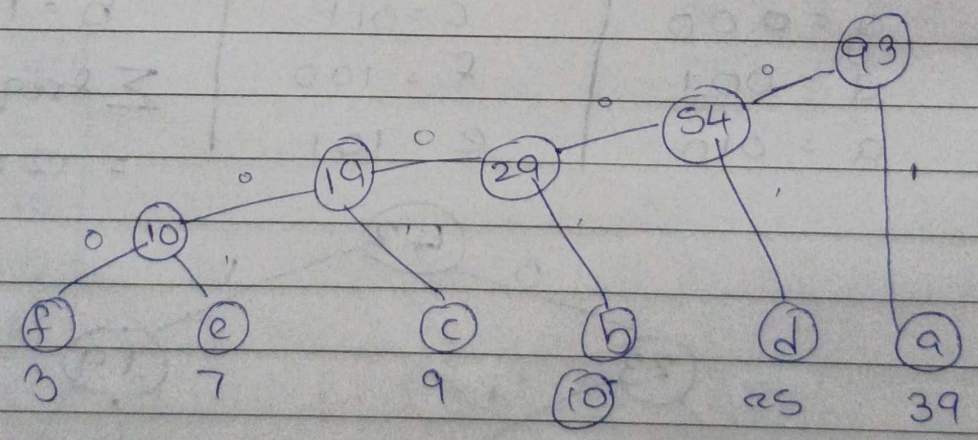


f = 000 b = 011 $\Sigma = (3\times3)+(7\times3)+(9\times3)+(10\times3)+(25\times3)+(39\times3) =$

e = 001 d = 110

c = 010 a = 111 = 279 bits

variable length



a = 1

d = 10 $= (39\times1)+(25\times2)+(10\times3)+(9\times4)+(7\times5)+3\times5$

b = 100

c = 1000 = 39 + 50 + 30 + 36 + 25 + 15

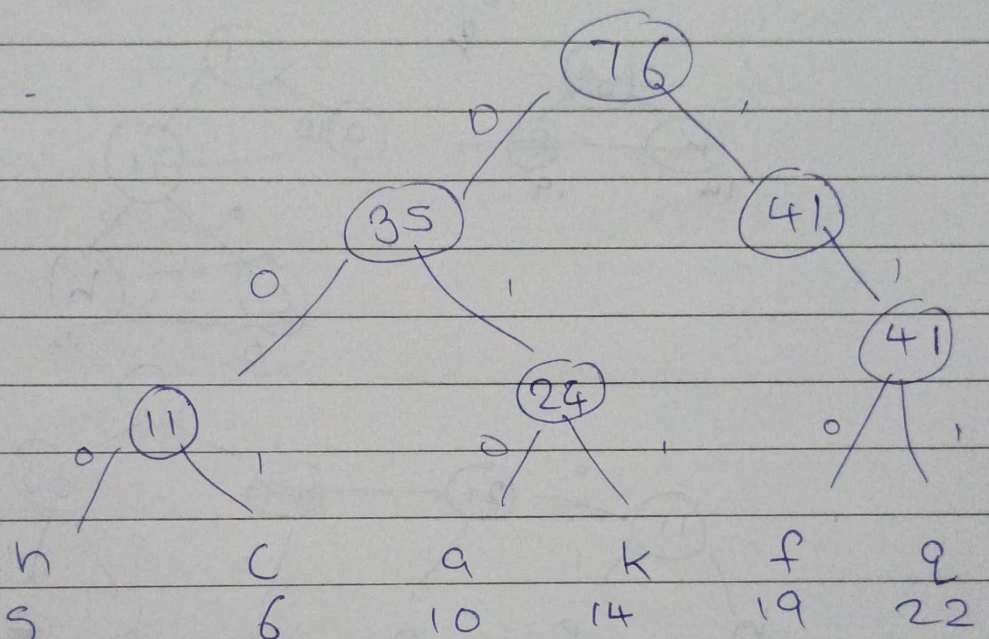
e = 10000 = 205 bits.

f = 00000

Q

a	c	f	h	k	q	→	h	c	a	k	f	q
10	6	19	5	14	22		5	6	10	14	19	22

fixed -



h = 000 k = 011

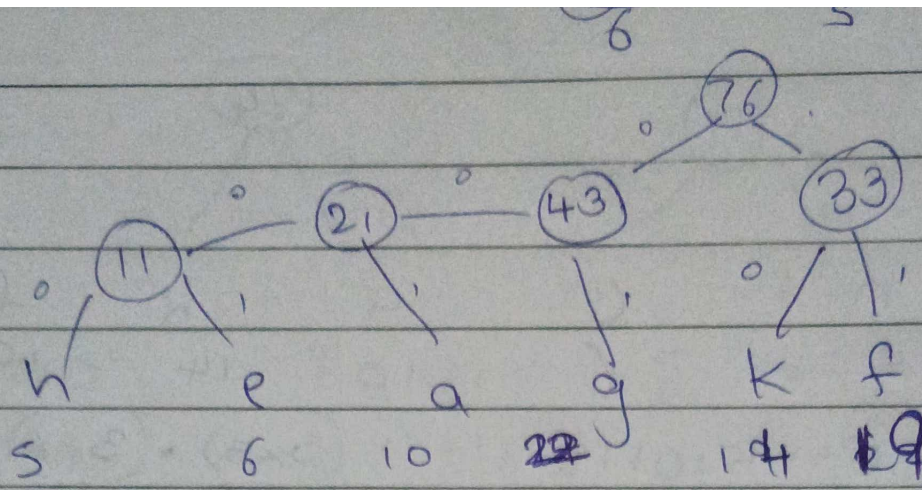
c = 001 f = 110

a = 010 a = 111

$= (3\times5) + (3\times6) + (3\times10) + (3\times14) + (3\times19) + (3\times22)$

$= 15 + 18 + 30 + 42 + 57 + 66$

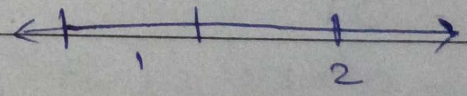
$= 228$



$= (5\times4) + (6\times4) + (10\times3) + (14\times2) + (19\times2) + (22\times2)$

$= 184$

Job Scheduling with deadlines

seq	Total profit
1	70
2	12
3	18
4	35
1 3	88
3 1	88
2 1	82
4 1	105
2 3	30
4 3	53

n	p_i	d_i
1	70	2
2	12	1
3	18	2
4	35	1