ASSIGNMENT - 2.

B008149Ib

- Madhan Thangavel.

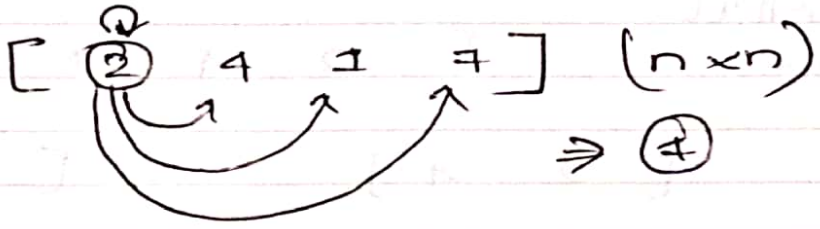
1) TO PROVE

$$n^2 > n \log n.$$

consider an algorithm to determine smallest among a given array.

```
small (a, s) {
    for(i=0; i<n; i++) {
        for (j=0; j<n; j++) {
            if (a[i] < a[j])
    }}}     small = a[i];
```

Example



It takes 16 comparisions to determine the value.

$$4 \times 4 = 16.$$

DIVIDE & CONQUER.

consider,

```
DAC ( a[], i, j )
{
  if small (a[i], i, j)
     return solution (a[i,j);
  else
     m= divide (a, i, j);
     b = DAC ( a, i, mid);
     c = DAC ( a, mid+1, j)
     d = combine (b, c)
  return (d);
}
```

EXAMPLE

[2 4 1 7]

[2 4] [1 7]

[2] [1]

[1]

It takes nlogn comparisions.

$n \log n$

$= 4 \log 4$

$= 4 \times 2 \log 2$

$= 8 < 16$

Thus by DAC approach, we get more efficient result.

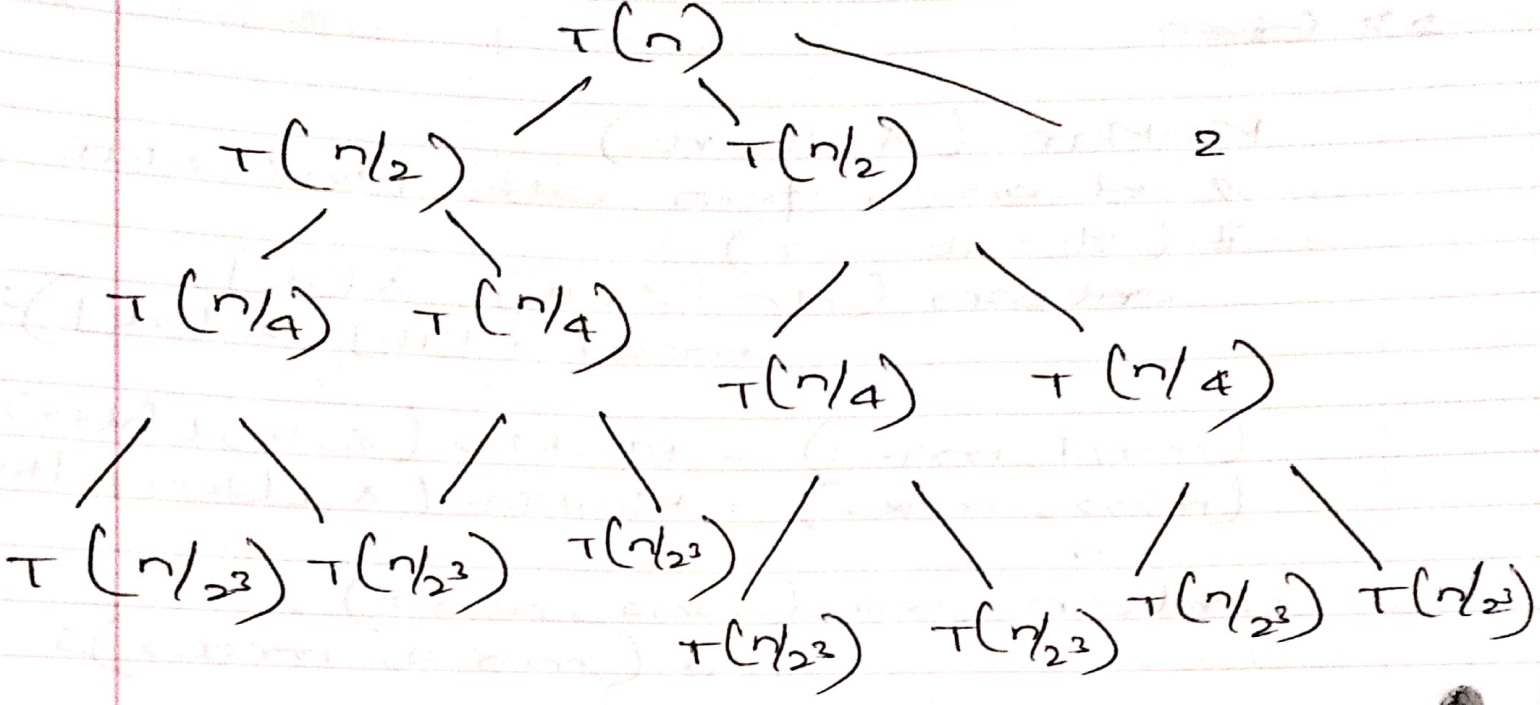
2) Given

```
MinMax (A, lt, rt)
// return a pair with min & max
if (rt - lt ≤ 1)
    return (min (A[lt], A[rt]),
            max (A[lt], A[rt]));

(min1, max1) = MinMax (A, lt, L(rt+lt)/2
(min2, max2) = MinMax (A, L(lt+rt)/2]+1, rt

return (min (min1, min2),
        max (max1, max2));
```

a) RECURRENCE RELATION

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

b) RECURRENCE TREE



// continue for k times,

$T(n/2^k)$

Let us assume, $n/2^k = 2$

$n = 2^{k+1}$

$k+1 = \log n$

$k = \log n - 1$

$$T(n) = n/2 + 2\left(2^{\log n - 1} - 1\right)$$

$$= n/2 + 2(n/2 - 1) \Rightarrow \frac{3n}{2} - 2$$

c) PROVE USING INDUCTION :

* Base case (1)
* Inductive Hypothesis (2)
* Inductive step. (3)

1) Base case, $n = 2$

$$T(2) = \frac{3 \times 2}{2} - 2 \Rightarrow 1.$$

Base case is True.

2) Inductive Hypothesis, $n = 2k$.

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

2) Inductive step, $n = 2(k+1)$

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

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

$$= 2^3 T(n/2^3) + 2^3 + 2^2 + 2 \cdots$$

$$\vdots$$

// for k times

$$\Rightarrow 2^k T(n/2^k) + 2^k \ldots + 2$$

Assuming $n/2^k = 2$

$$n = 2^{k+1}$$

$$k = \log n - 1$$

Now, $2\log n - 1 \cdot T(2) + 2\left(2^{\log_2 n - 1} - 1\right)$

$$= n/2 + n - 2 \Rightarrow 2n/2 - 2$$

sub $n = 2(k+1)$

$$\therefore \frac{2(2k+1)}{2} - 2 \Rightarrow 3(k+1) - 2$$

Hence $n = 2(k+1)$ holds true,

Therefore all the steps in Induction are true.

3> Given

A sorted Array of Integers A[1, ... n]

To prove:

for index i, A[i] = i

→ We can use binary searching to evaluate the result, as it uses Divide and conquer technique.

consider,

```
Algorithm (A[], start, end)
{
    Mid = (start + end) / 2;
    if (start == end)
        return (" index not found);
    else if (A[mid] == Mid)
        return A[Mid];
    else if (A[Mid] > Mid)
        Algorithm (A[], start, end) // Recursive
    else if (A[Mid] < Mid)
        Algorithm (A[], start, end) // Recursive
}
```

* Recursive R/W : $T(n) = \begin{cases} 1, & n=1 \\ T(n/2)+1, & n>1 \end{cases}$

$T(n) = T(n/2) + 1$ <TIME COMPLEXITY>

$T(n)$ is order of $\Theta(\log n)$.

4) Given:

an Array A[1, .. n]

prove: Majority element, if more than 1/2 of its entries are same.

consider,

```
Majority (A[], start, end)
{
    A[] = //Initialize :
    if ( n == 1)
        return A[1] ;
    Mid = (start + end ) / 2 ;

    A1 = Majority (A, start, Mid);
    A2 = Majority (A, Mid, end);

    if ( A1 == A2)
        return A1 ;

    sum_left  = Element count (A, A1);
    sum_right = Element count (A, A2);

  if (sum_left > Mid + 1)
      return A1 ;

  else if ( sum_right > Mid + 1)
      return A2 ;

  else
      return ("No element found");
}
```

```
Element count (A, B)
{
    count = 0;
    A[] = { 1 .... n }
    for (i=1; i<=n ; i++)
        count = count + 1 ;
    return count;
}
```

TIME COMPLEXITY, $T(n) = 2T(n/2) + 2n$

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

// Master's Theorem.

5> 1) $T(n) = 9 * T(n/3) + n^2 + 4$

Using Masters theorem.

$a = 9, \quad b = 3 \quad f(n) = n^{+2} \quad k = 2$

Therefore, $p = 0$

$$\log_b a = \log_3 9$$

$$= 2 \log_3 3 = 2$$

since $p = 0 > -1$

$$\Theta(n^k \log^{p+1} n) \Rightarrow \Theta(n^2 \log n)$$

* 2) $T(n) = 6 * T(n/2) + n^2 - 2$

Using Masters Theorem,

$a = 6, \quad b = 2 \quad f(n) = n^2 \quad k = 2$

Therefore, $p = 0$

$$\log_b a = \log_2 6 \approx 2.5$$

Where $\log_2 6 > k \;//\; 2.5 > 2$

Hence, $\theta(n^{\log_b a}) = \theta(n^{\log_2 6})$

$= \theta(n^3)$

3) $T(n) = 4 * T(n/2) + n^3 + 7$

Using Masters Theorem,

$a = 4, \quad b = 2, \quad f(n) = n^3$

$k = 3 \quad P = 0$

Therefore,

$\log_b a = \log_2 4 = 2 \log_2 2$

$= 2$

Hence $\log_2 4 < k \quad // \; 2 < 3$

Since $P = 0, \Rightarrow \theta(n^k \log^P n)$

$\Rightarrow \theta(n^3 \log^0 n)$

$\Rightarrow \theta(n^3)$

b) $T(n) - 8T(n-1) - 21T(n-2) + 18T(n-3)$ for $n > 2$

$T(0) = 0$

$T(1) = 1$

$T(2) = 2$

The given eqn be written as,

$(A^3 - 8A^2 + 21A - 18)\, T(n) = 0$

Factorizing, $(A^2 - 8A^2 + 21A - 18)$

$(A-2)(A-3)(A-3)$

Therefore

$A_1 = 2, \quad A_2 = 3, \quad A_3 = 3$

Using characteristic equation,

$T(n) = c_1 A_1^n + c_2 A_2^n + c_3 A_2^n \cdot n$

Given $T(0) = 0$.

$T(0) = c_1 \cdot 2^0 + c_2 \cdot 3^0 + c_3 \cdot 3^0 \cdot n$

$= c_1 + c_2$

Given

$$T(1) = 1$$

$$T(1) = c_1 \cdot 2^1 + c_2 \cdot 3^1 + c_3 \cdot 1 \cdot 3^1$$

$$= 2c_1 + 3c_2 + 3c_3$$

Given

$$T(2) = 2$$

$$T(2) = c_1 \cdot 2^2 + c_2 \cdot 3^2 + c_3 \cdot 2 \cdot 3^2$$

$$= 4c_1 + 9c_2 + 18c_3$$

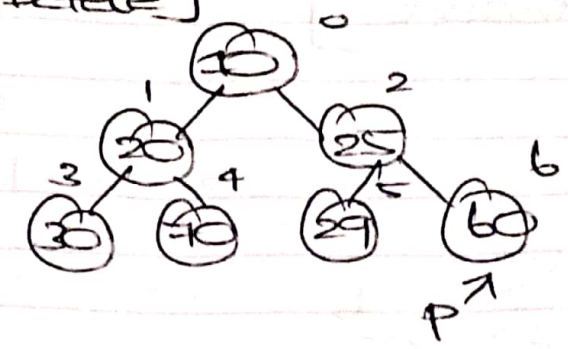
on solving $T(0)$, $T(1)$ & $T(2)$.
we get

$$c_1 = -4, \quad c_2 = 4, \quad c_3 = -1.$$

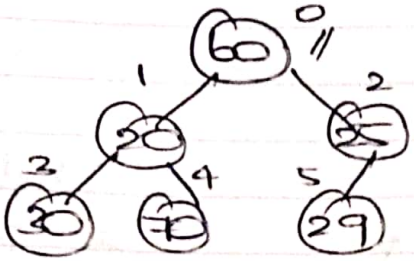
therefore,

$$T(n) = -4 \cdot 2^n + 4 \cdot 3^n - 3^n$$

(7) Heap [Delete]

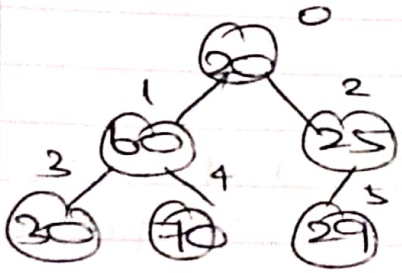


step 1



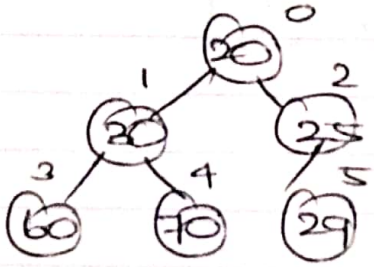
→ Remove small number & replace by 6th index.

step 2



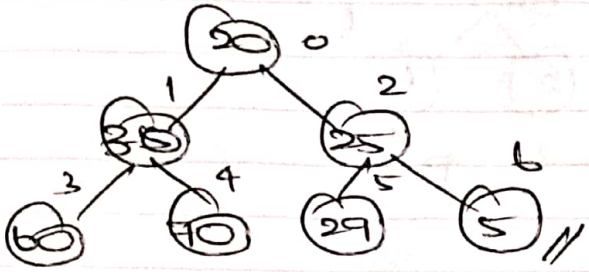
→ shifting to perform Minheap

step 3



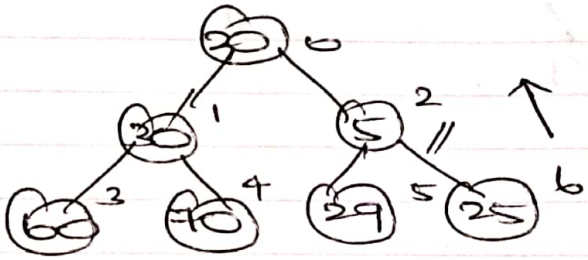
→ shifting to perform Minheap

* Step 0 [Insert]



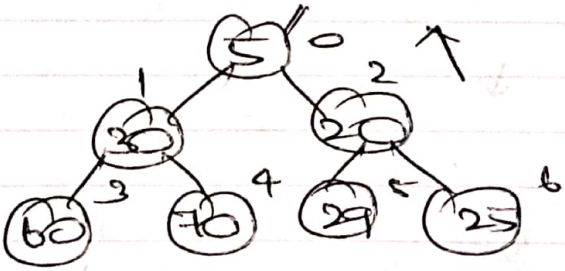
→ Add 5 to last node

Step 1



→ shift 6 to 2 position.

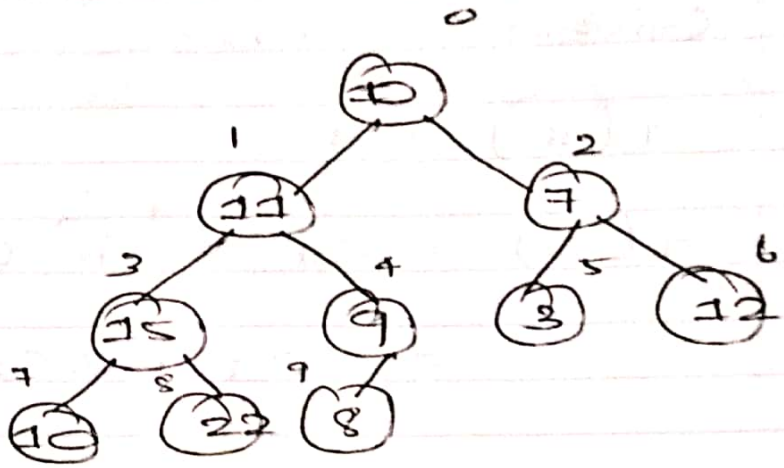
FINAL



→ shift 2 to 0th position.

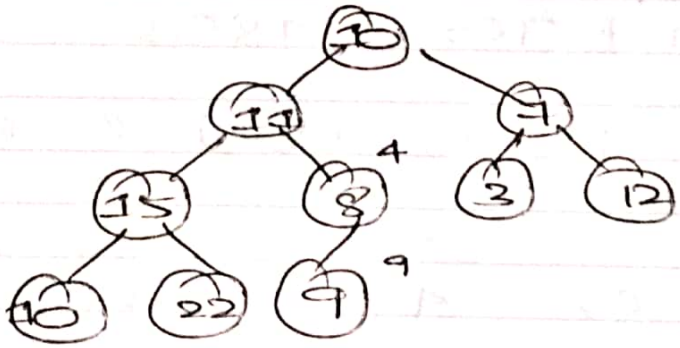
8>

a) Binary Tree :

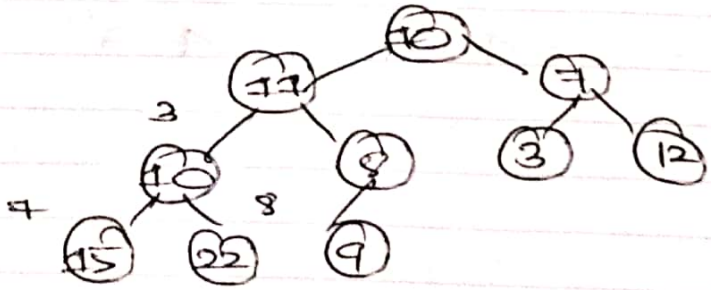


b) Min heap (Fast make heap)

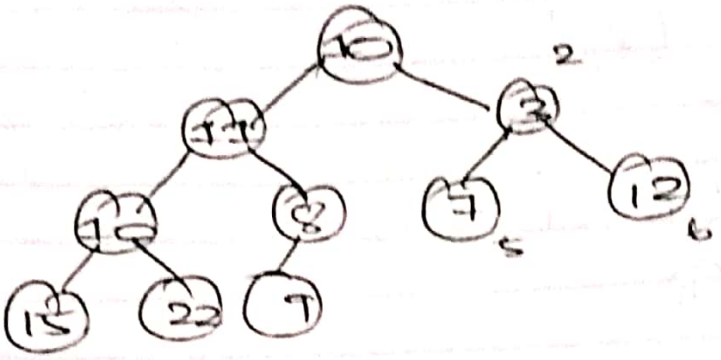
step 1 After siftdown 4



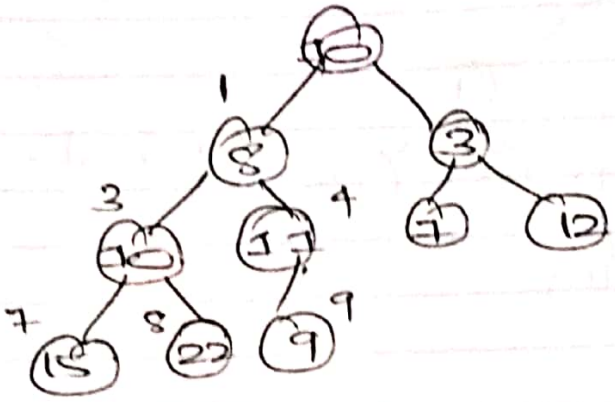
step 2 After siftdown 3



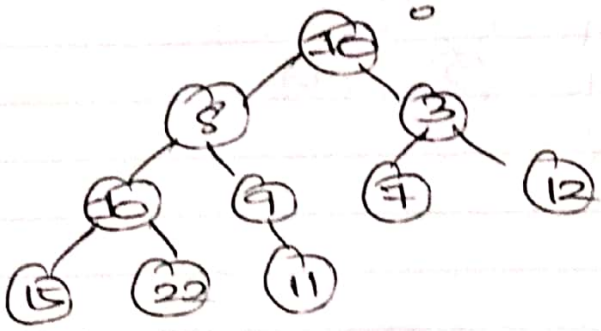
Step 3 After sift down 2



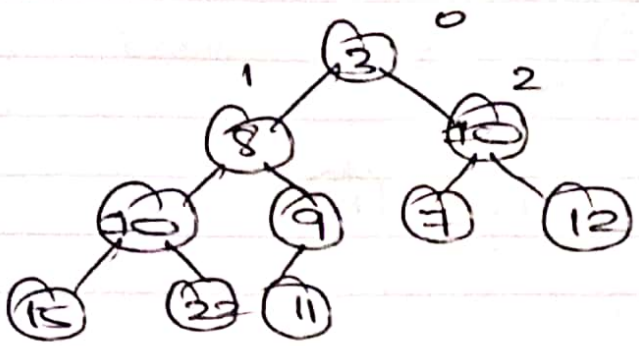
Step 4 After siftdown 1



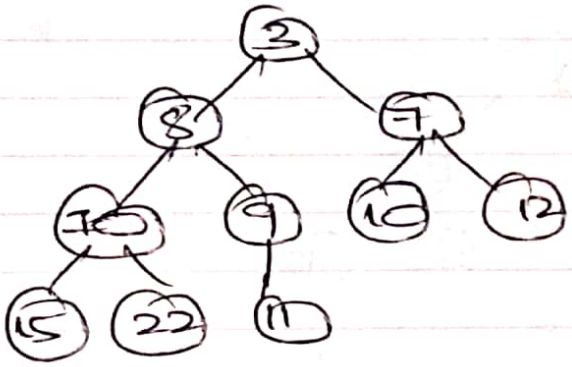
Step 5 After sift down 4



Step 6 After siftdown 0

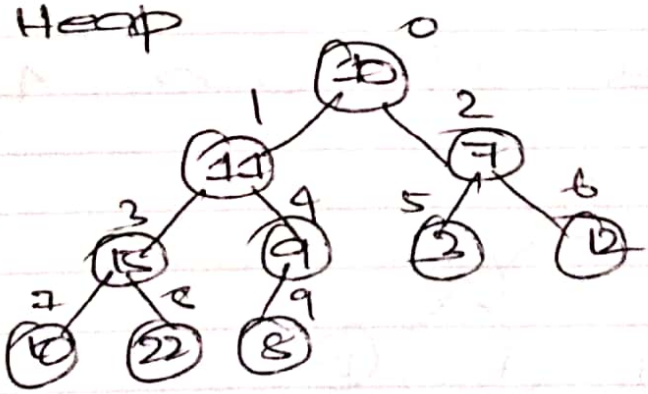


Step 7 After siftdown 2

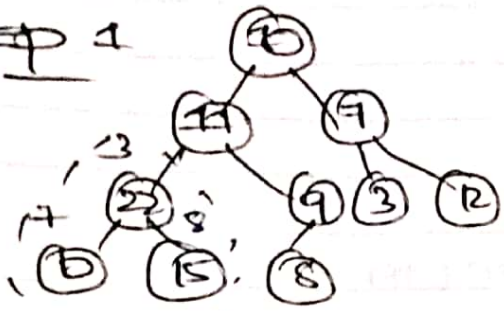


(Final Heap)

c) Max Heap

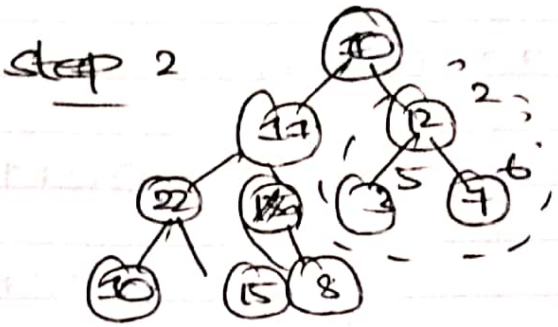


step 1



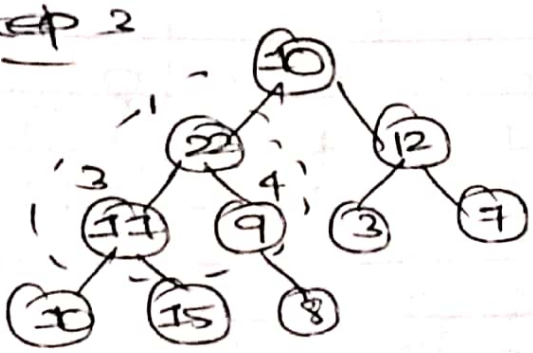
→ shiftup 3 & 8

step 2



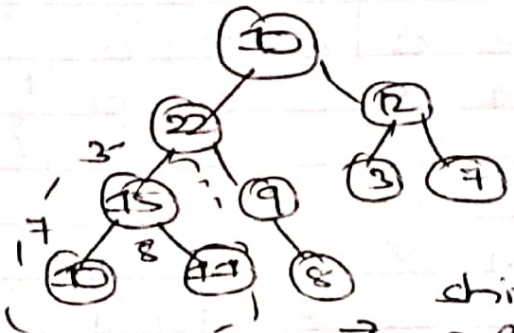
→ shiftup 2 & 6

step 3



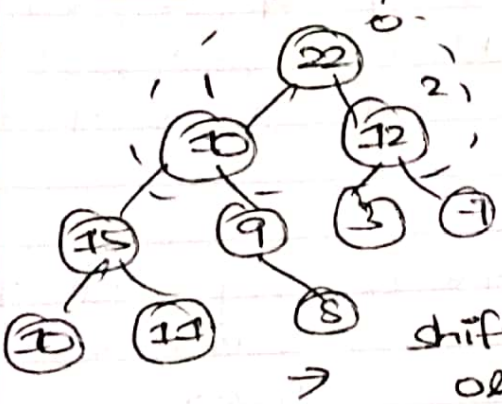
→ shiftup 1 & 3

step 4



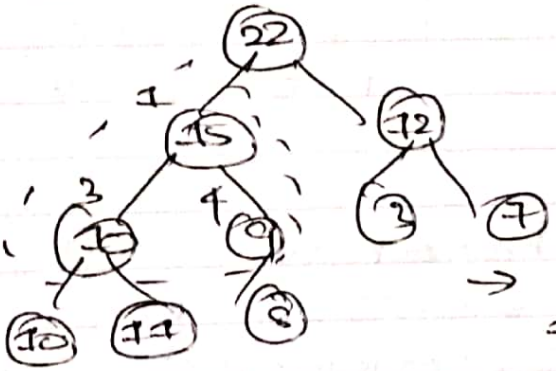
→ shiftup 3 & 8

step 5



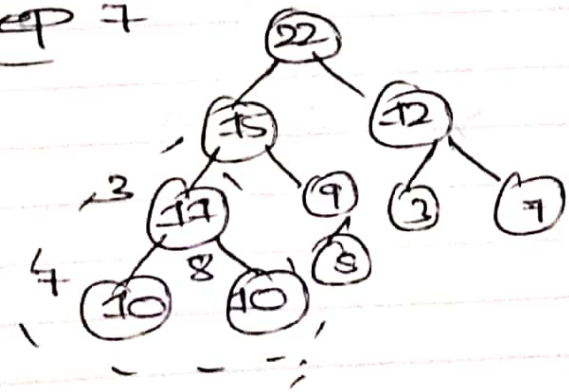
→ shiftup 0 & 1

step 6

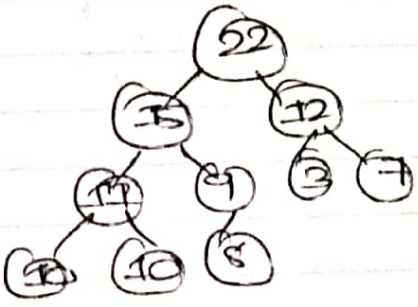


→ shiftup 1 & 3

step 7



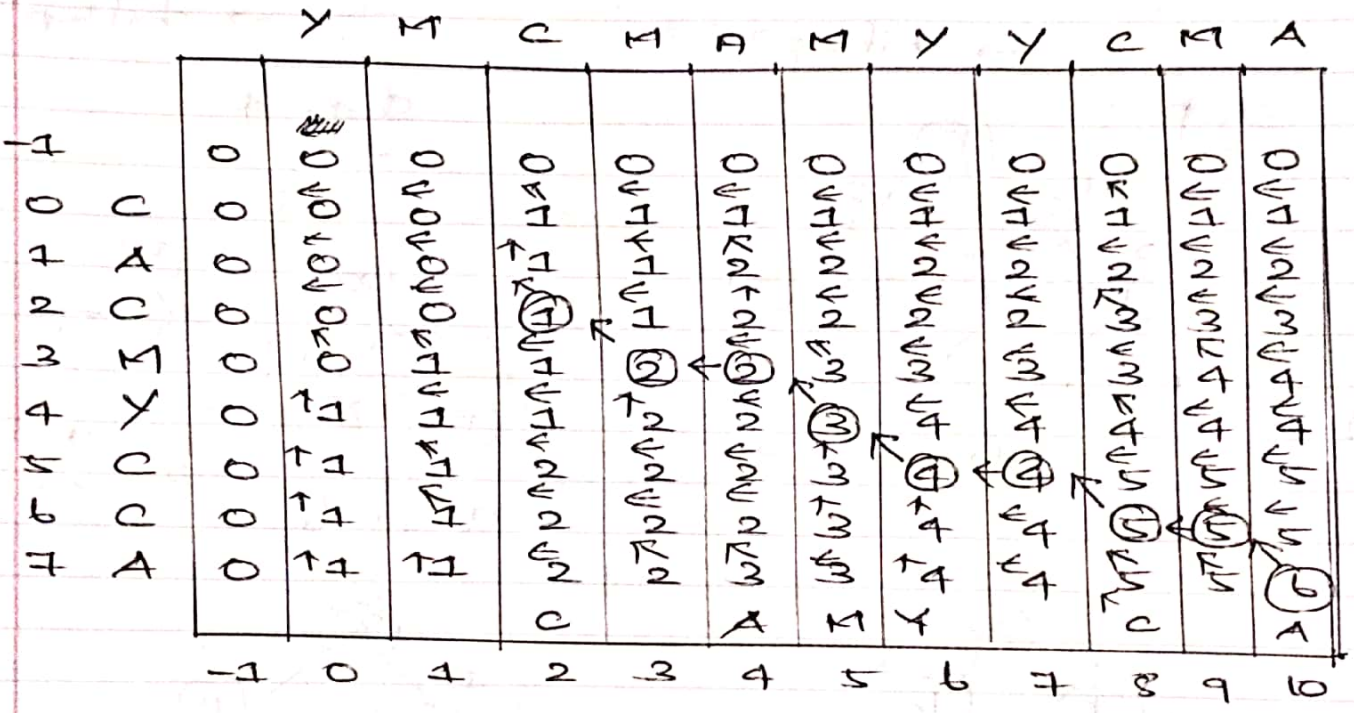
(Final)



9) Given :

X = CACMYCCA

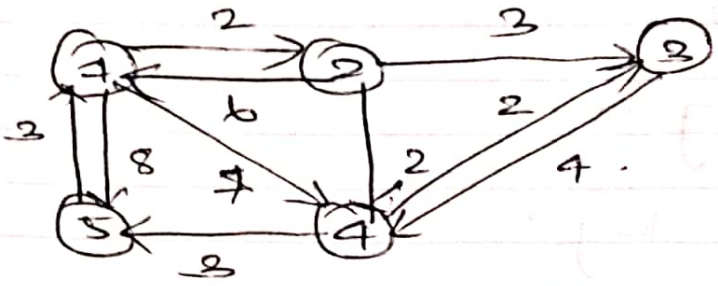
y = YMCMAMYYCMA



Therefore, the following diagonal arrows, the longest common subsequence of the given two strings is

C A M Y C A.

10) FLYOD'S ALGORITHM :



D Matrix for above diagram is.

	1	2	3	4	5
1	0	2	∞	1	8
2	6	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

P Matrix is

P =

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

step 1

$$D^k[i,j] = \min\{D^{k-1}[i,j],\ D^{k-1}[i,k] + D^{k-1}[k,j]\}$$

$$D^1[2,3] = \min\{3,\ 6+\infty\}$$

$$= 3$$

$A1 = D^1 =$

0	2	∞	1	8
6	0	3	2	14
∞	∞	0	4	∞
∞	∞	2	0	3
3	5	∞	4	0

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

step 2

$$D^2[1\ 3] = \min\{D^2(1,3),\ D^1(1,2) + D^1(2,3)\}$$

$$= \min\{\infty + 2+3\} = 5$$

D^2 =

0	2	5	1	8
6	0	3	2	14
∞	∞	0	4	∞
∞	∞	2	0	3
3	5	8	4	0

Φ =

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

Step - 3

$$D^3[1,2] = \text{Min}\{D^2(1,2), D^2(1,2) + D^2(2,1)\}$$

$$= \text{Min}\{2, 5+\infty\} = 2$$

D^3 =

0	2	5	1	8
6	0	3	2	14
∞	∞	0	4	∞
∞	∞	2	0	3
3	5	8	4	0

$$P = \begin{vmatrix} 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \end{vmatrix}$$

step 4

$$D^4(1,3) = \min \{ D^3(1,1), D^3(1,4), D^3(4,3) \}$$

$$= \min \{ 1+2 \}$$

$$= 3$$

$$D^4 = \begin{array}{|ccc|c|c|} \hline 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ \infty & \infty & 0 & 4 & 7 \\ \hline \infty & \infty & 2 & 0 & 3 \\ \hline 3 & 5 & 6 & 4 & 0 \\ \hline \end{array}$$

$$P = \begin{vmatrix} 0 & 0 & 4 & 0 & 6 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 4 & 4 & 4 & 0 \end{vmatrix}$$

Step 5

$$D^5[4,1] = \text{Min}\{D^5(4,1), D^4(4,5) + D^4(5,1)\}$$

$$= \text{Min}\{\infty, 3+3\} = 6$$

D^5 =

0	2	3	1	4
6	0	3	2	5
10	12	0	4	7
6	8	2	0	3
3	5	6	4	0

$$P = \begin{bmatrix} 0 & 0 & 4 & 0 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 5 & 5 & 0 & 0 & 4 \\ 5 & 5 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \end{bmatrix}$$

=

11) Sum of subsets problem,

* Recursive formula which can be associated with sum of subsets problem can be generalized as

S → set
n → number of elements

Therefore,

sum of subsets (s, n, sum)

= { sum of subsets (s, n-1, sum)

TIME COMPLEXITY is

$$T(n) = \begin{cases} 1, & n=0 \\ T(n+1)+n & n \geq 1 \end{cases}$$

* Algorithm (sumofsubsets)

```
sumofsubsets (s, n, sum)
{
    n = len(s)
    if sum == 0
        return true
    else if (sum != 0 && n == 0)
        return false

    if (s[n-1] > sum)
        return sumofsubsets (s, n-1, sum)
}
```

* EXAMPLE

Let us consider a set,

$S = \{ 1, 2, 5 \}$

Sum = 7

Creating a B-Matrix for dy. prog for the algorithm

Here if $P_i + P_j = 7$, return True. else return false.

$P_1 + P_2 = 1 + 2 = 3$

$\Rightarrow$ return False

$P_2 + P_2 = 2 + 5 = 7.$

$\Rightarrow$ return True

$P_1 + P_3 = 1 + 5 = 6$

$\Rightarrow$ return False

B - Matrix

→ sum

i ↓	1	2	3	4	5	6	7
1	✓	✓	x	x	x	x	x
2	x	x	x	x	✓	x	x
3	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x

✓ $\Rightarrow$ True x $\Rightarrow$ False