Example :-

You are given two numbers, how do you find the greatest common divisor (GCD) ?

→ Let two numbers are 4 and 2.

$$\begin{array}{r|l} 2 & 4 \\ \hline 2 & 2 \\ \hline & 1 \end{array} \qquad \begin{array}{r|l} 2 & 2 \\ \hline & 1 \end{array}$$

$\Rightarrow 2 \times 2 \times 1$

$\Rightarrow 2 \times 1$

$GCD = 2 \times 1 = 2$.

Euclid's Algorithm :-

```
function gcd (a, b)
        if b = = 0
                return a
        else
                return gcd (b, a mod b).
```

Downloaded from CSIT Tutor

* Algorithm Properties :-

1) Input / Output → There must be some input fr the standard set of inputs and the algorithm must produce output.

2) Definiteness → Each step must be clear and unambiguous.

3) Finiteness → Algorithm must terminate after finite time or steps.

4) Correctness → Correct set of output values must be produced from each set of objects.

5) Effectiveness → Each step must be carried out in finite time.

* Expressing Algorithms :-

There are many ways of expressing algorithms, like natural language, pseudocode and real programming language syntax etc.

Downloaded from CSIT Tutor

Random Access Machine Model (RAM model) :-

RAM model is a base model for our study of design and analysis of algorithms to have design and analysis in machine independent scenario. In this model, each basic operation, +, −, , takes one step. Similarly, loops and subroutines are not basic operations.

Best, Worst and Average Case :-

1). Best Case Complexity →

It gives lower bound on the running time of the algorithm for any instance of input. This indicates that the algorithm can never have lower running time than best case for particular class of problem.

2) Worst Case Complexity :→

It gives upper bound on the running time of the algorithm for all the instances of the input. This ensures that no input can overcome the running time limit posed by worst case complexi

3) Average Case Complexity →

It gives average number of steps required on any instance of input.

Downloaded from CSIT Tutor

Q. Write an algorithm to display the fibonacci series.

1. Input the value of no. of values i.e. n.
2. Give inputs $a=1$ and $b=1$.
3. Assign c as $c=a+b$ and display c.
4. Assign $a=b$, $b=c$ and decrease n by 1.
5. While $n != 0$, goto step 3.
6. End of program.

Correctness :-

Value of $n=5$

		Display
for $n=5 \Rightarrow$	$a=1, b=1, c=2$	2
for $n=4 \Rightarrow$	$a=1, b=2, c=3$	2 3
for $n=3 \Rightarrow$	$a=2, b=3, c=5$	2 3 5
for $n=2 \Rightarrow$	$a=3, b=5, c=8$	2 3 5 8
for $n=1 \Rightarrow$	$a=5, b=8, c=13$	2 3 5 8 13
for $n=0 \Rightarrow$	Terminate.	

Efficiency :-

Time Complexity $\Rightarrow T(n) = O(n)$

Space Complexity $\Rightarrow S(n) = O(4) = O(1)$

[$\because$ variables used = 4, we used $O(4)$. but to represent any constant value, we use $O(1)$].

Downloaded from CSIT Tutor

Some of the formulas that are helpful :-

Exponents :-

1). $x^a \cdot x^b = x^{ab}$.

2). $x^a / x^b = x^{a-b}$.

3). $(x^a)^b = x^{ab}$

4). $x^n + x^n = 2x^n$.

5). $2^n + 2^n = 2^{n+1}$.

Logarithms :-

1) $\log a^b = \dfrac{\log_c b}{\log_c a}$; $c > 0$

2) $\log ab = \log a + \log b$.

3) $\log a/b = \log a - \log b$.

4) $\log(a^b) = b \cdot \log a$.

5) $\log x < x$ for all $x > 0$.

6) $\log 1 = 0$, $\log 2 = 1$, $\log 1024 = 10$.

7) $a^{\log_b n} = n^{\log_b a}$.

Downloaded from CSIT Tutor

Series :-

1) $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

2) $\sum_{i=0}^{n} a^i \leq 1/(1-a)$; if $0 < a < 1$.

$= \dfrac{a^{n+1} - 1}{a - 1}$; else.

3) $\sum_{i=1}^{n} i = n(n+1)/2$.

4) $\sum_{i=0}^{n} i^2 = n(n+1)(n+2)/6$.

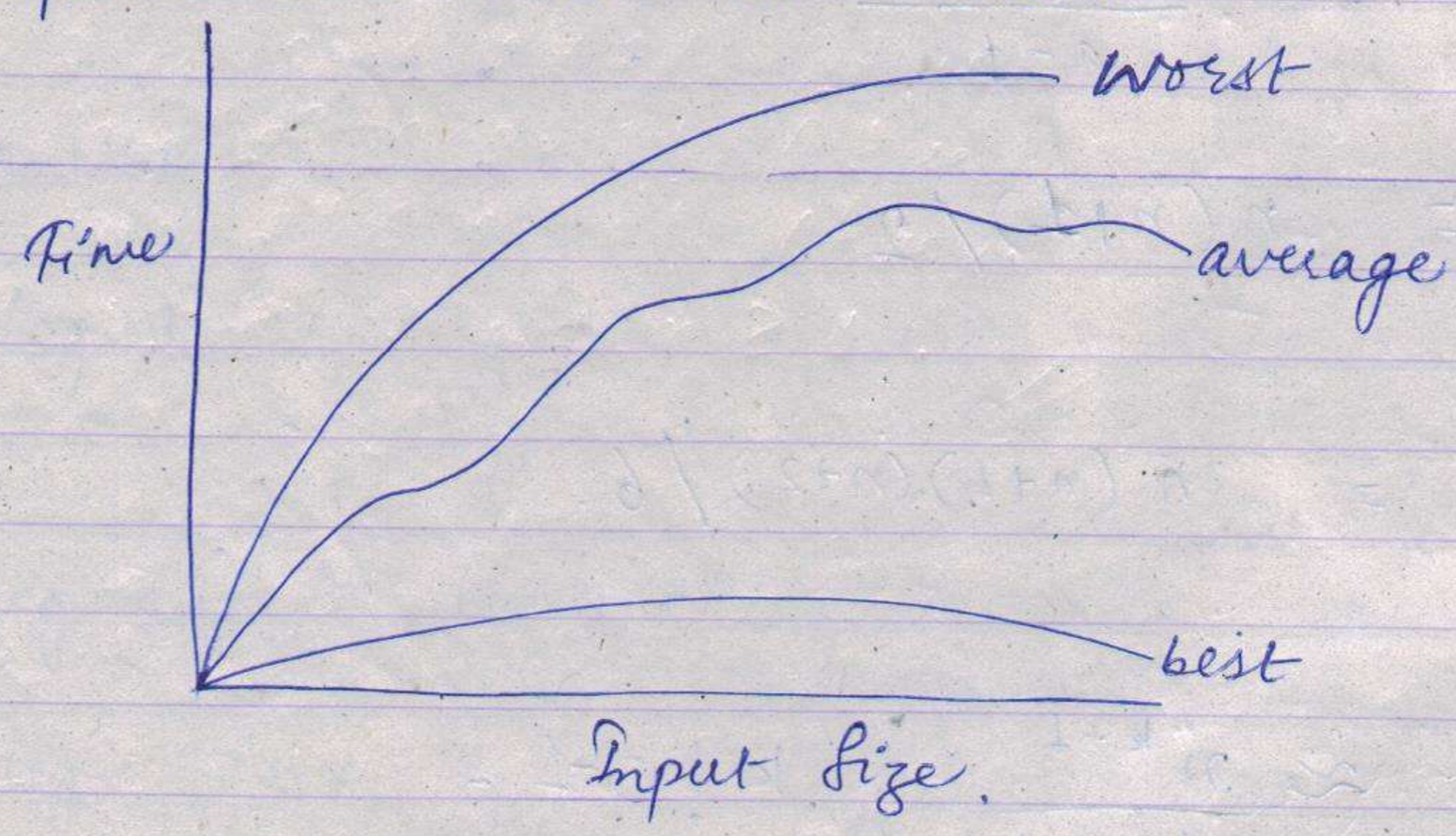
5) $\sum_{i=0}^{n} i^k \approx \dfrac{n^{k+1}}{|k+1|}$; $k! = -1$.

$\sum_{j=0}^{\frac{n-1}{2}} j^2 \Rightarrow \dfrac{(n-1)^2}{2} \Rightarrow \dfrac{n^2 - 2n + 1}{2} \Rightarrow n^2$

Downloaded from CSIT Tutor

Asymptotic Notation :-

Complexity Analysis of an algorithm is very hard if we try to analyze exactly. So, if we analyze the algorithm in terms of bound, i.e. upper and lower bound, then it would be easier. For this purpose, the concept of asymptotic notation is used.



Q. Why we concentrate on worst case mostly? Why not best case and average case?

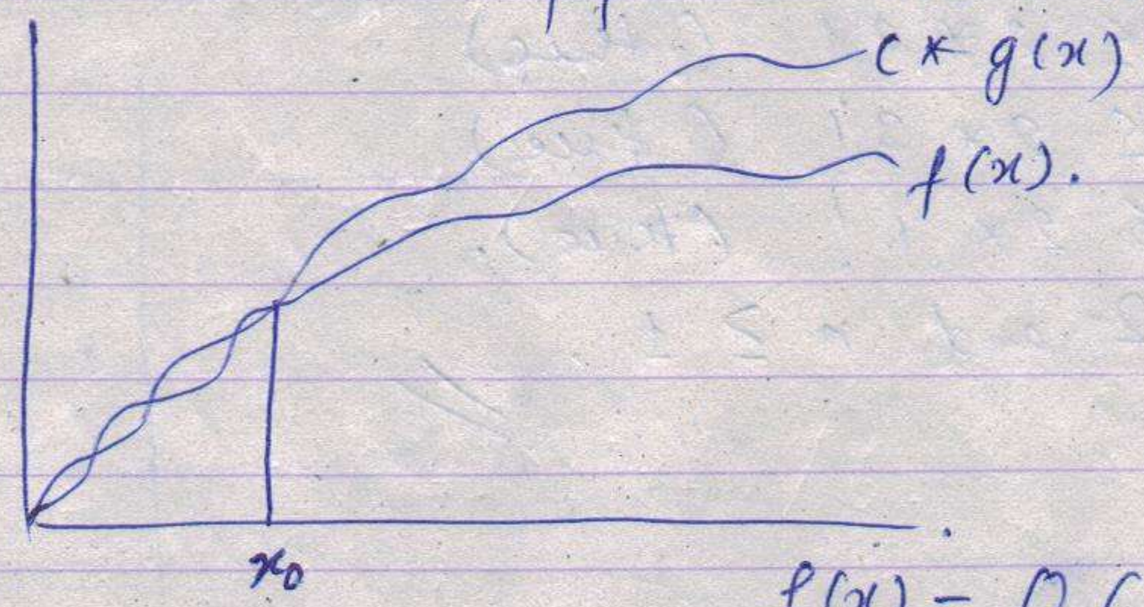
→ We concentrate on worst case because the probability of achieving the best case and average case is obvious in any problem. We select the worst case in any problem as, even if the worst case occurs, we can solve it.

Downloaded from CSIT Tutor

Big oh (O) notation :-

When we have only asymptotic upper bound then we use O notation. A function $f(x) = O(g(x))$ (read as $f(x)$ is big oh of $g(x)$), iff there exists two positive constants c and x_0, such that, for all $x > x_0$, $0 \le f(x) \le c * g(x)$.

Here, $g(x)$ is an upper bound of $f(x)$.



$f(x) = O(g(x))$.

For Fibonacci Series,

we take $n = x$ and then,

$f(x) = x + 2$. [∵ 2 steps required for executio before n steps].

Let $g(x) = x$

So, $c = 3$.

So, $f(x) \le c * (g(x))$.

or, $x + 2 \le 3 * x$

or, $2 + x \le 3x$ (true).

for $x_0 = 1$, $3 \le 3$ (true)

for $x_0 = 2$, $4 \le 6$ (true).

∴ $c = 3$, $g(x) = x$ and x_0

Downloaded from CSIT Tutor

Q. Show that $2^n = O(n!)$

→ Here,

$f(n) = 2^n$, $g(n) = n!$

So, $f(n) \leq c * g(n)$.

Now, for $c = 1$,

$n = 1 \Rightarrow 2^1 \leq 1 \times 1!$ (false).

So, for $c = 2$,

$n = 1 \Rightarrow 2 \leq 2 * 1!$ (true)

$n = 2 \Rightarrow 2^2 \leq 2 * 2!$ (true)

$n = 3 \Rightarrow 2^3 \leq 2 * 3!$ (true)

$n = 4 \Rightarrow 2^4 \leq 2 * 4!$ (true).

$\therefore$ $c = 2$ and $n \geq 1$.

Downloaded from CSIT Tutor

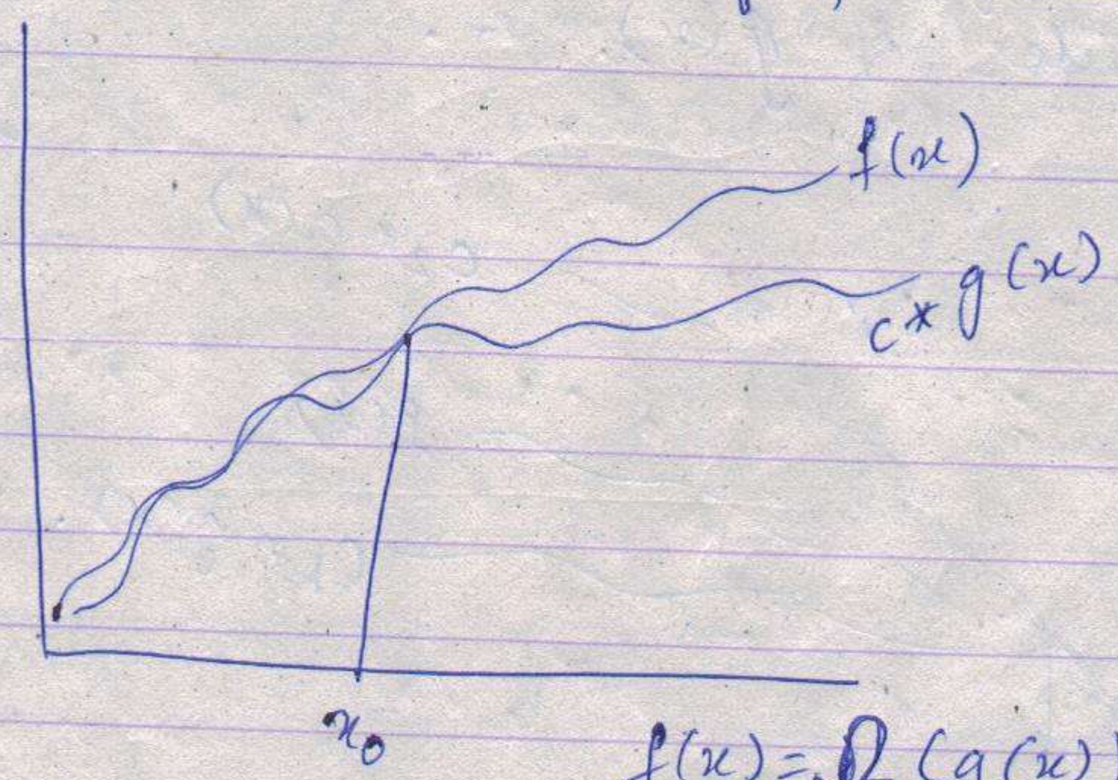
21st Feb, 2014

Big Omega (Ω) notation :-

Big Omega notation gives asymptotic lower bound. A function $f(x) = \Omega(g(x))$ (read as $f(x)$ is big omega of $g(x)$) iff there exists two positive constants c and x_0 such that for all $x \geq x_0$,

$$0 \leq c * g(x) \leq f(x).$$

Here, $g(x)$ is a lower bound of $f(x)$.



$f(x) = \Omega(g(x))$.

$f(x) = x + 2$
$g(x) = x$.
Then, $c = 1$.

for $x = 1$, $c * g(x) \leq f(x) \Rightarrow 1 \times 1 \leq 1 + 2$ (true).
for $x = 2$, $c * g(x) \leq f(x) \Rightarrow 1 \times 2 \leq 2 + 2$ (true).
for $x = 3$, $c * g(x) \leq f(x) \Rightarrow 1 \times 3 \leq 3 + 2$ (true).
for $x = 4$, $c * g(x) \leq f(x) \Rightarrow 1 \times 4 \leq 4 + 2$ (true).

So, $f(x) = \Omega(g(x))$.

So, $c = 1$ and $x_0 \geq 1$.

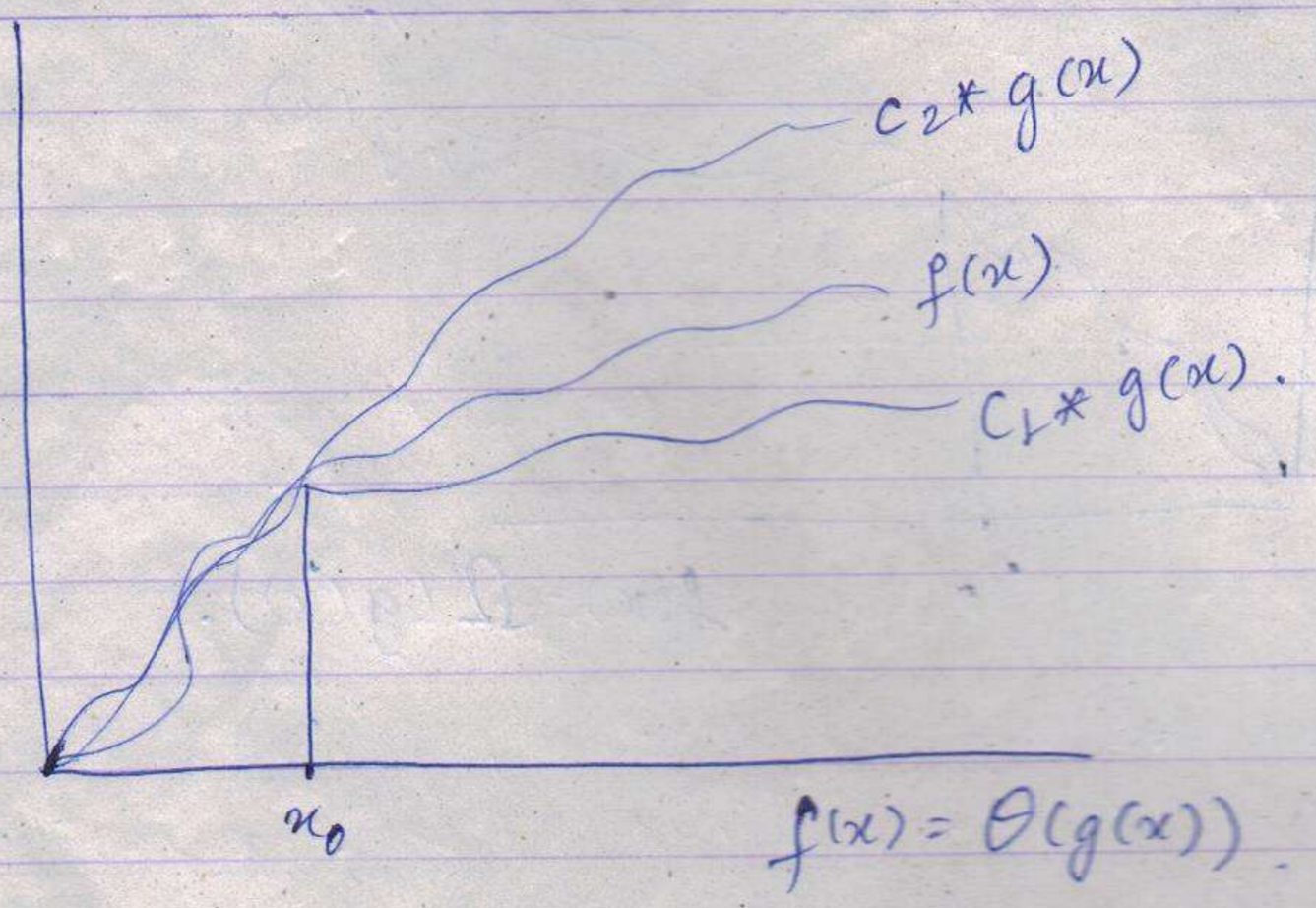
Downloaded from CSIT Tutor

Big theta (Θ) notation :-

When we need asymptotically tight bound then we use Θ notation. A function $f(x) = \Theta(g(x))$ (read as, $f(x)$ is big theta of $g(x)$) iff there exists three positive constants, c_1, c_2 and x_0 such that for all $x \geq x_0$,

$$0 \leq c_1 * g(x) \leq f(x) \leq c_2 * g(x).$$

Here, $f(x)$ is order of $g(x)$.



$f(x) = x + 2$

$g(x) = x$ Show that $f(x) = \Theta(g(x))$.

Here,

$c_1 = 1$, $c_2 = 3$.

$c_1 * g(x) \leq f(x) \leq c_2 * g(x) \Rightarrow 1 * x \leq x+2 \leq 3 * x$

for $x = 1 \Rightarrow 1 \leq 3 \leq 3$ (true) $\therefore f(x) = \Theta(g(x))$.

for $x = 2 \Rightarrow 2 \leq 4 \leq 6$ (true) $\therefore c_1 = 1$, $c_2 = 3$, and

for $x = 3 \Rightarrow 3 \leq 5 \leq 9$ (true) $x_0 \geq 1$.

Downloaded from CSIT Tutor

(little oh)
Small oh (o) :- $f(x) < c * g(x)$.
Small omega (ω) :- $c * g(x) < f(x)$.
(little omega)

9. If $f(x) = 3x^2 + 2x + 1$ and $g(x) = x^2$. Show that $f(x) = \Theta(g(x))$.

Soln :-

Here, $c_1 = 1$, $c_2 = 6$. (Suppose).

Then, $c_1 * g(x) \leq f(x) \leq c_2 * g(x)$.

or, $1 * x^2 \leq 3x^2 + 2x + 1 \leq 6 \times x^2$

for $x = 1$, $1 \leq 6 \leq 6$ (true)

for $x = 2$, $4 \leq 17 \leq 24$ (true)

for $x = 3$, $9 \leq 34 \leq 54$ (true).

$\therefore$ $f(x) = \Theta(g(x))$. for $c_1 = 1$, $c_2 = 6$ and $x_0 \geq 1$.

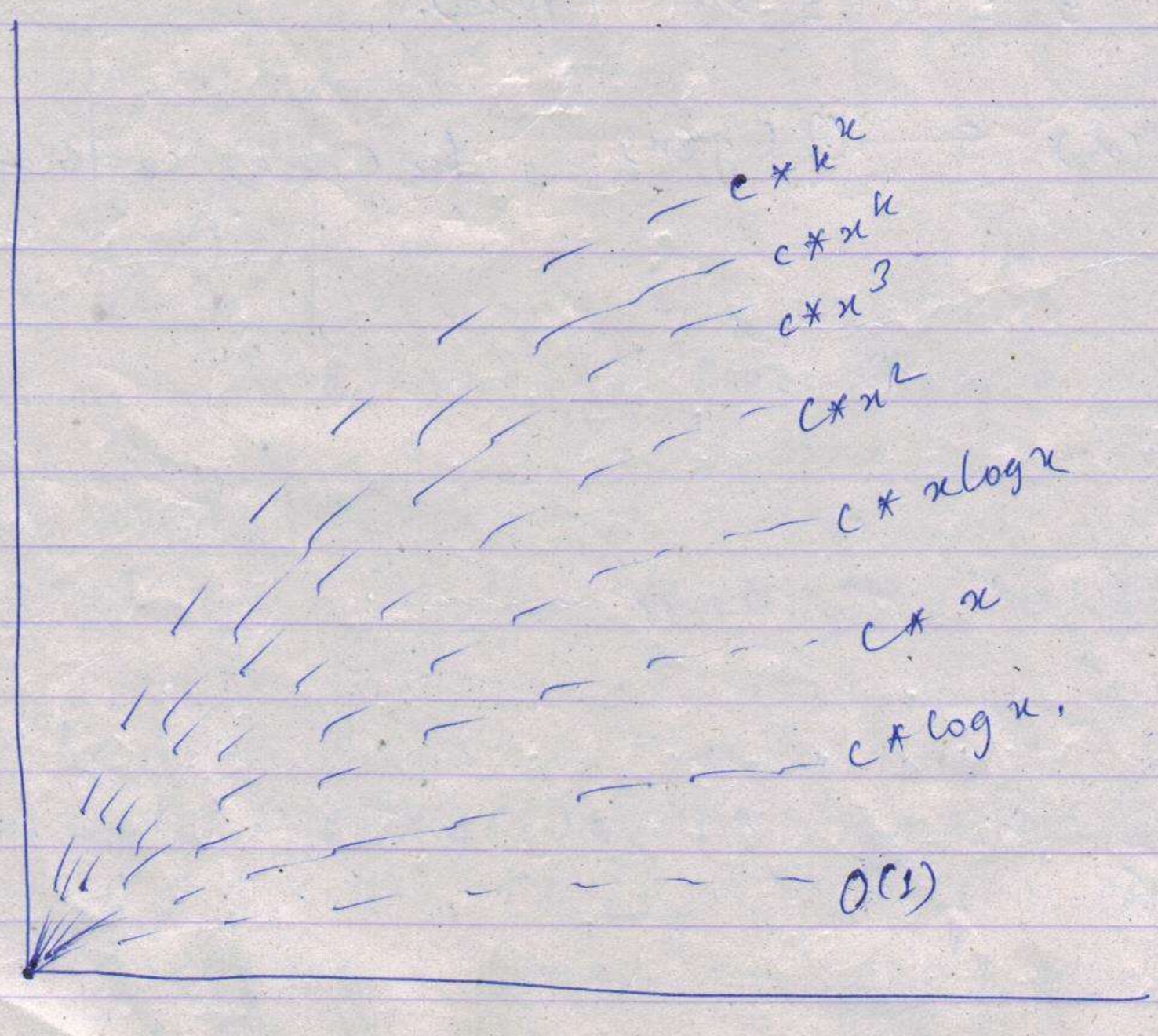
Downloaded from CSIT Tutor

Some commonly used function in algorithm analysis:-

$f(x) = O(1)$ constant.
$f(x) = c * \log x$ logarithmic.
$f(x) = c * x$ linear.
$f(x) = c * x\log x$ linearithmic.
$f(x) = c * x^2$ quadratic.
$f(x) = c * x^3$ cubic
$f(x) = c * x^k$ polynomial in k.
$f(x) = c * k^x$ exponential.

order of growth:-

$O(1) < c * \log x \leq c * x \leq c * x\log x \leq c * x^2 < c * x^3 < c * x^k < c * k^x$.



Downloaded from CSIT Tutor

NOTE :- $f(x) = x^2$, $g(x) = x$

$= O(x^2)$ $= O(x)$

So, $f(x) + g(x) = O(x^2)$. [take max value]

$f(x) * g(x) = O(x^3)$ [take $O(f(x) * g(x))$]

Example :-

Algorithm :- Insertion Sort.

```
1)  InsertionSort (A [ ])
2)  {
3)      for (i=1 ; i<n ; i++)
4)      {
5)          x = A [i];
6)          j = i-1 ;
7)          while ( j≥0 and A [j] > x)
8)          {
9)              A [j+1] = A [j];
10)             j = j-1 ;
11)         }
12)         A [j+1] = x;
13)     }
14) }
```

array →	0	1	2	3	4	5
	2	1	4	3	6	5

$i = 1$.

$x = A[i] = A[1] = 1$

$j = i - 1 = 0$

$0 \geq 0$ and $A[0] = 2 > 1$ (true)

$A[j+1] = A[1] = 2$

$j = j - 1 = -1$

$A[0] = 1$. So, swapping of $a[0]$ and $a[1]$.

Downloaded from CSIT Tutor

test for array :-

[57, 90, 80, 48, 35, 91, 1, 83, 32, 53].

A[i] 0 1 2 3 4 5 6 7 8 9

$i = 1$.

$x = A[1] = 90$.

$j = i - 1 = 0$

$0 \geq 0$ and $A[0] > 90$ i.e. $57 > 90$ (false).

$A[j+1] = A[1] = 90$.

So, [57, 90, 80, 48, 35, 91, 1, 83, 32, 53].

$i = 2$.

$x = A[2] = 80$

$j = 1$

$1 \geq 0$ and $A[j] = A[1] = 90 > 80$ (true).

$A[j+1] = A[2] = A[1] = 90$

$j = j - 1 = 1 - 1 = 0$

~~$A[j+1] = A[2] = 9$~~

$0 \geq 0$ and $A[j] = A[0] = 57 > 80$ (false)

$A[1] = 80$.

So, [57, 80, 90, 48, 35, 91, 1, 83, 32, 53].

$i = 3$

$x = A[3] = 48$

$j = i - 1 = 2$.

$2 \geq 0$ and $A[j] = A[2] = 90 > 48$ (true)

$A[j+1] = A[3] = A[2] = 90$.

$j = j - 1 = 2 - 1 = 1$.

$1 \geq 0$ and $A[j] = A[1] = 80 > 48$? (true

[57, 80, 90, 90, 35, 91, 1, 83, 32, 53].

Downloaded from CSIT Tutor

$A[j+1] = A[2] = A[1] = 80$

$j = j-1 = 1-1 = 0.$

So, ~~A[~~ [57, 80, 80, 90, 35, 91, 1, 83, 32, 53].

$0 \geq 0$ and $A[j] = A[0] = 57 > 48$? (true).

$A[j+1] = A[1] = A[0] = 57.$

$j = j-1 = 0-1 = -1.$

$A[j+1] = A[0].$

So, [48, 57, 80, 90, 35, 91, 1, 83, 32, 53].

$i = 4.$

$x = A[4] = 35.$

$j = i-1 = 3$

$3 \geq 0$ and $A[j] = A[3] = 90 > 35$? (true)

$A[j+1] = A[4] = A[3] = 90.$

Downloaded from CSIT Tutor

Time Complexity:- Insertion Sort.

Insertion Sort (A[·])	Cost	Times
1) for (i=1; i<n; i++)	c_1	
2) {	~~c_2~~	
3) x = A[i];	c_3	n
4) j = i-1;	c_4	n
5) while (j >= 0 and A[j] > x)	c_5	$\sum_{i=1}^{n-1} i$,
6) {	~~c_6~~	–
7) A[j+1] = A[j];	c_7	$\sum_{i=1}^{n-2} i$,
8) j = j-1;	c_8	$\sum_{i=1}^{n-2} i$
9) }	~~c_9~~	–
10) A[j+1] = x;	c_{10}	$n-1$
11) }	~~c_{11}~~	–

[Note:- in 5:- $\sum_{i=1}^{n-1} i$ is $1+2+3+\ldots+n-1$.

in 7, 8:- $\sum_{i=1}^{n-2} i$ is $1+2+3+\ldots+n-2$.]

1) Worst Case:-

$$T(n) = c_1 * n + c_3 * (n-1) + c_4 * (n-1) + c_5 * \left(\sum_{i=1}^{n-1} i\right) + c_7 * \left(\sum_{i=1}^{n-2} i\right) + c_8 * \left(\sum_{i=1}^{n-2} i\right) + c_{10} * (n-1).$$

$$= c_1 * n + c_3 * (n-1) + c_4 * (n-1) + c_5 * \frac{n(n-1)}{2} + c_7 * \frac{(n-1)(n-2)}{2} + c_8 * \frac{(n-1)(n-2)}{2} + c_{10} * (n-1).$$

Downloaded from CSIT Tutor

$= O(n) + O(n) + O(n) + O(n^2) + O(n^2) + O(n^2) + O(n).$
$= O(n^2).$

2) Best Case :-

$T(n) = c_1 * n + c_3 * (n-1) + c_4 * (n-1) + c_5 * (n-1) + c_{10} * (n$
$= \Omega(n) + \Omega(n) + \Omega(n) + \Omega(n) + \Omega(n)$
$= \Omega(n).$

[Note :- $c_5 * (n-1)$
→ while is been $(n-1)$ times here as only ~~only~~ one time is checked as it is in sorted order in best case].

3) Average Case :-

$T(n) = O(n^2/2)$
$\approx \Theta(n^2).$ //

[use average of worst case & best case].
↳ $\frac{O(n^2) + \Omega(n)}{2} = \frac{O(n^2)}{2}$
$= O(n^2)$

Downloaded from CSIT Tutor

Merge Sort :-

Algorithm :-

```
MergeSort(A, l, h)
{
    if (l < h)
    {
        q = ⌊(l+h)/2⌋;
        MergeSort(A, l, q);
        MergeSort(A, q+1, h);
        Merge(A, l, q, h);
    }
}

Merge(A, low, mid, high).
{
    l = m = low;
    k = mid + 1;
    while ((l <= mid) and (k ≤ high))
    {
        if (A[l] <= A[k])
        {   B[m] = A[l];
            l = l + 1;
        }
        else
        {   B[m] = A[k];
            k = k + 1;
        }
        m = m + 1;
    }
```

Downloaded from CSIT Tutor

```
if ( l < mid)
{
    for ( i = k ; i < high ; i++)
    {
        B[m] = A[i];
        m = m+1;
    }
}
else
{
    for ( i = l ; i < mid ; i++)
    {
        B[m] = A[i];
        m = m+1;
    }
}
for ( j = low ; j ≤ high ; j++)
    A [j] = B[j] ;
}
```

Downloaded from CSIT Tutor

Trace for

0	1	2	3	4	5	6	7	8	9
57	90	80	48	35	91	1	83	32	53

Here, $l = 0$, $h = 9$.

So, if $(l < h) \Rightarrow$ <u>true</u>

$q = \lfloor 0+9/2 \rfloor = \lfloor 4.5 \rfloor = 4$.

So, partition is as:-

0	1	2	3	4	5	6	7	8	9
57	90	80	48	35	91	1	83	32	53

1st | 2nd

In 1st,

$l = 0$, $h = 4$

if $(l < h) \Rightarrow$ <u>true</u>

$q = \lfloor 4/2 \rfloor = 2$.

0	1	2	3	4
57	90	80	48	35

3rd | 4th

3rd: $l = 0$, $h = 2$, $q = 1$

4th: $l = 3$, $h = 4$, $q = 3$

0	1	2	3	4
57	90	80	48	35

$l = 0$, $h = 1$, $q = 0$

In 2nd,

$l = 5$, $h = 9$

if $(l < h) \Rightarrow$ <u>true</u>

$q = \lfloor 7 \rfloor = 7$

5	6	7	8	9
91	1	83	32	53

5th | 6th

5th: $l = 5$, $h = 7$, $q = 6$

6th: $l = 8$, $h = 9$, $q = 8$

5	6	7	8	9
91	1	83	32	53

$l = 5$, $h = 6$, $q = 5$

0	1	2	3	4	5	6	7	8	9
57	90	80	48	35	91	1	83	32	53

Downloaded from CSIT Tutor

1) $l = m = 0$, high = 9, mid = 4

k = mid + 1 = 5.

while ((0 <= 4) and (5 <= 9)) true

{ if(A[0] <= A[5]) true.

{

B[m] = B[0] = A[l] = A[0] = 57

l = l + 1 = 0 + 1 = 1;

}

m = m + 1 = 0 + 1 = 1;

}

B[m] = [57]

A[ ] = [90 80 48 35 91 1 83 32 53]

2) $l = m = 1$; high = 9, mid = 4

k = mid + 1 = 5

while ((1 <= 4) and (5 <= 9)) true

{

if (A[1] <= A[5]) true

{

B[m] = B[1] = A[l] = A[1] = 90

l = l + 1 = 1 + 1 = 2;

}

m = m + 1 = 1 + 1 = 2;

}

B[m] = [57 90]

A[ ] = [80 48 35 91 1 83 32 53]

Downloaded from CSIT Tutor

$2T(n-1)+1$

Q. Show that time complexity of Tower of Hanoi is 2^n.

Tower of Hanoi :-

i) if $n==1$, move disk 1 from source to destination.

ii) if $n \geq 2$, move disk $n-1$ from source to temp using dest. as temp. peg.

iii) move $n-1$ from ~~dt~~ temp. to dest using source as temp. peg.

$$T(n) = 1 \quad ; n = 0, 1$$

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

or, $T(n) = 2T(n-1) + 1$.

Iteration method :-

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

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

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

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

$$= 2^3 T(n-3) + 2^2 + 2^1 + 2^0$$

$$= 2^3 (2T(n-4) + 1] + 2^2 + 2^1 + 2^0$$

$$= 2^4 T(n-4) + 2^3 + 2^2 + 2^1 + 2^0.$$

$\vdots$

= to k times,

$$= 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \cdots 2^0.$$

If $k = n-1$,

$$= 2^k T(1) + 2^{k-1} + \cdots 2^0$$

$$\Rightarrow 2^k + 2^{k-1} + \cdots + 2^0$$

$$\Rightarrow 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2^0$$

Downloaded from CSIT Tutor

$2^{n-1} + 2^{n-2} + \cdots 2^1 + 2^0$

$\cancel{= 2^{n-1} + 1}$

Now $\because \sum_{i=0}^{n} 2^i = 2^{n+1} - 1 = 2^0 + 2^1 + \cdots 2^{n-1}$,

$= 2^{(n-1)+1} - 1$ $\quad [\because n = n-1$ in n

$= 2^n - 1 \leq 2^n$

$\therefore T(n) = O(2^n)$.

Downloaded from CSIT Tutor

Recurrence Relation:

In the computer algorithms, recurrence relation can model the complexity of the running time particularly divide and conques algorithms. There are many techniques for solving recurrence relation.

1) Iteration Method.
2) Substitution Method.
3) Recurrence Tree Method.
4) Master's Method.

1) Iteration Method :-

Iteration method needs more experience in solving algebra. Here we are concerned with following steps:-

i) Expand the relation so that summation dependent on n is obtained.

ii) Bound the summation.

Eg.:- $T(n) = 2T(n/2) + 1$, $T(1) = 1$.

$= 2[2T(n/4) + 1] + 1$

$= 2^2 T(n/4) + 2^1 + 2^0$

$= 2^2 [2T(n/8) + 1] + 2^1 + 2^0$

$= 2^3 T(n/8) + 2^2 + 2^1 + 2^0$

$\vdots$

$=$ upto k-terms,

$= 2^k T(n/2^k) + 2^{k-1} + \cdots + 2^1 + 2^0$

If $2^k = n$,

Downloaded from CSIT Tutor

$= 2^k \cdot T(1) + 2^{k-1} + \dots + 2^1 + 2^0$

$= 2^k + 2^{k-1} + \dots + 2^1 + 2^0$

$= n + 2^k \times \frac{1}{2} + 2^k \times \frac{1}{2^2} + \dots + 2^1 + 2^0$

$= n + n \cdot \frac{1}{2} + n \cdot \frac{1}{2^2} + \dots + 2^1 + 2^0$

$\sum_{i=0}^{k} 2^i = 2^k + 2^{k-1} + \dots + 2^1 + 2^0$

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

$= 2^k \cdot 2 - 1$

$= 2^{\log n} \cdot 2 - 1$

$= n \cdot 2 - 1$

$= 2n - 1$

$\leq 2n$

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

$2^{\log n} \cdot 2 - 1$

n

$2^k = n$

$\log 2^k = \log n$

$2^n = \log n$

$\log 2^n = k$

$k = \log n$

Downloaded from CSIT Tutor

Eg. (2) $T(n) = T(n/3) + n \quad ; n > 1$

$T(1) = 1 \quad ; n = 1.$

Now, $T(n) = T(n/3) + n.$

$= [T(n/3^2) + n/3] + n$

$= T(n/9) + n/3 + n$

$= [T(n/27) + n/3^2] + n/3 + n$

$= T(n/27) + n/9 + n/3 + n$

$\vdots$

upto k-terms,

$= T(n/3^k) + n/3^{k-1} + n/3^{k-2} + \dots + n/3^0$

Put $3^k = n$,

$= T(n/n) + n/3^{k-1} + n/3^{k-2} + \dots + n/3^0$

$= T(1) + n/3^{k-1} + n/3^{k-2} + \dots + n/3^0$

$= \frac{n}{3^{k-1}} + \frac{n}{3^{k-2}} + \dots + \frac{n}{3^0} + 1.$

$= \frac{3^k}{3^{k-1}} + \frac{3^k}{3^{k-2}} + \dots + \frac{3^k}{3^0} + 3^0$

$= 3^{k-k+1} + 3^{k-k+2} + \dots + 3^{k-0} + 3^0$

$= 3^1 + 3^2 + \dots 3^k + 3^0$

$= 3^0 + 3^1 + 3^2 + \dots 3^k$

$= 3^{k+1} - 1 \qquad \left[\sum_{i=0}^{k} 3^i\right]$

$= 3^k \cdot 3^1 - 1$

$= n \cdot 3 - 1 \le 3n \approx O(n).$

Downloaded from CSIT Tutor

Now, $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$

So, $= \frac{3^{n+1}-1}{3-1}$

$= \frac{3^k \cdot 3 - 1}{2}$

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

$\leq 3n/2$

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

Downloaded from CSIT Tutor

2) Substitution Method :-

In this method, to obtain lower or upper bound,

i) We guess the solution.

ii) Prove that solution to be true by mathematical induction.

Example :-

$$T(n) = 1 \quad ; n = 1$$
$$T(n) = 2\,T(n/2) + n \quad ; n > 1.$$

Suppose,

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

i.e. $T(n) = c * n\log n.$

$$T(n) = 2\,T(n/2) + n.$$
$$= 2\left(c * \frac{n}{2} \log \frac{n}{2}\right) + n$$
$$= \cancel{2}c * \frac{n}{\cancel{2}} \log \frac{n}{2} + n$$
$$= c * n \log \frac{n}{2} + n$$
$$= cn\,(\log n - \log 2) + n$$
$$= cn \log n - cn \log 2 + n$$
$$= cn \log n - cn * 1 + n$$
$$= cn \log n - cn + n$$
$$= cn \log n - n\,(c-1)$$
$$\leq cn \log n \qquad [\because cn\log n - n(c-1) \leq cn \log n]$$
$$\therefore T(n) \leq cn \log n$$
$$\therefore T(n) = O(n \log n).$$

Downloaded from CSIT Tutor

So, checking for the boundary condition,
we know, $T(1) = 1$.

So,

$T(1) \leq c * n \log n$

or, $1 \leq c * 1 \log 1$

or, $1 \leq c * 1 * 0$ $[\because \log 1 = \log 2^0 = 0]$

or, $1 \leq c * 0$

$\therefore 1 \leq 0$ [false]

So, not valid for $n = 1$.

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

$= 2\,T(1) + 2$

$= 4$

$T(2) \leq c * n \log n$

or, $4 \leq c * 2 \log 2$

or, $4 \leq c * 2$ [true for $c \geq 2$]

$\therefore$ valid for $T(2)$, i.e. $n = 2$.

$T(3) = 2\,T(3/2) + 3$

$= 2T(1) + 3$

$= 5$

$T(3) \leq c * n \log n$

or, $5 \leq c * 3 \log 3$

or, $5 \leq c * 3 * 1 \ldots$

$\therefore 5 \leq c * 3$ [true for $c \geq 2$].

$\therefore$ valid for $n = 3$, i.e. $T(3)$.

Hence for $n \geq 2$ and $c \geq 2$, the condition satisfies

$\therefore T(n) = O(n \log n)$.

Downloaded from CSIT Tutor

9. $T(n) = 2T(n/2) + 1$.

Soln :-

Suppose $T(n) = O(n)$

i.e. $T(n) = c * n$

$$T(n) = 2T(\frac{n}{2}) + 1$$
$$\leq 2[c * n/2] + 1$$
$$\leq 2c \cdot \frac{n}{2} + 1$$
$$\leq cn + 1.$$

[$\leq cn$ is not possible].

Stop. and modify the relation by subtracting some constant value.

So, $T(n) = c*n - b$.

So,

$$T(n) = 2T(n/2) + 1$$
$$\leq 2[c * n/2 - b] + 1$$
$$\leq 2c\frac{n}{2} - 2b + 1$$
$$\leq cn - 2b + 1$$
$$\leq cn - (2b - 1).$$
$$\leq cn$$

$\therefore T(n) \leq cn$

$\therefore T(n) \leq O(n)$.

Now, checking boundary condition,

$T(1) = 1$.

So, $T(1) \leq c * n$

or, $1 \leq c * 1$ (true).

Downloaded from CSIT Tutor

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

$= 2 \cdot T(1) + 1$

$= 3$

So, $T(2) \leq c * n$

or, $3 \leq c * 2$ [true] (for $c = 2$)

$T(3) = 2T(3/2) + 1$

$= 2T(1) + 1$

$= 3$

So, $T(3) \leq c * n$

or, $3 \leq c * 3$ [true].

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

$= 2T(2) + 4$

$= 6 + 4$ [$\because T(2) = 3$]

$= 7$

$\therefore T(4) \leq c * n$

or, $7 \leq c * 4$ (true)

$\therefore$ for $n \geq 1$ and $c = 2$, the time complexi

true.

$\therefore T(n) = O(n)$. //

Downloaded from CSIT Tutor

Q. $T(n) = T(n/3) + n$.
$T(n) = 1$; $n = 1$.

Soln :-

Suppose, $T(n) = O(n)$
i.e. $T(n) \leq c * n$.

Now,

$$T(n) = T(n/3) + n$$
$$\leq c * n/3 + n$$
$$\leq \frac{cn}{3} + n$$

~~$\leq cn$ is not possible.~~

$$\leq n\left(\frac{c}{3} + 1\right)$$
$$\leq dn \quad [\text{where } d > (c/3 + 1)]$$

$\therefore T(n) \leq dn$; where $d > c$ and d and c are constan

So, Checking boundary condition,

$T(1) = 1$.
$T(1) \leq c * 1$
$\therefore 1 \leq c * 1$ (true)

$T(2) = T(2/3) + 2$
$= T(0) + 2$
$= 1 + 2$
$= 3$

$T(2) \leq c * 2$
$\therefore 3 \leq 2 * c$ [for $c = 2$]

$T(3) = T(1) + 3$
$= 4$

$T(3) \leq c * 3$
$\therefore 4 \leq c * 3$ (true) for $c = 2$

$\therefore T(n) = O(n)$
for $n \geq 1$ and $c = 2$

Downloaded from CSIT Tutor

Tower of Hanoi :-

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

$T(n) = 1 \quad ; \quad n = 1.$

Soln :-

Suppose, $T(n) = O(2^n)$

i.e. $T(n) \leq c * 2^n.$

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

$$\cancel{= 2[2^n - 1] + 1}$$

$$\leq 2[c * 2^{n-1}] + 1$$

$$\leq 2c * 2^{n-1} + 1$$

$$\leq 2c * \frac{2^n}{2} + 1$$

$$\leq c * 2^n + 1.$$

Stop and modify the relation by subtracting some constant value.

So, $T(n) = c * 2^n - b.$

So, $T(n) = 2T(n-1) + 1.$

$$\leq 2[c * 2^{n-1} - b] + 1$$

$$\leq 2c * \frac{2^n}{2} - 2b + 1$$

$$\leq c * 2^n - (2b - 1).$$

$$\leq cn^n$$

$\therefore T(n) \leq cn^n$

$\therefore T(n) = O(2^n).$

Downloaded from CSIT Tutor

Checking for boundary conditions

$T(1) = 1$

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

~~$\text{or, } 1 = 2T(0)+1$~~

$\therefore T(1) \leq c * 2^n$

$\text{or, } 1 \leq c * 2^1$ (true)

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

$= 2T(1)+1$

$= 3$

$\therefore T(2) \leq c * 2^n$

$\text{or, } 3 \leq c * 2^2$

$\therefore 3 \leq c * 4$ (true)

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

$= 2T(2)+1$

$= 7$

$\therefore T(3) \leq c * 2^n$

$\text{or, } 7 \leq c * 2^3$

$\therefore 7 \leq c * 8$ (true)

$\therefore$ for $c = 1$ and $n \geq 1$, it is true as,

$T(n) = O(2^n)$.

Downloaded from CSIT Tutor

Changing Variables :-

$$T(n) = 2T(\sqrt{n}) + 1.$$

Suppose,

$n = 2^m$; $m = \log n$; $n^{1/2} = 2^{m/2}$

$T(n) = 2T(n^{1/2}) + 1$

$T(2^m) = 2T(2^{m/2}) + 1$

$T(2^m) = S(m)$ (suppose).

$$\boxed{S(m) = 2S(m/2) + 1.}$$

In the form of $T(n) = 2T(n/2) + 1$

~~So, $T(n) = \theta(n)$.~~

So, $T(n) \le c * n$.

(Solve the process) and then $T(n) = O(n)$

Then, $S(m) = O(m)$ [as $T(n) = O(n)$].

or, $T(2^m) = O(m)$.

$\therefore T(n) = O(\log n)$. //

[NOTE :- We take 2^m i.e. power of 2 as the process is Divide & Conquer method. So, in this process, we use 2^m for division process is halved].

Downloaded from CSIT Tutor

3) Recursion Tree Method :-

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

$T(n) = 1.$

$\Rightarrow \quad T(n) = T(n/2) + 1$

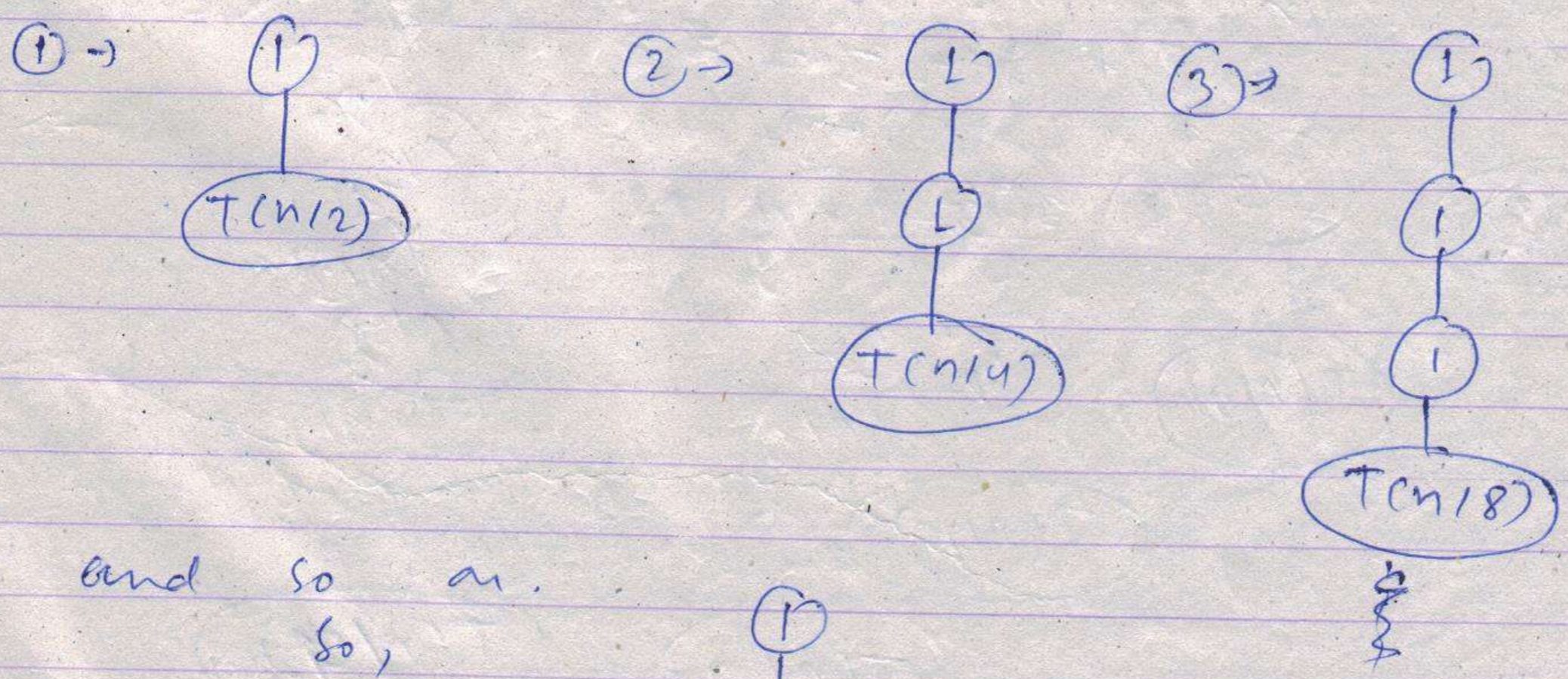
$= T(n/4) + 1 + 1$

$= T(n/8) + 1 + 1 + 1.$

$\vdots$

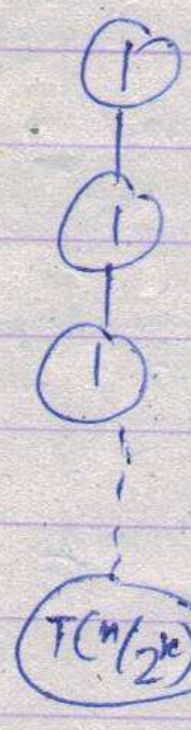
$=$ upto k-terms,

$= T(n/2^k) + 1 + 1 \ldots + 1 + 1 + 1.$



and so on.

So,



$T(n/2^k) = T(1)$ for $2^k = n$. $\Rightarrow k = \log n$

$1 + 1 + 1 + 1 + 1 + \ldots 1 = k * 1 = k = \log n$

$\therefore T(n) = O(\log n).$

Downloaded from CSIT Tutor

$$\frac{a^{n+1}-1}{a-1}$$

$O(n)$

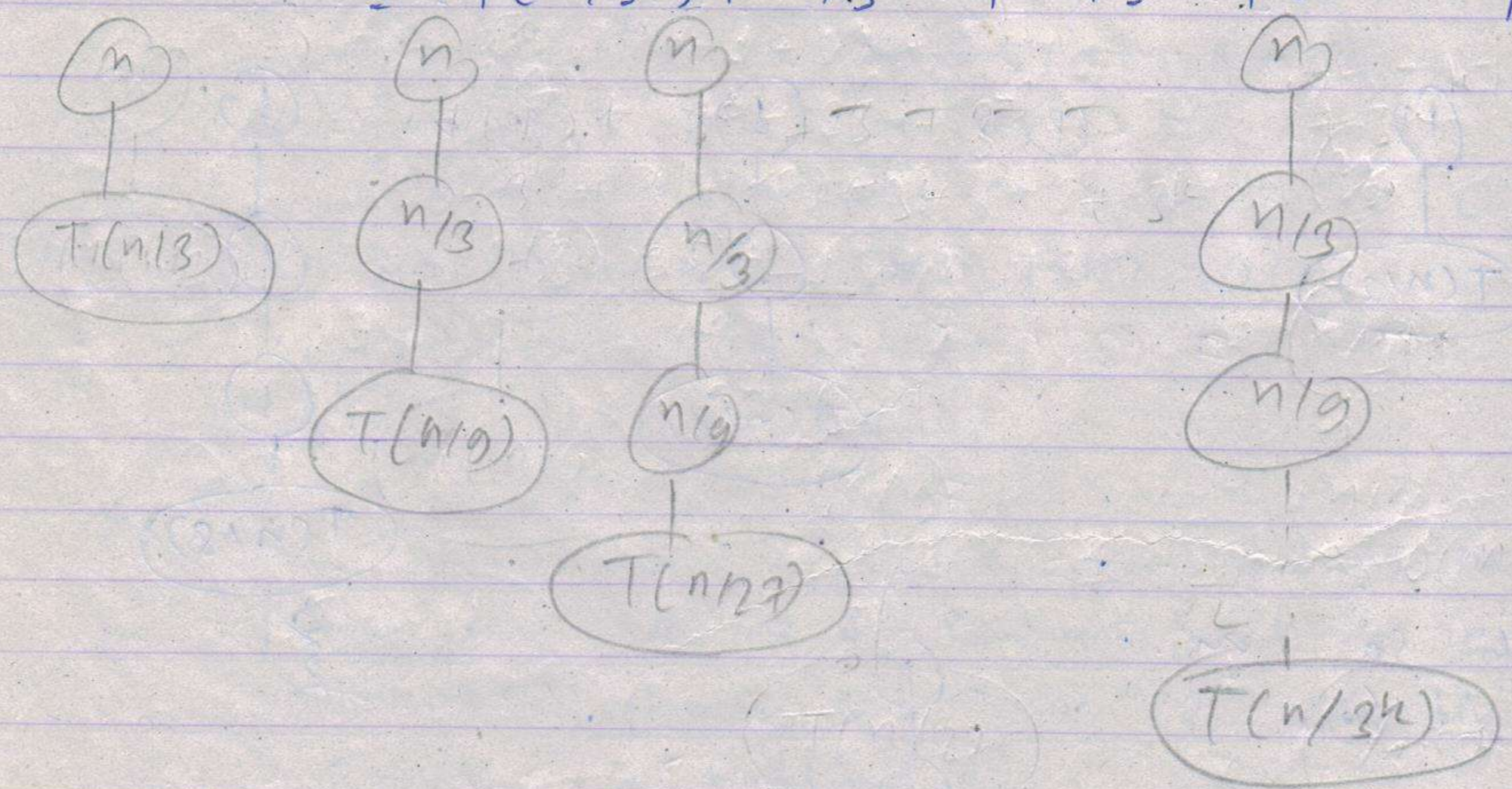
Eg. (2) $T(n) = T(n/3) + n$

$T(n) = 1$.

$$T(n) = T(n/3) + n$$
$$= T(n/9) + n/3 + n$$
$$= T(n/27) + n/9 + n/3 + n$$
$$= T(n/81) + n/27 + n/9 + n/3 + n.$$
$$\vdots$$

upto k-terms,

$$= T(n/3^k) + n/3^{k-1} + n/3^{k-2} + \dots + n/1$$



let $n = 3^k$

$$= T(1) + 3^1 + 3^2 + \dots + 3^k$$
$$= 3^0 + 3^1 + 3^2 + \dots + 3^k$$
$$= \frac{3^{k+1}-1}{2} = \frac{3^k \cdot 3^1 - 1}{2}$$

Downloaded from CSIT Tutor

$$= \frac{3n}{2} - \frac{1}{2}$$

$$\leq 3n/2$$

$$\approx O(n) \quad //$$

উদা. 3:- $T(n) = 2T(n/2) + n$; $T(n) = 1$; $n = 1$.

সমাধান:-

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

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

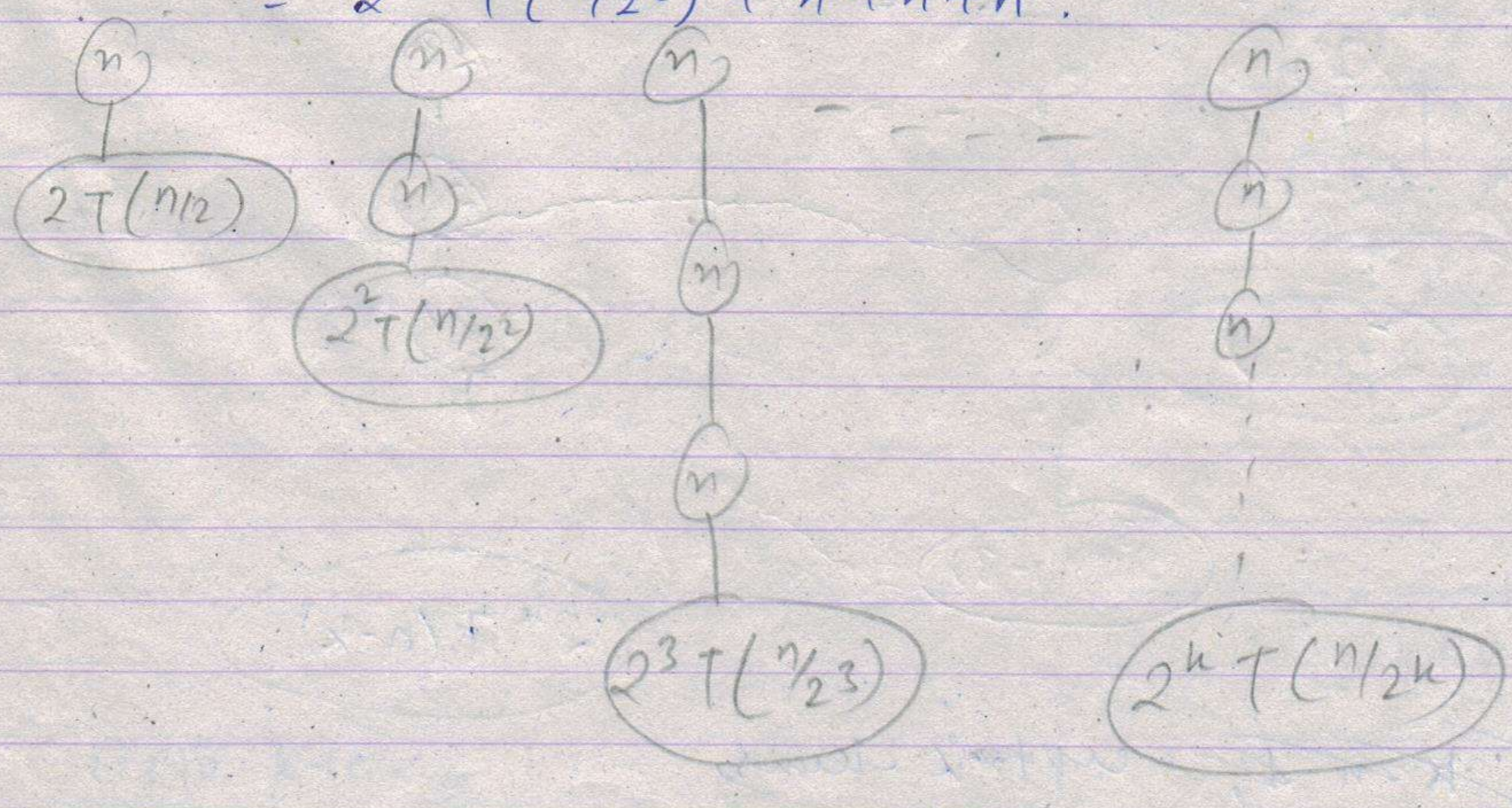
$$= 2^2 T(n/4) + n + n$$

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

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

$$\vdots$$

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



Downloaded from CSIT Tutor

Let $n = 2^k$ $\Rightarrow$ $k = \log n$.

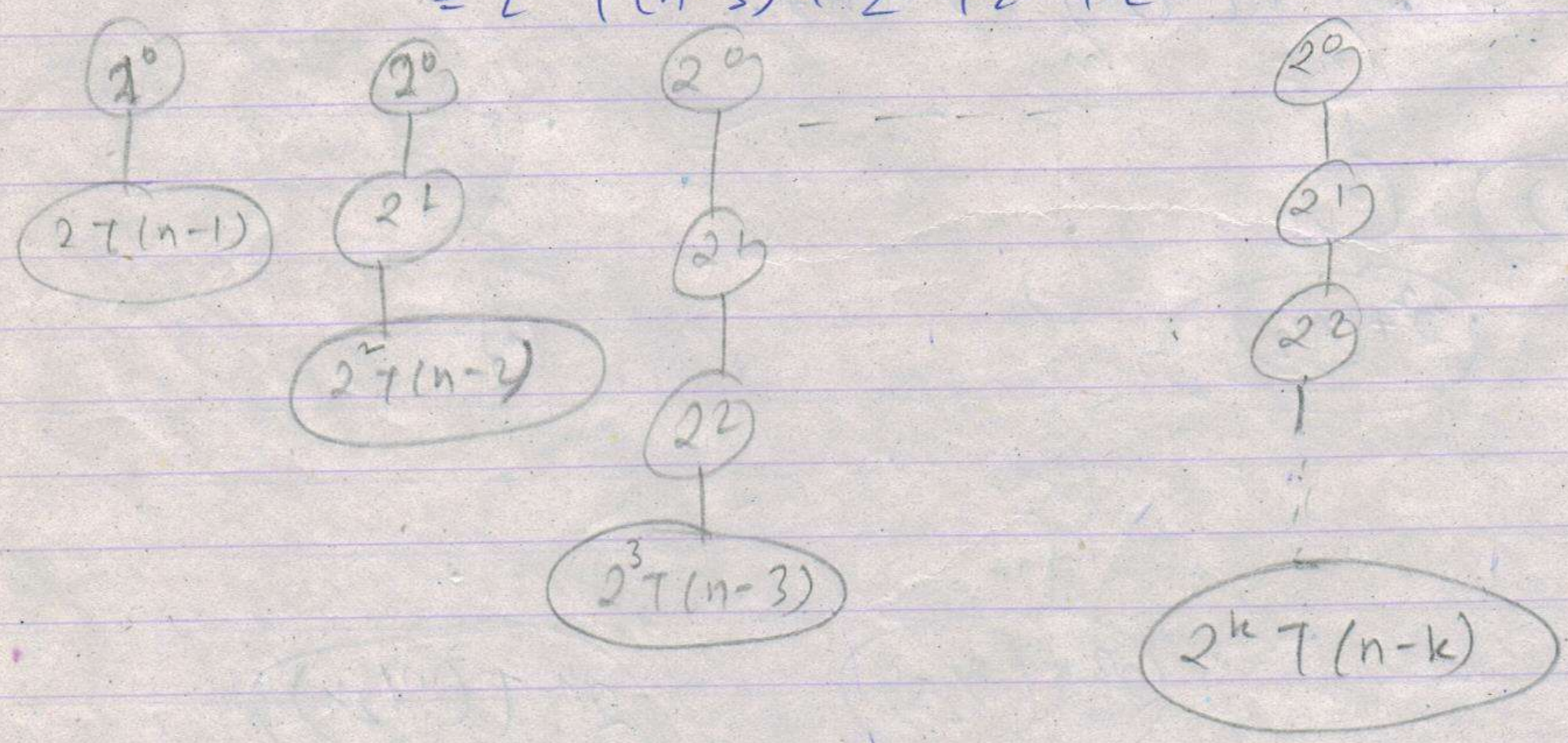
Then

$$T(n) = 2^k T(n/2^k) + n + n + n + \cdots + n$$
$$= n + n + n + n + \cdots + n$$
$$= kn = n \log n.$$
$$= O(n \log n)$$
$$\therefore T(n) = O(n \log n).$$

Eg. 4:- $T(n) = 2T(n-1) + 1$; $T(1) = 1$ for $n = 1$.

Solⁿ:-

$$T(n) = 2T(n-1) + 1$$
$$= 2[2T(n-1-1) + 1] + 1$$
$$= 2^2 T(n-2) + 2 + 1$$
$$= 2^2[2T(n-3) + 1] + 2^1 + 2^0$$
$$= 2^3 T(n-3) + 2^2 + 2^1 + 2^0$$



~~If $k = n-1$,~~ upto k terms,

$$= 2^k T(n-k) + 2^{k-1} + \cdots + 2^0$$

If $k = n-1$,

Downloaded from CSIT Tutor

$= 2^k T(1) + 2^{k-1} + \cdots + 2^0$

$= 2^k + 2^{k-1} + \cdots + 2^0$

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

$= 2^k \cdot 2^1 - 1$

$\leq 2^k$

$\leq 2^n$

$\therefore T(n) = O(2^n)$.

Eg. 5 :- $T(n) = 2T(n/2) + 1$; $T(1) = 1$ for $n = 1$.

Soln :-

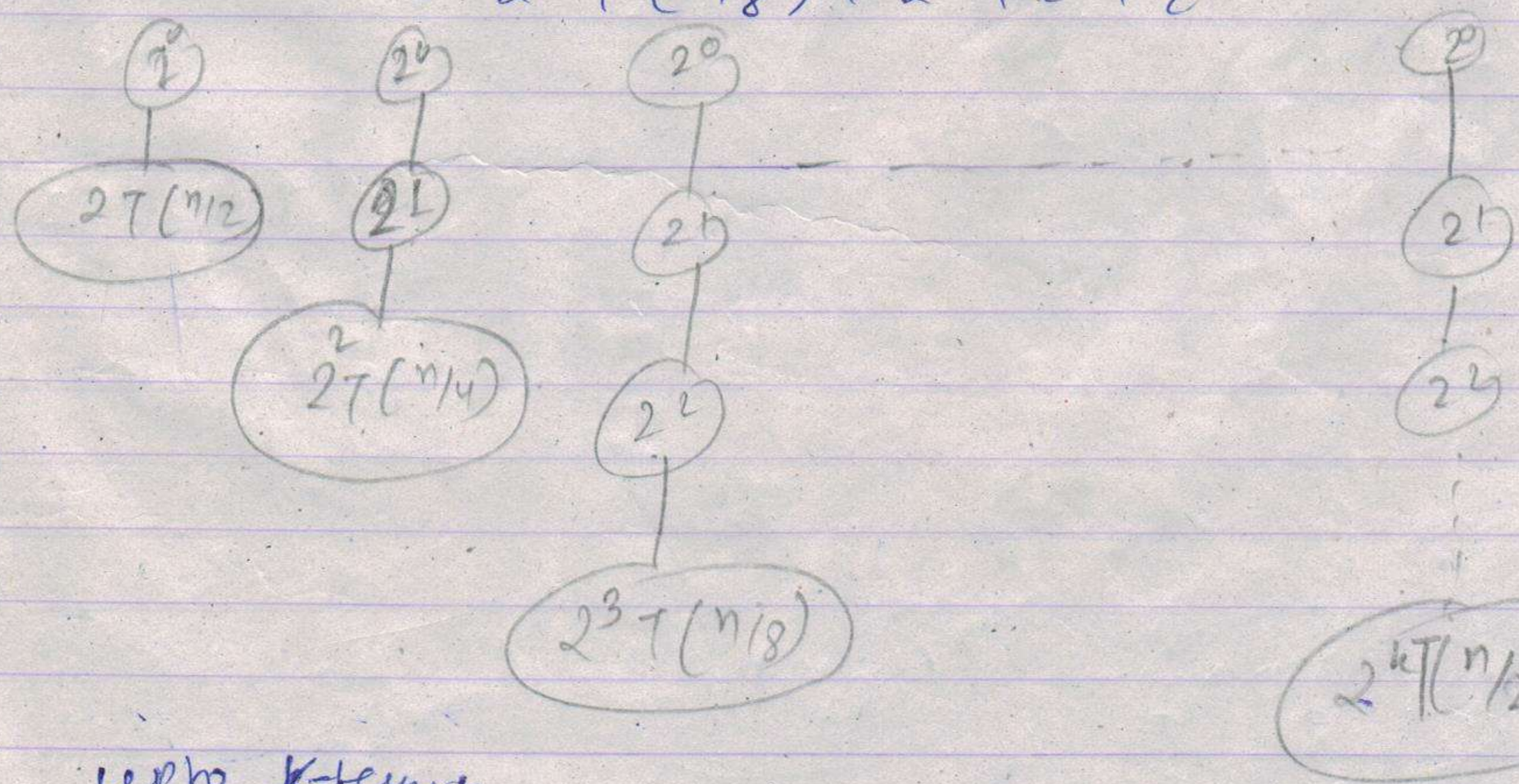
$T(n) = 2T(n/2) + 1$

$= 2[2T(n/4) + 1] + 1$

$= 2^2 T(n/4) + 2^1 + 2^0$

$= 2^2 [2T(n/8) + 1] + 2^1 + 2^0$

$= 2^3 T(n/8) + 2^2 + 2^1 + 2^0$



upto k-terms

$T(n) = 2^k T(n/2^k) + 2^{k-1} + 2^{k-2} + \cdots + 2^0$.

Downloaded from CSIT Tutor

If $n = 2^k$;

$\cancel{= 2^k + n + n + n + \cdots n}$

$= 2^k \cdot T(1) + 2^{k-1} + 2^{k-2} + \cdots + 2^0$

$= 2^k + 2^{k-1} + 2^{k-2} + \cdots + 2^0$

$= \cancel{\frac{2^{k+1} - 1}{2}}$

$= 2^k \cdot 2 - 1 = 2^* n - 1$

$\leq 2^n - 1$

$\approx 2n$

$\therefore T(n) = O(n)$.

Downloaded from CSIT Tutor

Eg :- $T(n) = 4T(n/4) + n \; ; \; n > 1$

$T(n) = 1 \; ; \; n = 1$.

Soln :-

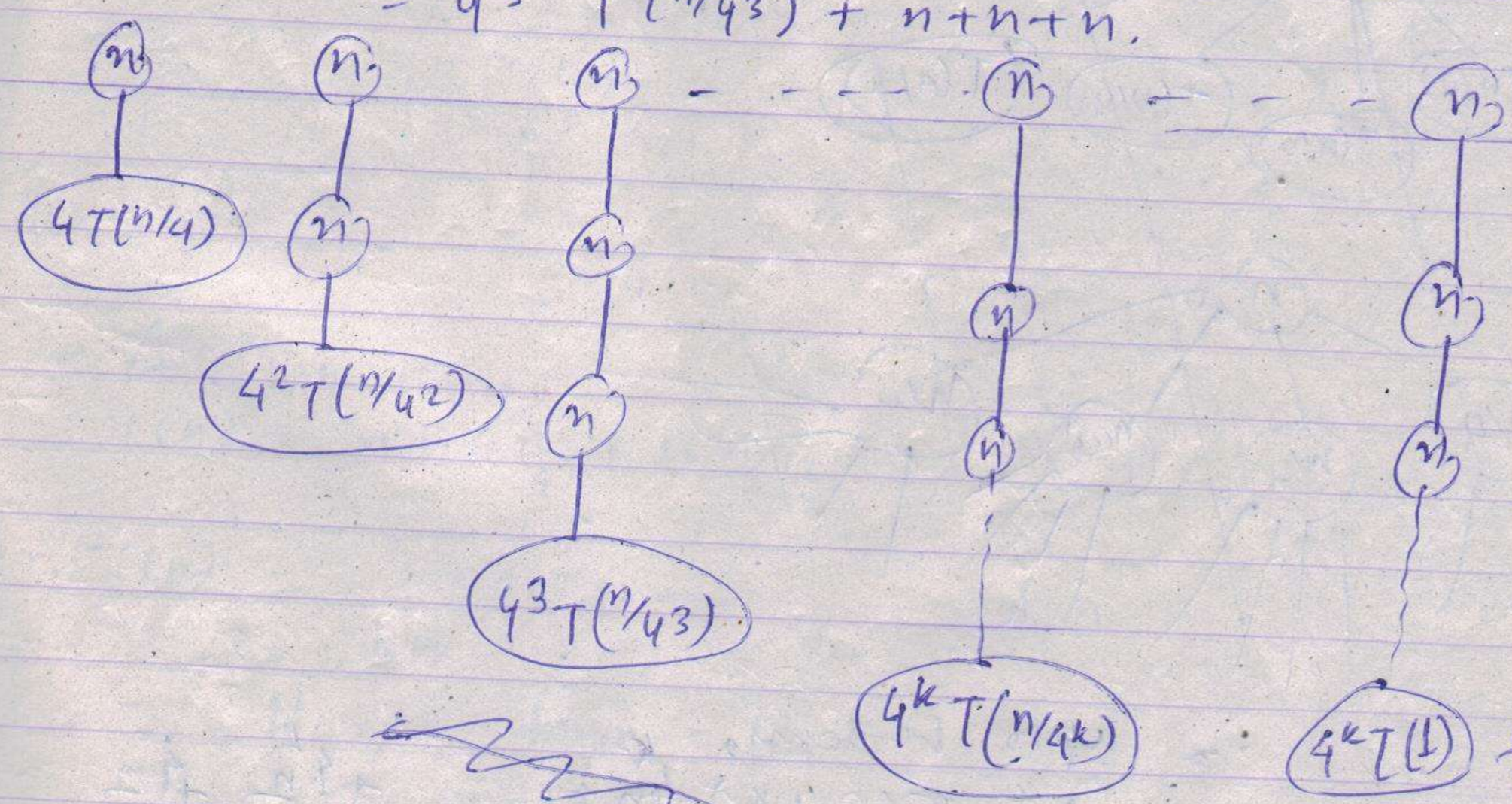
$T(n) = 4\,T(n/4) + n$

$= 4\,[4\,T(n/16) + n/4] + n$

$= 4^2\,T(n/4^2) + n + n$

$= 4^2\,[4\,T(n/4^3) + n/4^2] + n + n$

$= 4^3\,T(n/4^3) + n + n + n$.



upto k-terms, if $4^k = n$ ~~$\Rightarrow k = \log n / \log 4 = \log(n-4)$~~

$= 4^k\,T(n/4^k) + n + n + n + \cdots + n$

$= 4^k T(1) + n + n + n + \cdots + n$

$= 4^k + n + n + \cdots + n$

$= n + n + n + \cdots + n$

$= kn$

~~$= \log(n-4) \cdot n$~~

$= n \log_4 n$

$T(n) = O(n \log_4 n)$. //

$k = \log_4 n$

$4^k = n$

$\log_4 4^k = \log_4 n$

$k \underline{\log_4 4} = \log_4 n$

Downloaded from CSIT Tutor

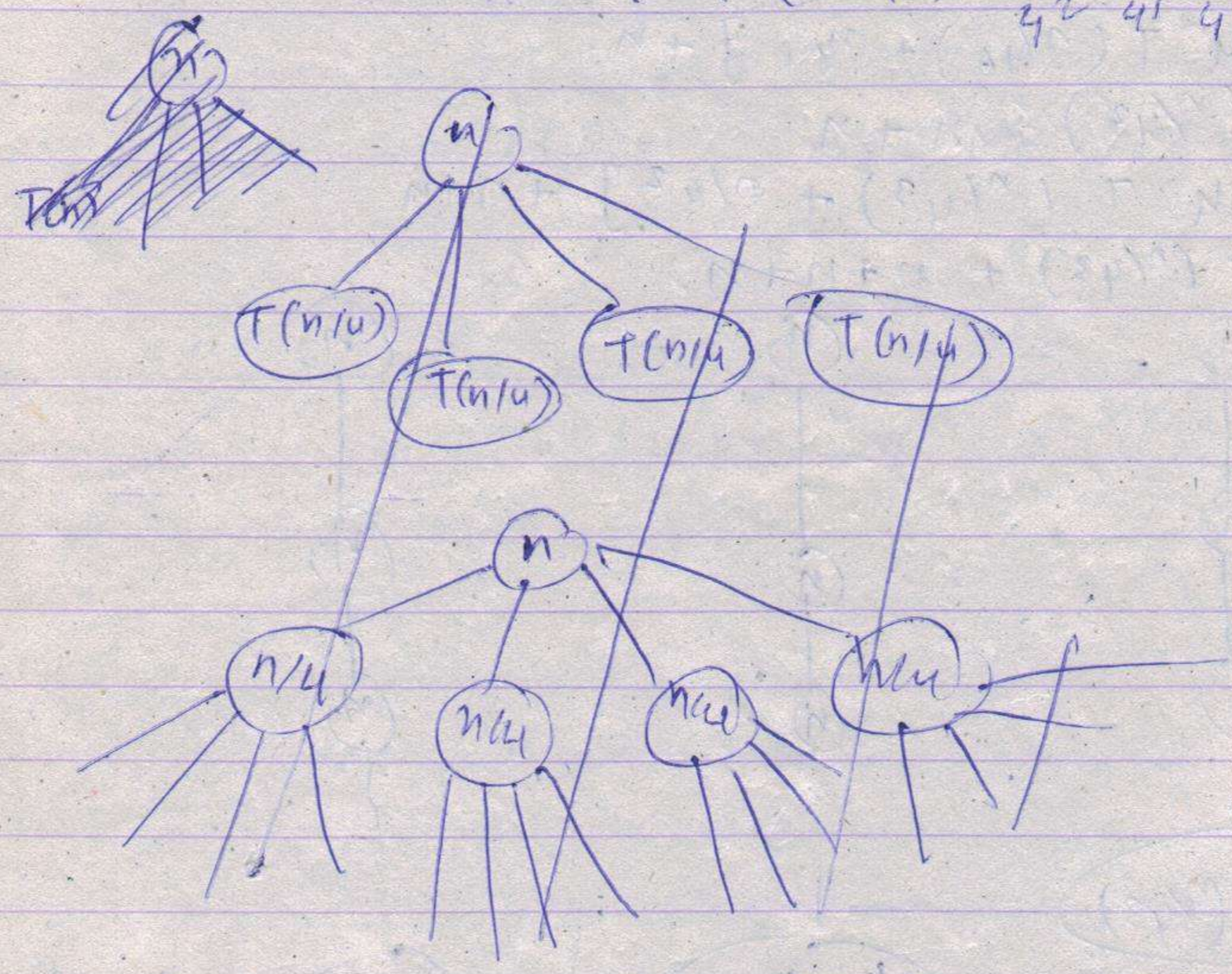
Soln :- $T(n) = 4T(n/4) + n$

$= 4[4T(n/4^2) + n/4] + n$

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

$= 4^2[4T(n/4^3) + n/4^2] + \frac{4n}{4} + n$

$= 4^3 T(n/4^3) + \frac{4^3 n}{4^2} + \frac{4^1 n}{4^1} + \frac{n}{4^0}$



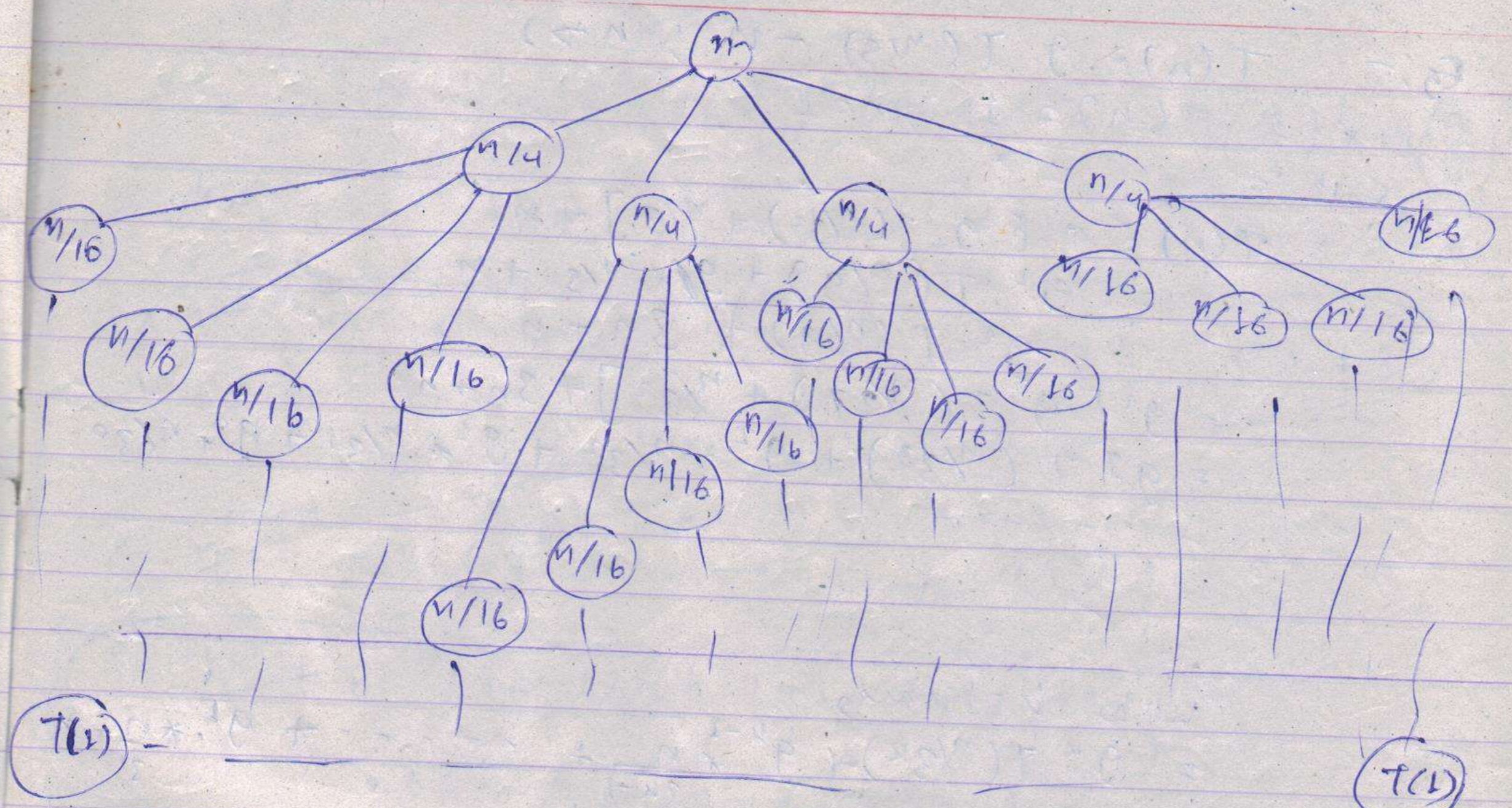
$\Rightarrow$ upto k-terms,

$= 4^k T(n/4^k) + \frac{4^{k-1} n}{4^{k-1}} + \dots + \frac{4^1 n}{4^1} + \frac{4^0 n}{4^0}$

If $4^k = n$,

$= 4^k T(1) + \frac{4^{k-1} n}{4^{k-1}} + \dots + \frac{4^1 n}{4^1} +$

Downloaded from CSIT Tutor



Adding terms,

$T(n) = n + 4\frac{n}{4} + 16\, n/16 + \cdots + 4^k \frac{n}{4^k}$

or, $T(n) = n + n + n + \cdots + n$

$= kn$

Now, $\log n = \log 4^k$

or, $\log_4 n = \log_4 4^k$

or, $\log_4 n = k \cdot \log_4 4$

$\therefore k = \log_4 n$. $(\because \log_4 4 = 1)$

$\therefore T(n) = (\log_4 n)\, n$

$\therefore T(n) = O(n \log_4 n)$ //

Downloaded from CSIT Tutor

$$T(n) = a\, T(n/b) + f(n)$$

a → no. of branch

$\frac{\varepsilon}{4}$ $\frac{\varepsilon}{4}$ $\frac{\varepsilon}{4}$

Eg:- $T(n) = 9\, T(n/3) + n \;;\; n > 1$

$T(n) = 1 \;;\; n = 1$

Soln:-

$$T(n) = 9\,[\,9\, T(n/3^2) + n/3\,] + n$$
$$= 9^2\, T(n/9) + 9 * n/3 + n$$
$$= 9^2\, T(n/9) + 3n + n$$
$$= 9^2\,[\,9\, T(n/27) + n/3^2\,] + 3n + n$$
$$= 9^3\, T(n/3^3) + 9^2 * n/3^2 + 9^1 * n/3^1 + 9^0 * n/3^0$$

- - - -

⋮

upto k-terms

$$= 9^k\, T(n/3^k) + 9^{k-1} * \frac{n}{3^{k-1}} + \cdots + 9^1 * \frac{n}{3^1} + n$$

If $n = 3^k$,

$\log_3 n = \log_3 3^k$

$\Rightarrow k = \log_3 n$

$$T(n) = 9^k \cdot T(1) + 9^{k-1} * \frac{n}{3^{k-1}} + \cdots + 9^0 * \frac{n}{3^0}$$

$$= 3^{2k} + 3^{2(k-1)} * \frac{n}{3^{k-1}} + \cdots + 3^0 * \frac{n}{3^0}$$

$$= 3^{2k} + 3^2 * n + \cdots$$

$$\left(\frac{9}{3}\right)^2 + \left(\frac{9}{3}\right)^1 + \left(\frac{9}{3}\right)^0$$

$$9^k \quad 3^{k-1} \quad \cdots \quad 3^2 + 3^1 + 3^0$$

$$\cancel{9^{\log_3 n}} + 3^{k-1} \quad 3 \quad 2 \quad \cdots \quad 3^k n$$

$$\cancel{n^{\log_3 9}} \quad \cancel{n^{\log_3 3}}^{\,2}$$

$$9^{\log_3 n} = n^{\log_3 9} = n^2$$

Downloaded from CSIT Tutor

$\log_3 3^2 \Rightarrow 2\log_3 3$

$9^{\log_3 n}$

$\Rightarrow n^{\log_3 9} = n^{\log_3 3^2}$

$\Rightarrow n^2$

$= 9^k + \left(\frac{9}{3}\right)^{k-1} * n + \left(\frac{9}{3}\right)^{k-2} * n + \cdots + \left(\frac{9}{3}\right)^1 * n + \left(\frac{9}{3}\right)$

$= 9^{\log_3 n} + 3^{k-1} * n + 3^{k-2} * n + \cdots + 3^1 * n + 3^0 * n$

$= n^{\log_3 9} + 3^{k-1} * 3^k + 3^{k-2} * 3^k + \cdots + 3^k + 3^0$

$= n^2 + 3^0 + 3^1 + \cdots +$

$= n^{\log_3 9} + n\left(3^{k-1} + \cdots + 3^1 + 3^0\right)$

$= n^2 + n\left[\frac{(k-1)(k-1+1)(k-1+2)}{6}\right]$

$= n^2 + n\left(\frac{k \cdot (k-1)(k+1)}{6}\right)$

$=$

$= n^2 + n\left[\frac{3^{k-1+1} - 1}{3-1}\right]$

$= n^2 + n\left[\frac{3^k - 1}{2}\right]$

$= n^2 + n\left[\frac{n-1}{2}\right]$

$= n^2 + \frac{n^2 - n}{2}$

$\Rightarrow \frac{2n^2 + n^2 - n}{2} = \frac{3n^2 - n}{2}$

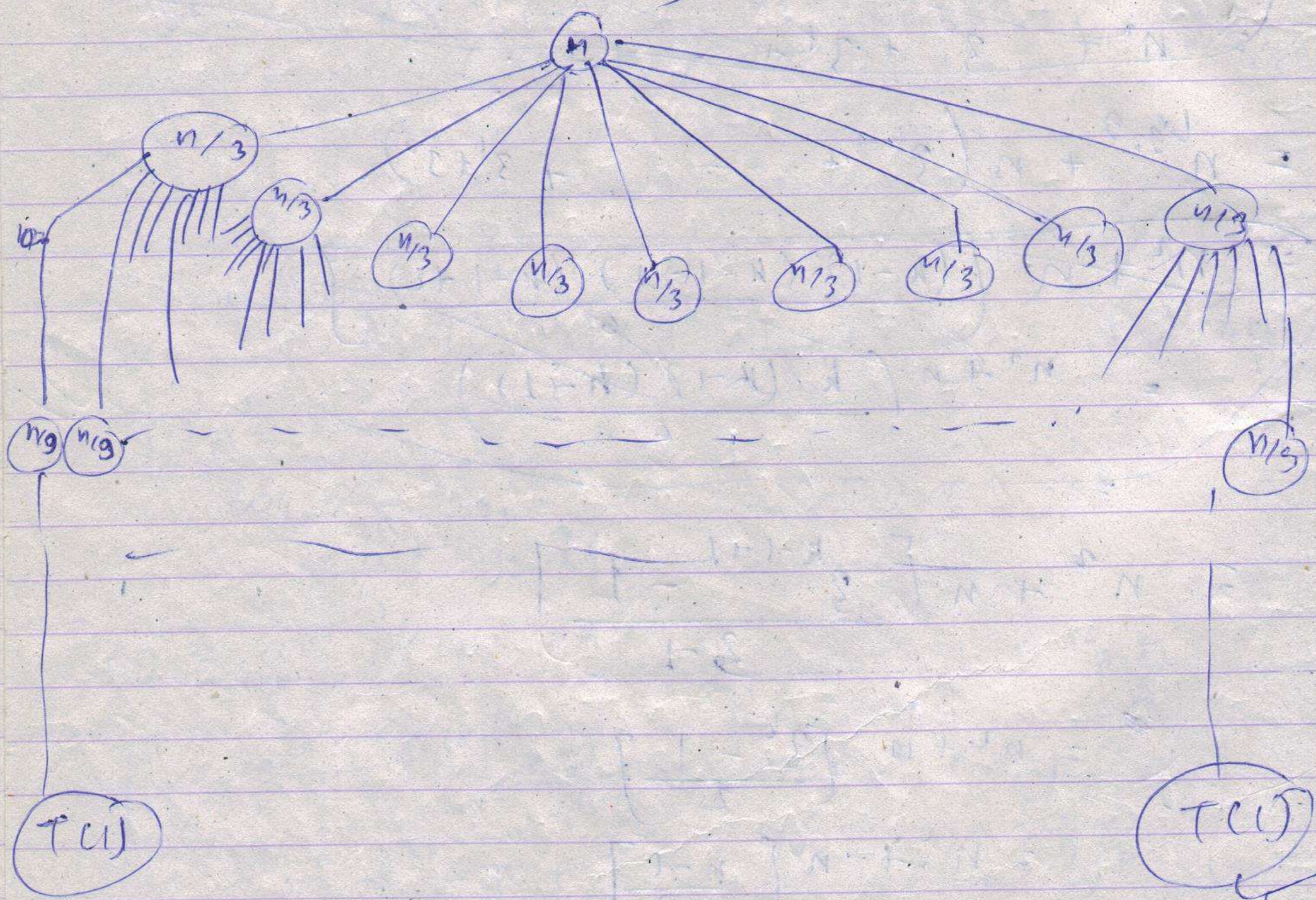
$= \frac{3}{2}n^2 - \frac{n}{2}$

Downloaded from CSIT Tutor

$$\leq \frac{3n^2}{2}$$

$$\approx n^2$$

$$\therefore T(n) = O(n^2)$$



Downloaded from CSIT Tutor

Master's Method.

Master method is used to solve the recurrences of the form,

$$T(n) = a\ T(n/b) + fn.$$

Where $a >= 1$ and $b > 1$ are constants and fn is asymptotically positive function. This method is very handy because most of the recurrence relations we encounter in the analysis of algorithms are of the above kinds and it is very easy to use this method.

The above recurrence divides problems of size n into a sub-problem each of size n/b where a and b are positive constants. The cost of dividing and combining the problem are given by fn.

Master Theorem:-

Let $a >= 1$ and $b > 1$ be constants, let $f(n)$ be a function and let $T(n)$ be defined on the non-negative integers by the recurrence,

$$T(n) = a\ T(n/b) + fn,$$

where, n/b is einer $\lceil n/b \rceil$ or $\lfloor n/b \rfloor$. Then $T(n)$ can be bounded asymptotically as follows:-

1) ~~If $f(n) = O(n^{\log_b a - \epsilon}$~~

→

Downloaded from CSIT Tutor

1) If $f(n) = O(n^{\log_b a - \epsilon})$, for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.

2) If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \cdot \log n)$.

3) If $f(n) = \Omega(n^{\log_b a + \epsilon})$, for some constant $\epsilon > 0$, and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$, and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Eg: (1) :- $T(n) = 9T(n/3) + n$

$\Rightarrow$ Comparing to $T(n) = aT(n/b) + f(n)$,

$a = 9$, $b = 3$, $f(n) = n$.

Now,

$n^{\log_b a} = n^{\log_3 9} = n^2$ $[\neq f(n)]$

$f(n) = n = O(n^{\log_3 9 - \epsilon})$ where $\epsilon = 1$.

So, we choose Case 1 :-

So, $T(n) = \Theta(n^{\log_b a})$

$\therefore T(n) = \Theta(n^2)$.

Eg: (2) :- $T(n) = 2T(n/2) + n$.

$\Rightarrow$ Comparing with $T(n) = aT(n/b) + f(n)$.

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

So, $n^{\log_b a} = n^{\log_2 2} = n^1 = n$. $[= f(n)]$.

So, $f(n) = n = O(n^{\log_b a - \epsilon})$ where $\epsilon = 1$.

So, we choose case 2 :-

So, $T(n) = \Theta(n^{\log_b a} \cdot \log n)$

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

Downloaded from CSIT Tutor

Eg:- (3) :- $T(n) = 3T(n/4) + n\log n$.

Here, $a = 3$, $b = 4$, $f(n) = n\log n$.

So, $n^{\log_b a} = n^{\log_4 3} = n^{0.793}$ $[\neq f(n)]$

~~$f(n) = n\log n$~~

Now, 1st Case invalid as $f(n) = n\log n$

i.e. $n\log n > n^{0.793}$. So, we cannot deduct ϵ. So, case 3 is chosen.

Now,

$n^{\log_b a} = n^{0.793 + 0.207}$ [to make n^1]

$= n$. i.e. $\epsilon = 0.207$.

Now,

$a f(n/b) \leq c * f(n)$

~~$\Rightarrow 3 * f(n/4$~~

Here, $f(n) = n\log n$

$f(n/b) = f(n/4) = (n/4 \log n/4)$.

$a f(n/b) = a\left(\frac{n}{4} \cdot \log\frac{n}{4}\right) = 3\frac{n}{4} \cdot \log\frac{n}{4}$

$= \frac{3}{4} \cdot n\log\frac{n}{4} \leq \frac{3}{4} n\log n$,

$\leq c \cdot n\log n \leq c f(n)$

(where $c = 3/4$)

$\therefore a f(n/b) \leq c * f(n)$ is true.

$\therefore T(n) = \Theta(f(n))$

$= \Theta(n\log n)$. //.

Downloaded from CSIT Tutor

Q. Solve by Master's Method.

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

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

c) $T(n) = 4T(n/2) + n^3$.

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

Soln :-

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

Now, $n^{\log_b a} = n^{\log_2 4} = n^2 \quad [\neq f(n)]$

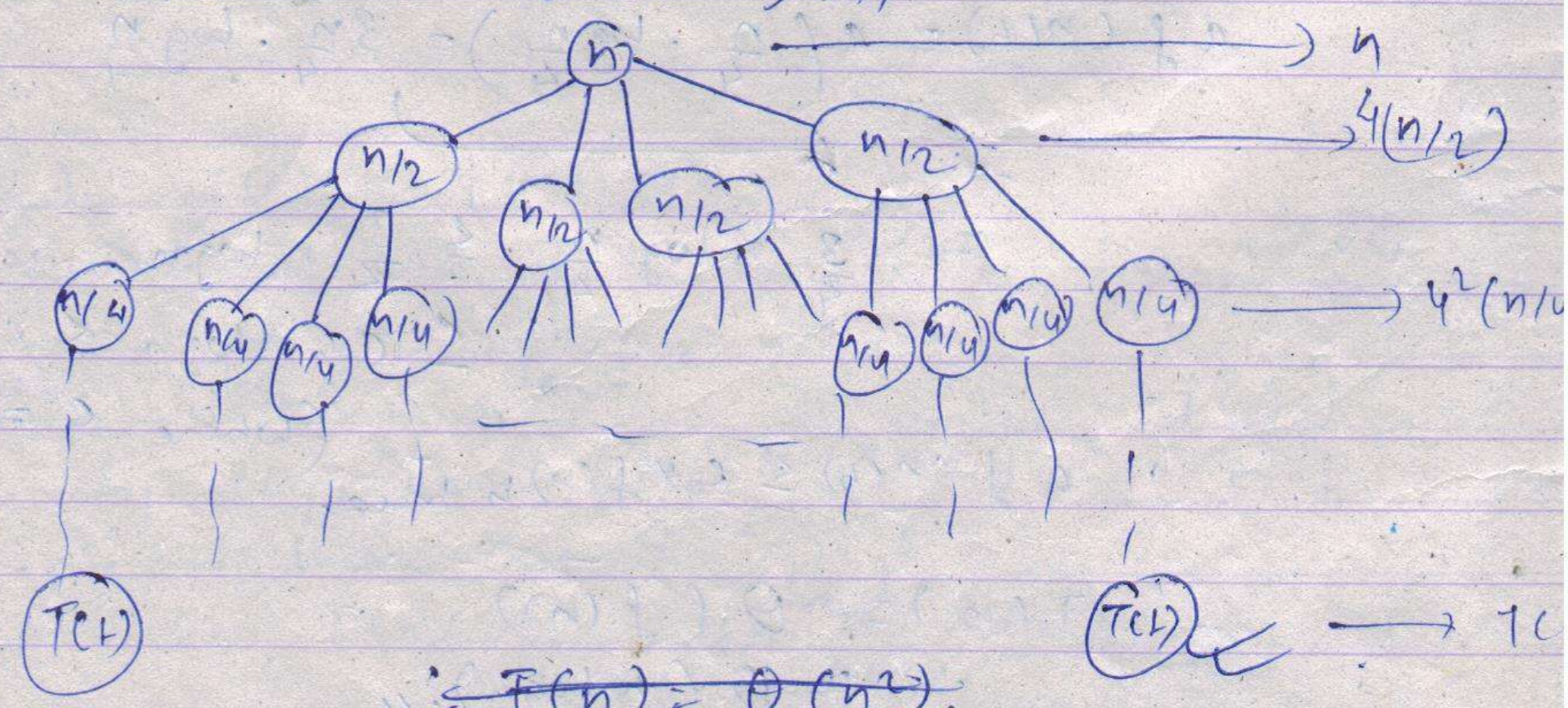
So,

$f(n) = n = O(n^{\log_a b - \epsilon})$ where, $\epsilon > 0$

Case 1 is chosen. So,

$T(n) = \Theta(n^{\log_b a})$

$= \Theta(n^2)$.



~~$\therefore T(n) = \Theta(n^2)$.~~

$\Rightarrow n + 4(n/2) + 16(n/4) + \cdots + (4/2)^k * n$

$\Rightarrow n(2^0 + 2^1 + \cdots 2^k)$

$\Rightarrow n * (2n - 1) = 2n^2 - n \leq 2n^2 \approx O(n^2)$

$T(n) \approx O(n^2)$

Downloaded from CSIT Tutor

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

Soln :-

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

Now,

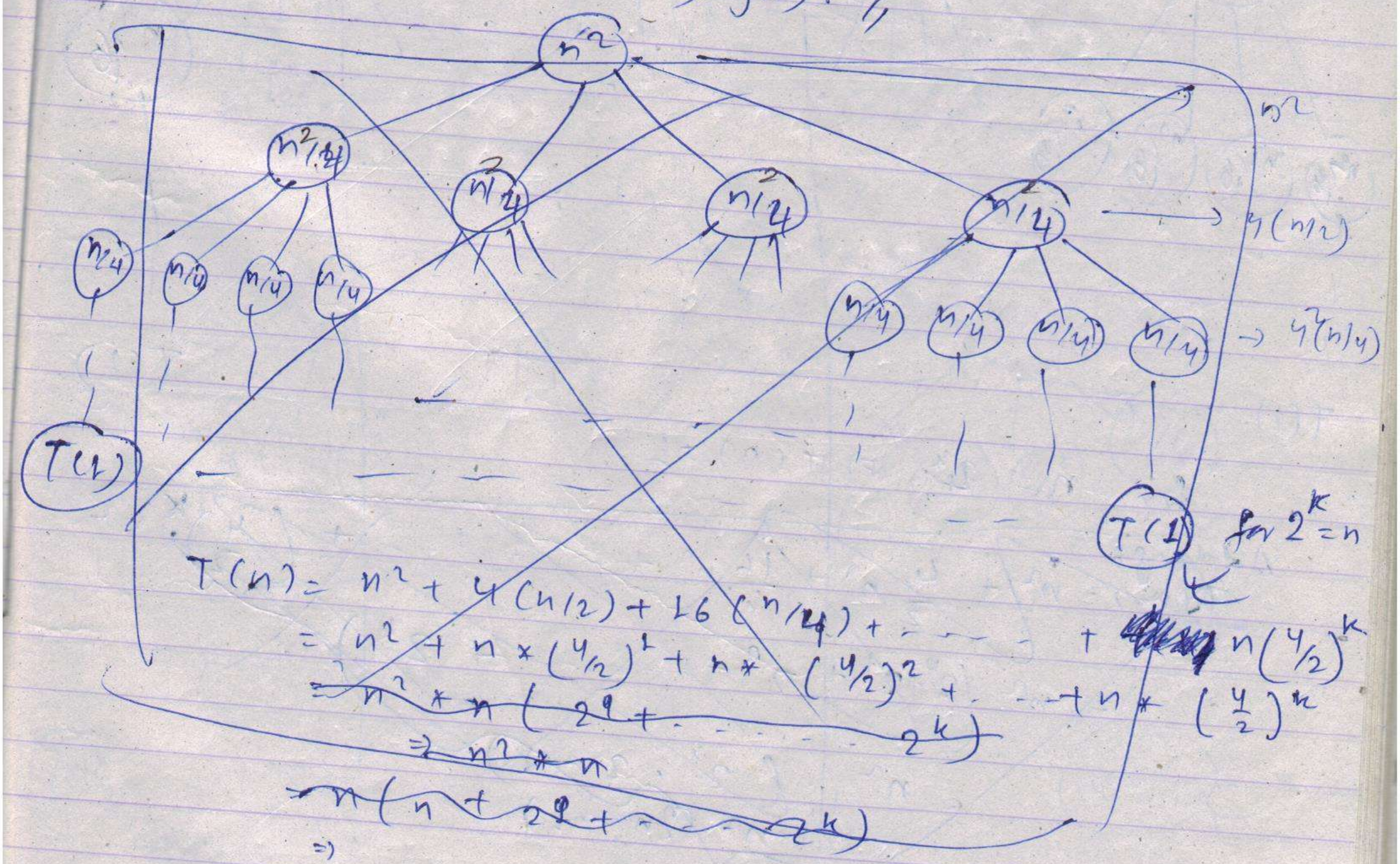
$n^{\log_b a} = n^{\log_2 4} = n^2 \quad [= f(n)]$

So, Case 2 is chosen.

i.e. $f(n) = \Theta(n^{\log_b a})$

$\therefore T(n) = \Theta(n^{\log_b a} \log n)$

$= \Theta(n^2 \log n)$. //



$T(n) = n^2 + 4(n/2) + 16(n/4) + \dots + n(4/2)^k$

$= n^2 + n \times (4/2)^1 + n \times (4/2)^2 + \dots + n \times (\frac{4}{2})^k$

$\cancel{= n^2 * n(2^0 + \dots 2^k)}$

$\cancel{\Rightarrow n^2 * n}$

$\cancel{\Rightarrow n(n + 2^0 + \dots 2^k)}$

$\Rightarrow$

Downloaded from CSIT Tutor

$$T(n) = n^2 + 4\frac{n^2}{4} + 16\frac{n^2}{16} + \cdots + 4^k \times \frac{n^2}{4^k}$$

$$= n^2 + n^2 + n^2 + n^2 + \cdots + n^2$$

$$= kn^2$$

for ~~$k = 4^k$~~ $n^2 = 4^k$, $T(n) = 1$.

~~do, $n^2 = 4^k$~~

~~or, $n = \sqrt{2^{2k}}$~~

~~or, $n = 2^{\sqrt{k}}$~~

~~$\log n = \log_2 2^{\sqrt{k}}$~~

Downloaded from CSIT Tutor

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

Soln :-

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

So, $n^{\log_b a} = n^{\log_2 4} = n^2 \quad [\neq f(n)]$

$\because n^{\log_b a} > f(n)$, Case I and II not valid.

So, Case III is chosen.

~~So~~ i.e.

$f(n) = n^3$

$n^{\log_b a} = n^2$

So, n^{2+L} i.e $n^{2+\epsilon}$

~~where, $\epsilon > 1$.~~

$= n^3$.

So, now,

$f(n) = n^3$

~~$f(n/b) = \frac{n^3}{2}$~~

$f(n/b) = f(n/4)$

$= \left(\frac{n}{4}\right)^3$

$a * f(n/b) = 2 * \frac{n^3}{64} = \frac{n^3}{32}$.

$c * f(n) = c * n^3$

$= \frac{1}{32} * n^3$

$\therefore a f(n/b) \leq c * f(n)$. where $c = 1/32$

$\therefore T(n) = \Theta(f(n))$

$= \Theta(n^3)$.

Downloaded from CSIT Tutor

Eg. :-

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

Soln :-

$a = 2$, $b = 2$, $f(n) = n\log n$

$n^{\log_b a} = n^{\log_2 2} = n$. [$\neq f(n)$]

So, $f(n) < n^{\log_b a}$ i.e. $n\log n < n$.

So, we choose case I. i.e.

~~$f(n) \geq n$~~

So, $n^{1+\epsilon}$ is not correct, $n^{1-\epsilon}$ also not correct. So, we have no case.

Now, $n^{\log_b a} = n$ is assymptotically lesser than $f(n) = n\log n$.

i.e. $n < n\log n$

We cannot solve this relation in Master's Method.

Downloaded from CSIT Tutor

$T(n/4) = n/4 \times \frac{1}{4} \times \frac{1}{4}$

Q.1. $T(n) = 16T(n/4) + n^2$.

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

Q.3. $T(n) = T(\sqrt{n}) + 1$.

Q.4. $T(n) = 3T(n/2) + n\log n$.

Q.5. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

Q.6. Can the Master method be applied to the recurrence
$T(n) = 4T(n/2) + n^2 \log n$. ?
Why or why not, give an asymptotic upper bound for this relation?

Q.1. $T(n) = 16T(n/4) + n^2$.

I. Iteration Method:-

$T(n) = 16\ T(n/4) + n^2$

$= 16\,[16\,T(n/16) + (n/4)^2] + n^2$

$= 16^2\ T(n/16) + 16\ n/4^2 + n^2$

$= 16^2\,[16\ T(n/64) + (n/16)^2] + 4^2 * n^2/4^2 + 4^0 * n^2/4^0$

$= 16^3\ T(n/4^3) + 16^2 * n^2/16^2 + 16$

$= 16\,[16\,T(n/16) + (n/4)^2] + n^2$

$= 16^2\ T(n/16) + 16\ n^2/16 + n^2$

$= 16^2\,[16\ T(n/64) + (n/16)^2] + 16\ n^2/16 + n^2$

$= 16^3\ T(n/64) + 16^2 * n^2/16^2 + 16^1 * n^2/16^1 + 16^0 * n^2/16^0$

upto k-terms,

$= 16^k\ T(n/4^k) + \left(\frac{16}{16}\right)^{k-1} * n^2 + \left(\frac{16}{16}\right)^{k-2} * n^2 \ldots + n^2$

for $4^k = n$, $\Rightarrow \log_4 n = \log_4 4^k \Rightarrow k = \log_4 n$.

Downloaded from CSIT Tutor

$= 16^k T(1) + n^2 + n^2 + \cdots n^2$

$= 16^k + kn^2 + n^2 + \cdots + n^2$

$= 4^{2k} + kn^2 + n^2 + \cdots + n^2 \quad (\because 4^k = n,\ 4^{2k} = n^2)$

$= n^2 + kn^2 + n^2 + \cdots + n^2$

$= n^2(1+k) \quad = kn^2$

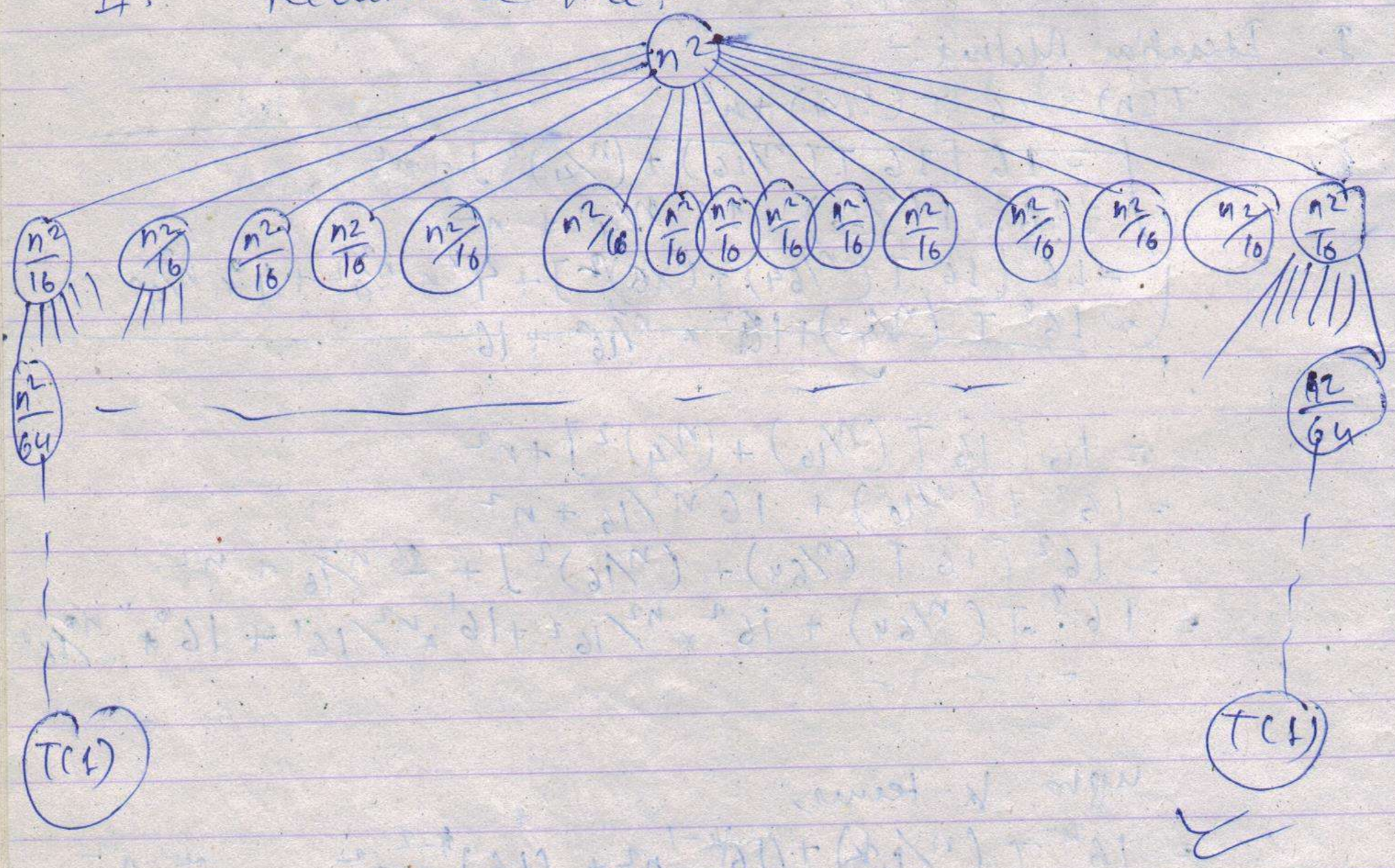
$= n^2 + kn^2$

$T(n) = n^2 \log_4 n$

$= \cancel{n^2 \log_4 n}$

$\therefore T(n) = O(n^2 \log_4 n)$ //

II. Recurrence Tree :-



Downloaded from CSIT Tutor

Adding these,

$$T(n) = n^2 + 16^1 \frac{n^2}{16^1} + 16^2 \frac{n^2}{16^2} + \cdots + 16^k \left(\frac{n^2}{16^k}\right)$$

for $4^k = n$, $T(n) = 1$.

$\Rightarrow k = \log_4 n$

$$\therefore T(n) = n^2 + n^2 + n^2 + \cdots + n^2$$

$$\cancel{= n^2 + n^2 + n^2 + \cdots + 4^{2k}}$$

$$\cancel{= n^2 + n^2 + n^2 + \cdots + n^2}$$

$$= kn^2$$

$$= n^2 \log n.$$

III. Substitution Method :-

Suppose, $T(n) = O(n^2 \log_4 n)$.

So, i.e. $T(n) \le c * n^2 \log_4 n$.

Now, $T(n) = 16T(n/4) + n^2$

$$\le 16 \left[c * (n/4)^2 . \log_4 (n/4) \right] + n^2$$

$$\le 16 * c * \frac{n^2}{16} \log_4 \frac{n}{4} + n^2$$

$$\le c * n^2 \log \frac{n}{4} + n^2$$

$$\le cn^2 (\log_4 n - \log_4 4) + n^2$$

$$\le cn^2 (\log_4 n - 1) + n^2$$

$$\le cn^2 \log_4 n - cn^2 + n^2$$

$$\le cn^2 \log_4 n - (cn^2 - n^2)$$

$$\le cn^2 \log_4 n - n^2 (c-1).$$

$$\therefore T(n) \le cn^2 \log_4 n = O(n^2 \log n).$$

Downloaded from CSIT Tutor

Checking boundary conditions,

$T(1) = 1$

$T(1) = cn^2 \log n$

$\leq c * 1 * \log_4 1$

$\leq c * \log_4 4^0$

$\leq c * 0$

≤ 0 (false)

$T(2) = 16T(n/4) + n^2$

$= 16\, T(1/2) + n^2$

$= 16\, T(0) + n^2$

$= 16 + 4$

$= 20$

$T(2) = c\, n^2 \log n$

$= c * 4 * \log_4 2$

$= c * 4 * \log_4 4^{1/2}$

$= c * 4 * 1/2$

$= c * 2$ (true for $c = 10$)

$T(3) = 16\, T(3/4) + n^2$

$= 16\, T(0) + n^2$

$= 16 * 1 + 9$

$\Rightarrow 25$

$T(3) = c\, n^2 \log n$

$= c * 9 * \log_4 3$

$= 7.137 * c$ (true for $c = 10$)

Downloaded from CSIT Tutor

So, for $n \geq 2$ and $c = 10$, we have,

$$T(n) = O(n^2 \log n).$$

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

Soln :-

I. Iterative Method :-

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

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

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

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

$= T(n-4) + (n-3) + n - 2 + n + n - 1$

$\cancel{= T(n-4) + 4n - 6}$

$= T(n-4) + (n-3) + (n-2) + (n-1) + (n-0).$

$\vdots$

upto k-terms,

$T(n) = T(n-k) + (n-(k-1)) + (n-(k-2)) + \cdots + (n-1) + (n-0)$

for $k = n-1$, $T(n) = 1$.

$T(n) = T(1) + (n - n + 2) + (n - n + 3) + \cdots + (n-1) + n.$

$= 1 + 2 + 3 + \cdots (n-1) + n$

$= \sum_{i=1}^{n} i$

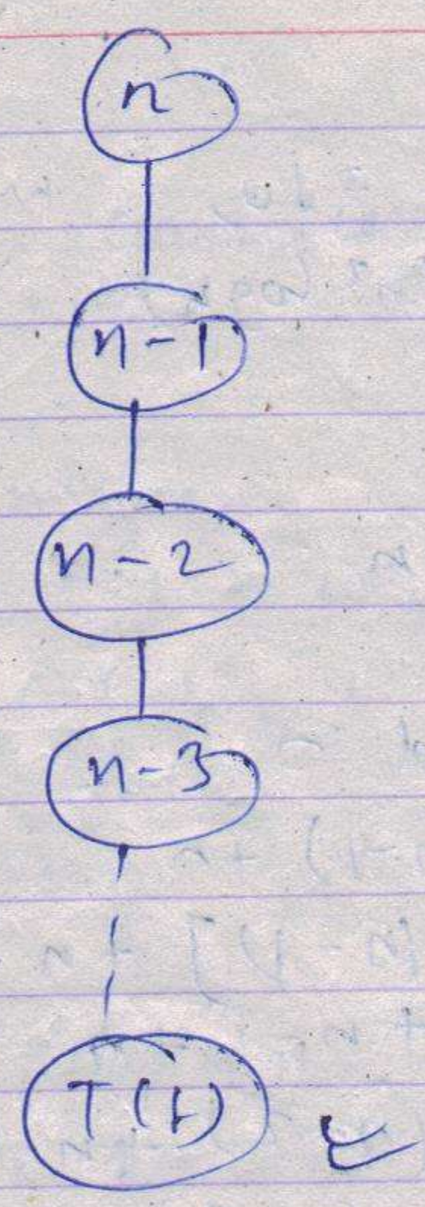
$= \frac{n(n+1)}{2} = \frac{n^2 + n}{2} = \left(\frac{n^2}{2}\right) + \left(\frac{n}{2}\right)$

$\leq \frac{n^2}{2} \approx n^2$

$\therefore T(n) = O(n^2).$

Downloaded from CSIT Tutor

II. Recursion :-



Adding these,

$T(n) = n + (n-1) + (n-2) + (n-3) + \cdots + (n-k)$

for $k = (n-1)$,

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

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

$= \frac{n^2}{2} + \frac{n}{2} \approx \frac{n^2}{2} \approx n^2$.

$\therefore T(n) = O(n^2)$.

Downloaded from CSIT Tutor

$\boxed{11}$. Substitution :-

let $T(n) = O(n^2)$.

So, $T(n) \leq c * n^2$.

Then,

$$
\begin{aligned}
T(n) &= T(n-1) + n \\
&= c * (n-1)^2 + n \\
&= c * (n^2 - 2n + 1) + n \\
&= cn^2 - 2cn + c + n \\
&= cn^2 - (2cn - c - n). \\
&= cn^2 - d. \\
&\leq cn^2
\end{aligned}
$$

$\therefore T(n) = O(n^2)$.

Checking boundary condition,

$T(1) = 1$, $\quad T(1) = c * n^2 = c * 1$ (for $c = 1$).

$T(2) = T(2-1) + n$
$= T(1) + n = 3$, $\quad T(2) = c * 2^2 = 4 * c$ (for $c = 1$)

$T(3) = T(3-1) + n$
$= 3 + 3$
$= 6$, $\quad T(3) = c * 3^2 = 9 * c$ (for $c = 1$).

$\therefore$ for $n \geq 1$ and $c = 1$, this is true for $T(n) = O(n^2)$.

Downloaded from CSIT Tutor

3. $\frac{n^{1/2}}{} = \frac{2^{m/2}}{}$

$n = 2^m$

$T(n) = T(\sqrt{n}) + 1.$

Soln :-

let $n = 2^m$ $\Rightarrow m = \log n.$

So, $n^{1/2} = 2^{m/2}$

$T(n) = T(n^{1/2}) + 1$

or, $T(2^m) = T(2^{m/2}) + 1$

Let, $T(2^m) = S(m).$

$S(m) = S(m/2) + 1.$ —— (1)

Comparing with $T(n) = T(n/2) + 1,$

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

So,

$S(m) = O(\log m)$

$\Rightarrow T(2^m) = O(\log m)$

$\Rightarrow T(n) = O(\log(\log n)).$

Downloaded from CSIT Tutor

Sorting :-

Applications of Sorting :-

1) Searching
2) Closest Pair.
3) Element Uniqueness.
4) Frequency distribution.
5) Convex hull.

Selection Sort :-

Algorithm :-

```
Selection_sort (A)
{
    for (i=0; i<n; i++)
        for (j=i+1; j<n; j++)
            if (A[j] < A[i])
                swap (A[i], A[j]);
}
```

Example :- Tracing :-

i =	0	1	2	3	4	5
	23	10	12	5	14	15

n = 6

Pass 1 :- for (i=0; i<6; i++) ⇒ i=0 = 23

for (j=i+1=1; j<n; j++) ⇒ j=1 = 10

if (A[j] < A[i]) ⇒ (10<23) true

swap (A[0], A[1]).

(A) ⇒	10	23	12	5	14	15
	0	1	2	3	4	5

Downloaded from CSIT Tutor

~~Pass 2 :- for(i=1 ; i<6 ; i++)~~

~~for(j=2 ; j<6 ; j++)~~

Pass 2 :- for i=0. ~~if (A[i]<A[j]) ⇒ (A[1]<A[2]) ⇒~~

Then, for (j=2 ; j<n ; j++)

if (A[j] < A[i]) ⇒ (12 < 10) ? false

swap(A[i], A[j]) NO.

⇒ [10 23 12 5 14 15]

Pass 3 :- for i=0

for (j=3 ; j<n ; j++)

if (A[j] < A[i]) ⇒ (5 < 10) ? true

swap(A[i], A[j]) Yes

⇒ [5 23 12 10 14 15]

Pass 4 :- for i=0

for (j=4 ; j<n ; j++)

if (A[j] < A[i]) ⇒ (14 < 5) ? false

swap(A[i], A[j]) NO.

⇒ [5 23 12 10 14 15]

Pass 5 :- for (i=0)

for (j=5 ; j<n ; j++)

if (A[5] < A[0]) ⇒ (15 < 5) ? false

swap(A[i], A[j]) NO.

⇒ [5 23 12 10 14 15].

Downloaded from CSIT Tutor

Pass 6 :- for i=0
for (j = 6; j < n; j++) NO

Pass 7 :- ~~for i=0~~ for (i = 1; i<n; i++)
for (j = 2; j<n; j++)
if (A[j] < A[i]) ⇒ (12 < 23)? tr
swap(A[i], A[j])

~~Pass 8 :- for i = 1,~~
~~for (j = 3; j < n; j++)~~
~~if (A[j] < A[i])~~

⇒ [5 12 23 10 14 15]
0 1 2 3 4 5

Pass 8 :- for i = 1
for (j = 3; j<n; j++)
if (A[3] < A[1]) Yes
swap [12, 10]

⇒ [5 10 23 12 14 15]
0 1 2 3 4 5

Pass 9 :- for i = 1, j = 4
if (A[4] < A[1]) No.

Pass 10 :- for i = 1, j = 5
if [A[5] < A[1]) NO.

Pass 11 :- for i = 1; j = 6 NO execution.

Pass 12 :- i = 2, j = 3
if [A[3] < A[2]) yes
swap(23, 12)

⇒ [5 10 12 23 14 15].
0 1 2 3 4 5

Downloaded from CSIT Tutor

13)
i=2, j=4
if (A[4] < A[2]) No.

14) i=2, j=5
(A[5] < A[2])? No.

15) i=2, j=6 No execution

16) i=3, j=4
if (A[4] < A[3]) ? Yes
⇒ [5 10 12 14 23 15]
0 1 2 3 4 5

17) i=3, j=5 No.
18) i=3, j=6 No
19) i=4, j=5 Yes
if (A[5] < A[4]) Yes
⇒ [5 10 12 14 15 23]
20) i=4, j=6 no execution
21) i=5, j=6 No.

⇒ [5 10 12 14 15 23]

Downloaded from CSIT Tutor

	Cost	Time
1) for (i=0; i<n; i++)	c_1	$n+1$
2) for (j=i+1; j<n; j++)	c_2	$\sum_{j=1}^{n} j$
3) if (A[j] < A[i])	c_3	$\sum_{j=1}^{n-1} j$
4) swap(A[i], A[j])	c_4	$\sum_{j=1}^{n-2} j$

i) Worst Case:-

$$T(n) = c_1 * (n+1) + c_2 * \left(\sum_{j=1}^{n} j\right) + c_3 * \left(\sum_{j=1}^{n-1} j\right) + c_4 * \left(\sum_{j=1}^{n-2} j\right)$$

$$\Rightarrow c_1 * (n+1) + c_2 * \frac{n(n+1)}{2} + c_3 * \frac{(n-1)(n)}{2} + c_4 * \frac{(n-2)(n-1)}{2}$$

$$\Rightarrow O(n) + O(n^2) + O(n^2) + O(n^2)$$

$$= O(n^2)$$

② Best Case:-

$$T(n) = c_1 * (n-1) + c_2 * \frac{n(n+1)}{2} + c_3 * \frac{n(n-1)}{2}$$

$$= \Omega(n) + \Omega(n^2) + \Omega(n^2)$$

$$= \Omega(n^2).$$

③ Average Case:- $\Theta(n^2/2)$

$$\approx \Theta(n^2)$$

Downloaded from CSIT Tutor

Bubble Sort:-

Algorithm:-

```
Bubblesort (A, n)
{
    for (i=n-1; i>0; i--)
        for (j=0; j<i; j++)
            if (A[j] > A[j+1])
            {
                temp = A[j];
                A[j] = A[j+1];
                A[j+1] = temp;
            }
}
```

Example:-

0	1	2	3	4	5
23	10	12	5	14	15

$n = 6$.

~~1) i = n-1 i>0 i--~~

Downloaded from CSIT Tutor

	$i=n-1$	$i>0$	$i--$	$j=0$	$j<i$	$j++$	$if(A[j]>A[j+1])$	Swap
1)	6	✓		0	✓		Yes	Yes[

=> [10 23 12 5 14 15]

2)				1	✓		A[1] > A[2] Yes	Yes [2

=> [10 12 23 5 14 15]
0 1 2 3 4 5

3)				2	✓		A[2] > A[3] Yes	Yes [23]

=> [10 12 5 23 14 15]
0 1 2 3 4 5

4)				3	✓		A[3] > A[4] Yes	Yes [2

=> [10 12 5 14 23 15]

5)				4	✓		A[4] > A[5] Yes	Yes [2

=> [10 12 5 14 15 23]

6)				5	✓		A[5] > A[6] false	No.

=> [10 12 5 14 15 23]

7)	5	✓		0	✓		A[0] > A[1] No	No

8)				1	✓		A[1] > A[2] Yes	Yes [1

=> [10 5 12 14 15 23]
0 1 2 3 4 5

9)				2	✓		A[2] > A[3] No	No
10)				3	✓		A[3] > A[4] No	No
11)				4	✓		A[4] > A[5] No	No
12)				5	✓		A[5] > A6] No	No.

Downloaded from CSIT Tutor

			$A[j] > A[j+1]$	swap.
13)	$i = 4$	$j = 0$	$A[0] > A[1]$ Yes	Yes [10]
	⇒ [5	10	12 14	15 23]
14)		$j = 1$	NO	no
15)		2	NO	NO
16)		3	NO	NO
17)		4	NO	NO
18)		5	NO	NO
19)	$i = 3$	0	"	no
20)		1	"	1
21		2	"	1
22)		3	"	
23)		4		"
24)		5		
25)	$i = 2$	0		1
26)		1		1
27)		2		1
28)		3		1
29)		4		1
30)		5		1
31)	$i = 1$	0		1
32)		1		1
33)		2		1
34)		3		1
35)		4		
36)		5		

37) $i = 0$ (false) ⇒ [5 10 12 14 15 23]

Downloaded from CSIT Tutor

	Cost	Time
① for [i = n-1 ; i > 0 ; i--)	c_1	n
② for [j = 0 ; j < i ; j++)	c_2	$\sum_{j=0}^{n-1} j$
③ if (A[j] > A[j+1])	c_3	$\sum_{j=0}^{n-2} j$
④ {	~~c_4~~	
⑤ temp = A[j] ;	c_5	$\sum_{j=0}^{n-3} j$
⑥ A[j] = A[j+1] ;	c_6	"
⑦ A[j+1] = temp ;	c_7	"
⑧ }	~~c_8~~	

① Worst Case :-

$$T(n) = c_1 * n + c_2 * \left(\sum_{j=0}^{n-1} j\right) + c_3 * \left(\sum_{j=0}^{n-2} j\right) + c_5 * \left(\sum_{j=0}^{n-3} j\right) + c_6 * \left(\sum_{j=0}^{n-3} j\right) + c_7 * \left(\sum_{j=0}^{n-3} j\right)$$

$$= O(n) + O(n^2) + O(n^2) + O(n^2) + O(n^2)$$

$$= O(n^2)$$

② Best Case :-

$$T(n) = c_1 * n + c_2 * \left(\sum_{j=0}^{n-1} j\right) + c_3 * \left(\sum_{j=0}^{n-2} j\right).$$

$$= \Omega(n) + \Omega(n^2) + \Omega(n^2)$$

$$= \Omega(n^2).$$

③ Average Case :-

$$T(n) = \Theta(n^2/2) \approx \Theta(n^2)$$

Downloaded from CSIT Tutor

Heap Sort :-

1) Constructing heap (max).
2) Building a Heap.
3) Sorting a heap.

2) Building a heap:-

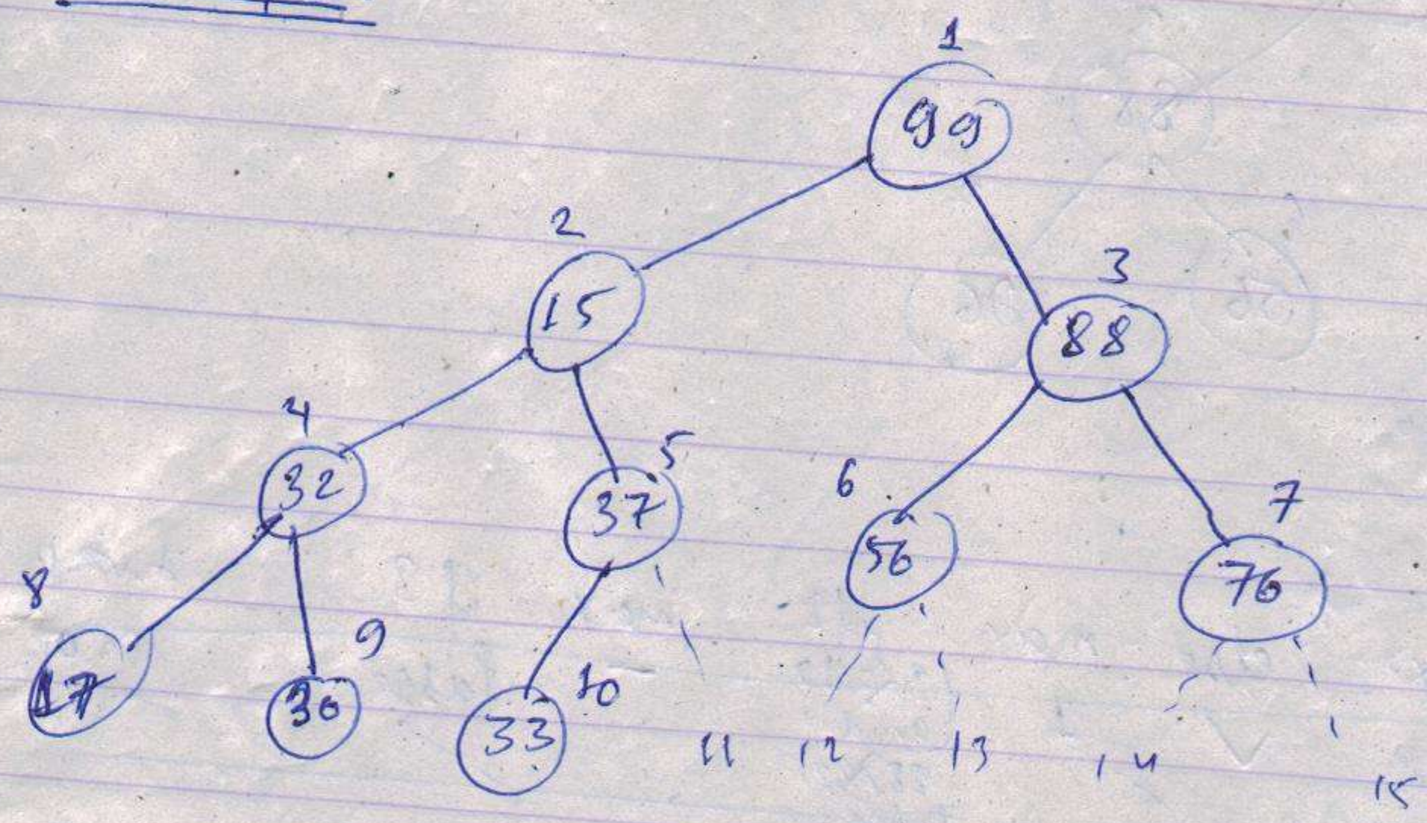
Algorithm :-

```
Buildheap (A)
{
    heapsize [A] = length [A]
    for i = ⌊ length [A] / 2 ⌋ to 1
        do heapify (A, i).
}

Heapify (A, i)
{   l = left (i)
    r = right (i)
    if ( l <= heapsize [A] and A[l] > A[i])        Suppose (if 1)
        max = l
    else
        max = i
    if (r <= heapsize [A] and A[r] > A[max])        (if 2)
        max = r
    if (max != i)                                   (if 3)
    {   swap (A[i], A[max])
        Heapify (A, max).
    }
}
```

Downloaded from CSIT Tutor

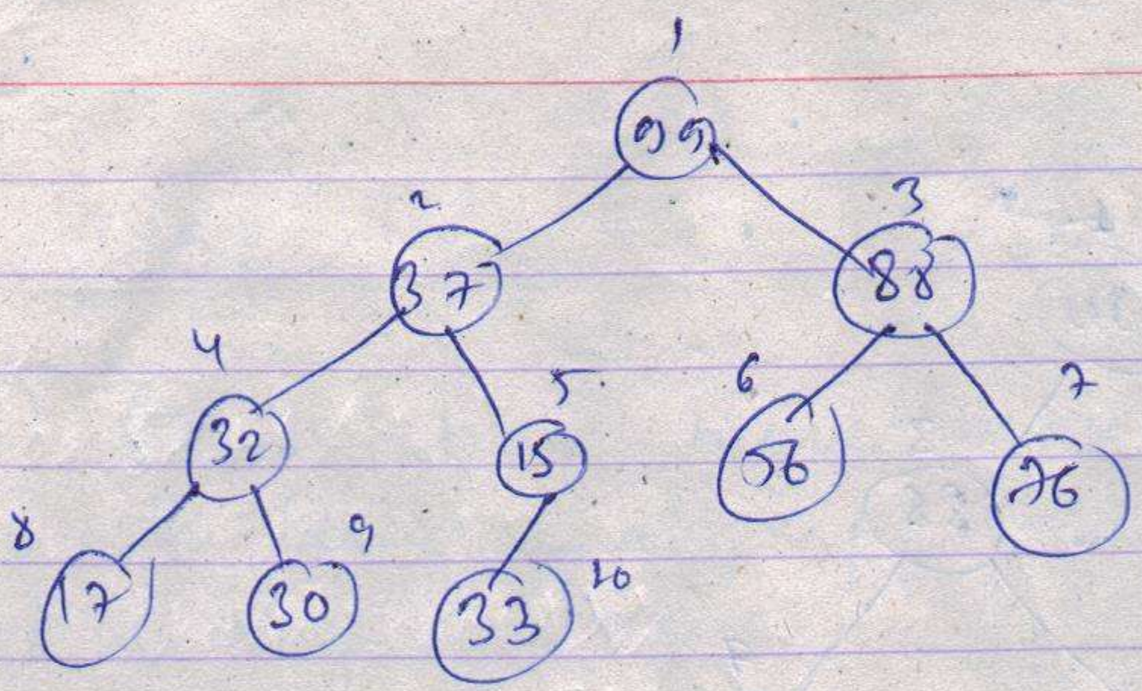
Example:-



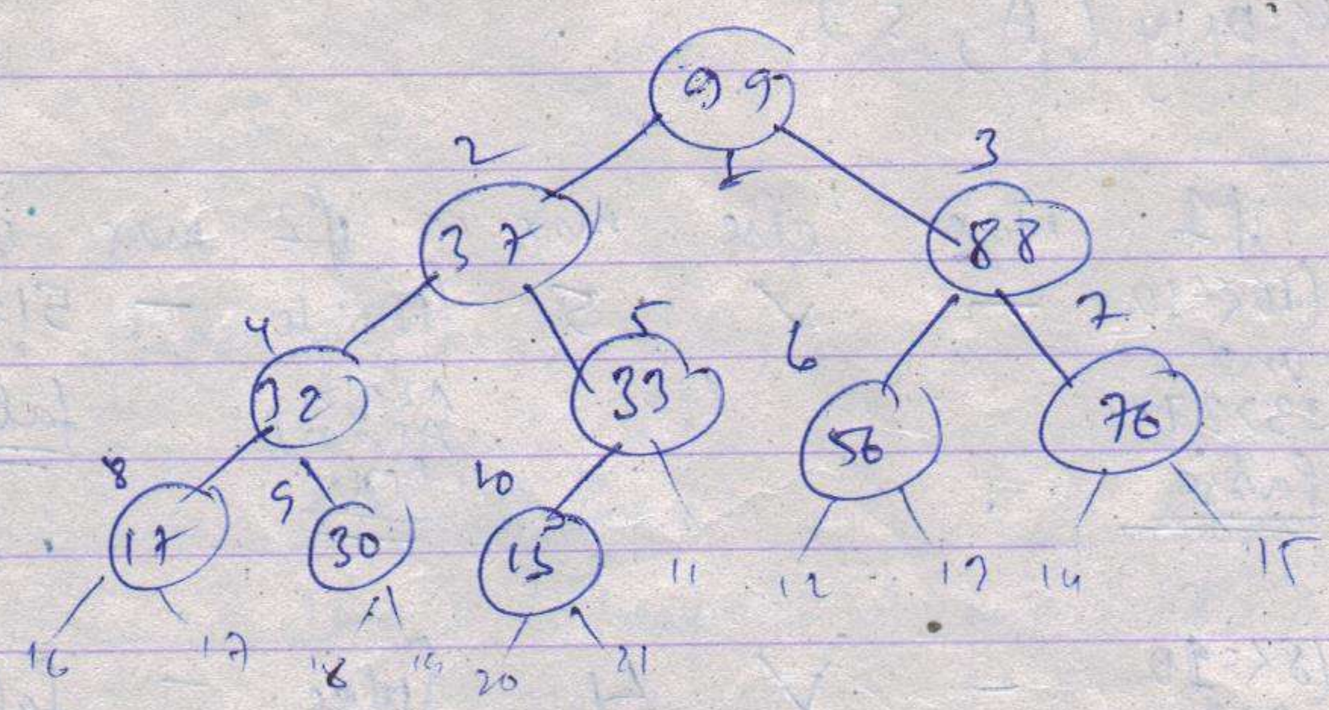
heapsize [A] = 10.
for(i = 10/2 = 5 to 1).
do heapify (A, 5)

i	l	r	if1	max	else	max	if2	max	if3	swap.
5	10	11	(10<=10 and 33>37) false	—	✓	5	(11<=10 and A[r]> A[max]) false	—	5!=5 false	No
4	8	9	(8<=10 and 17>32) false	—	✓	4	false	—	false	No
3	6	7	(6<=10 and 56>88) false	—	✓	3	false	—	false	No
2	4	5	(4<=10 and 32>15) true	32	false	—	(5<=10 and 37>32) true	5	(5!=2) true	Yes swap (37, 15) Heapify (A, 5)

Downloaded from CSIT Tutor



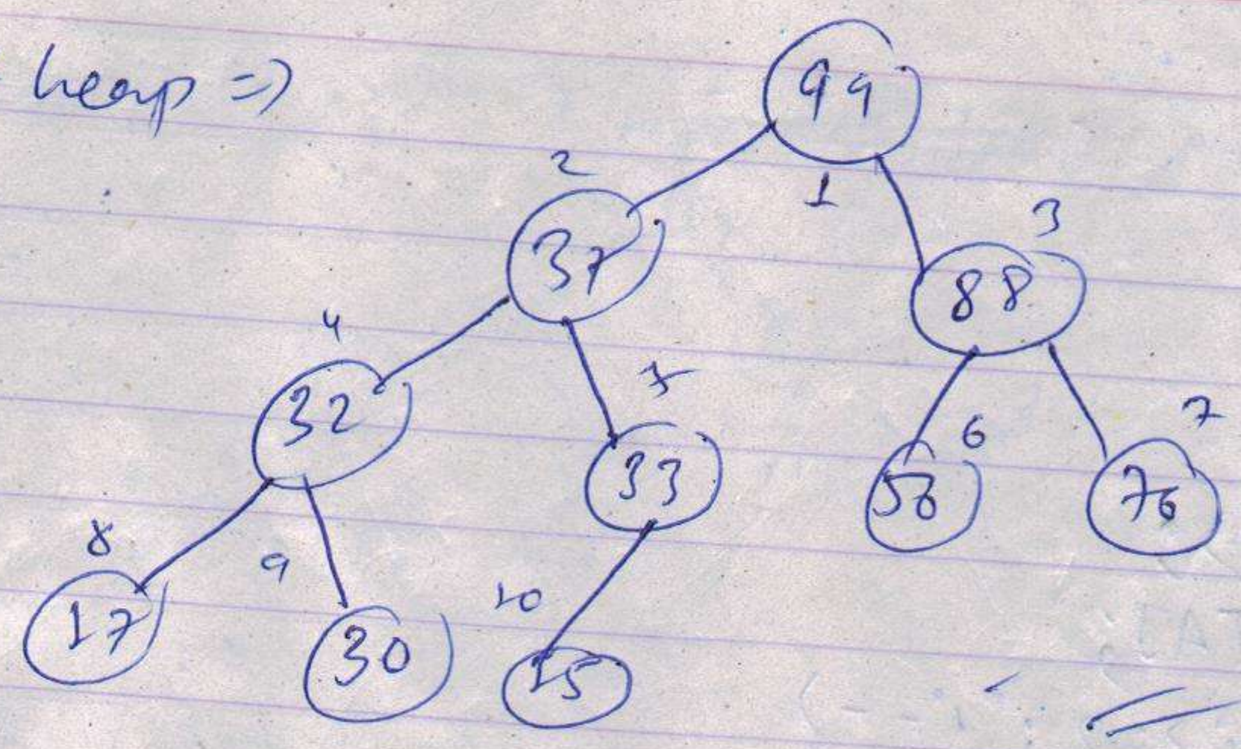
i	l	r	if 1	max	else	max	if 2	max	if 3	swap
1	2	3	(2<=10 and 37>99) false	—	✓	1	(3<=10 and 88>99) false	—	false	No
5	10	11	(10<=10 and 33>15) true	10	false	—	(11<=10) false	—	(10!=5) true	swap(15, Heapify (A, 10)



10	20	21	(20<=10) false	—	✓	10	(21<=10) (false)	—	false	No
1	2	3	false	—	✓	1	false	—	false	No

Downloaded from CSIT Tutor

Max heap =>



(Now do Heapsort (A) algorithm after this page)

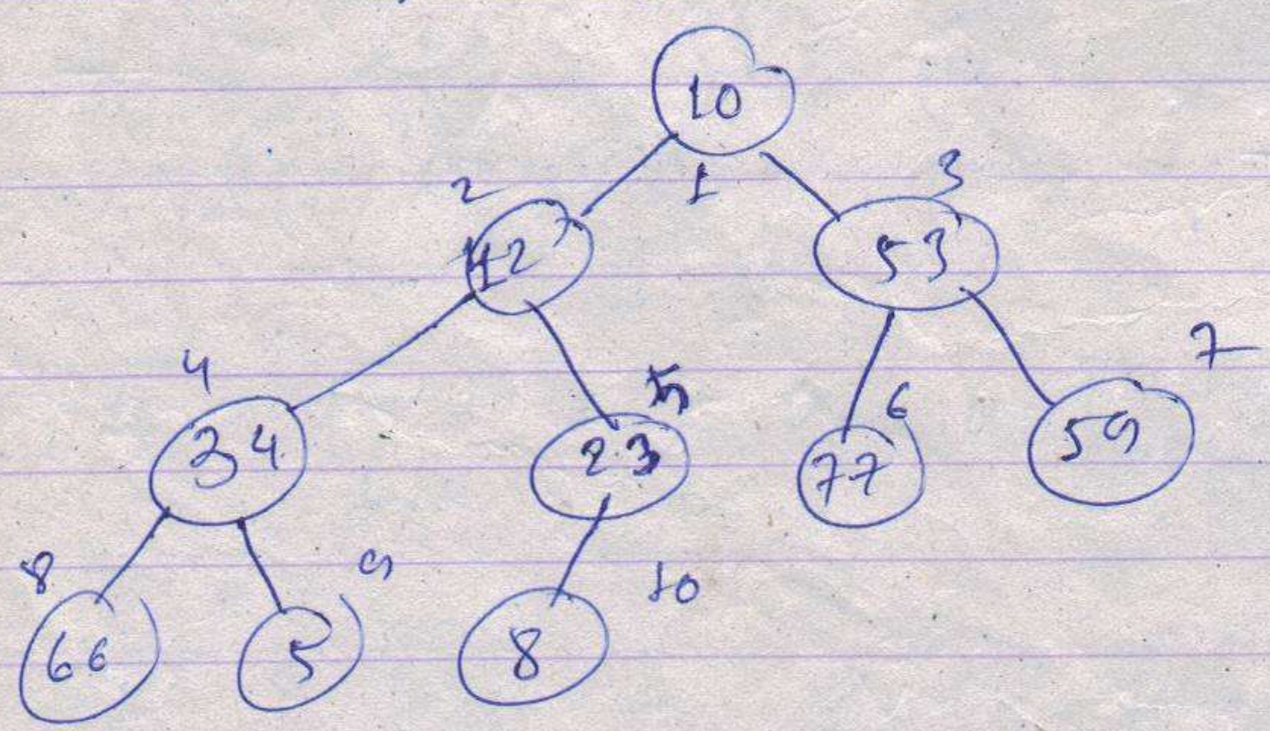
Downloaded from CSIT Tutor

Heapsort

```
HeapSort(A)
{
    Buildheap(A);
    m = length[A];
    for (i=m; i>=2; i--)
    {
        Swap(A[1], A[m])
        m=m-1;
        Heapify(A,1);
    }
}
```

Example:-

A[] = {10, 12, 53, 34, 23, 77, 59, 66, 5, 8}

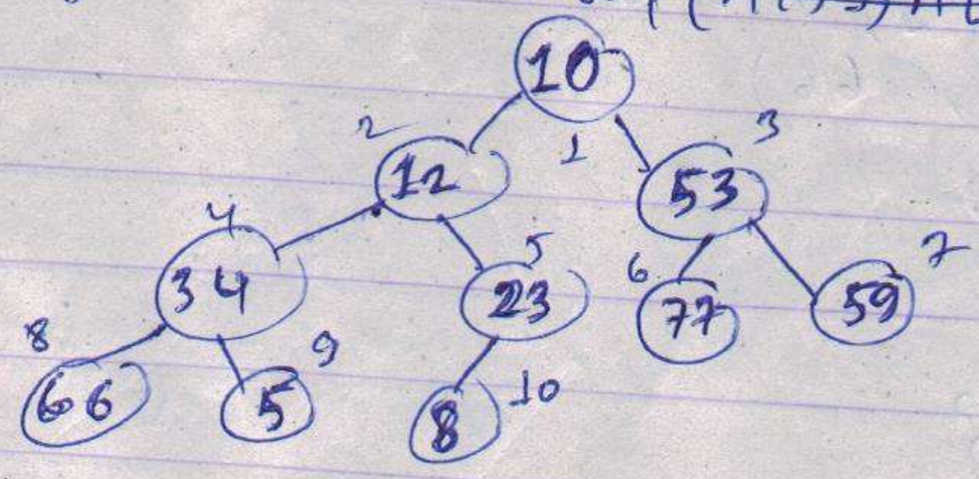


→ Convert to max heap first.

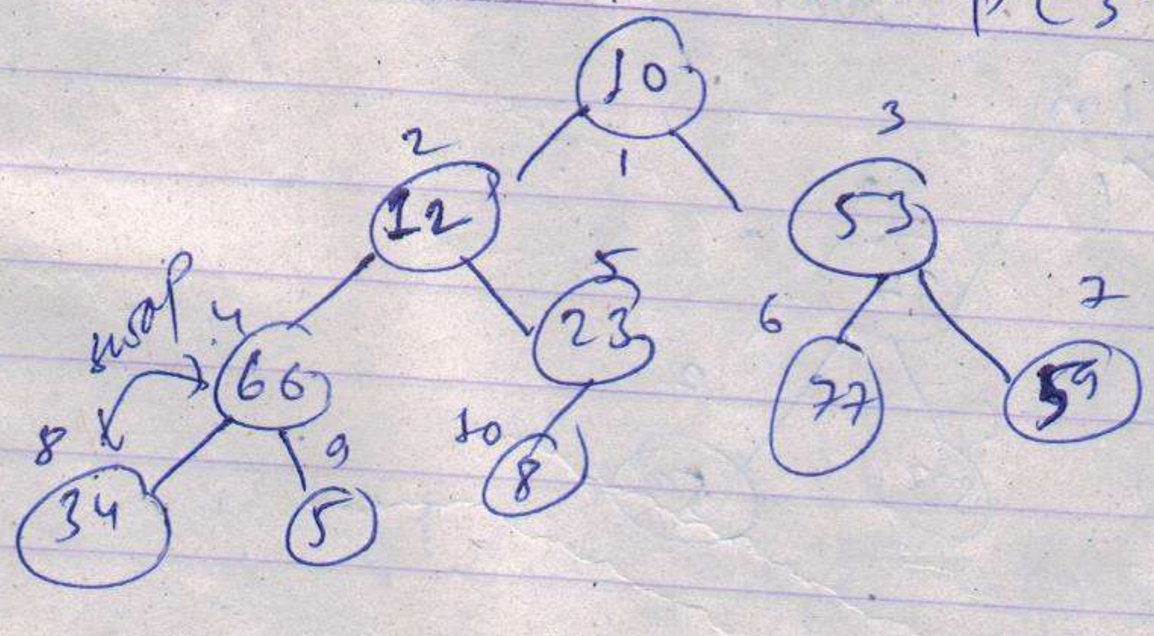
```
Buildheap(A)
{   heapsize[A] = 10
    i = 10/2 = 5 to 1;
        do Heapify.(A, 5)
}
```

Downloaded from CSIT Tutor

Heapify (A, 5) → ~~swap(A[5], A[~~ NO swap.

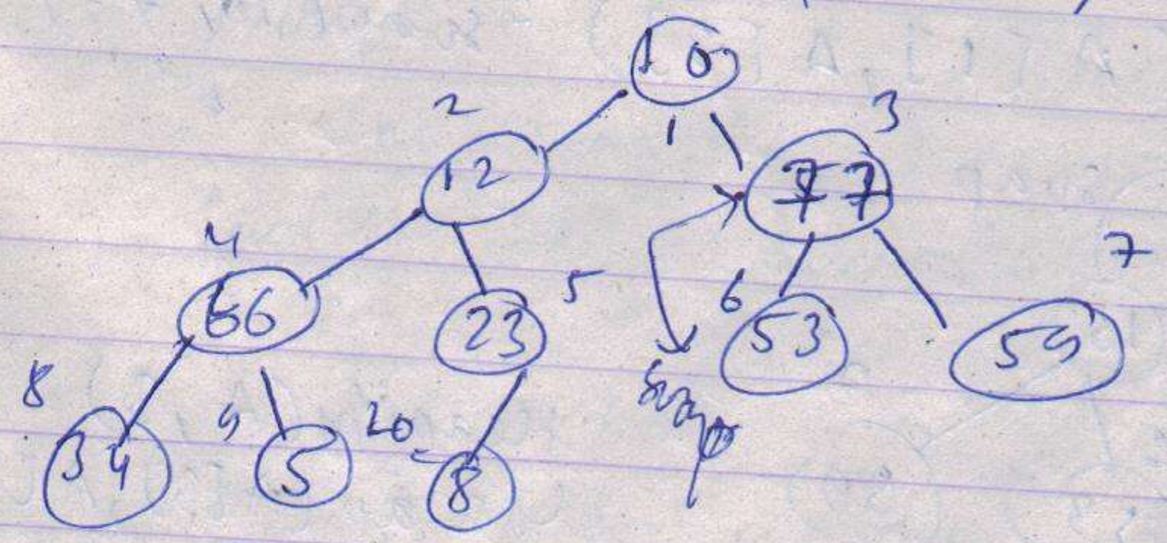


Heapify (4, 4) → swap(A[i], A[max])
→ swap(A[4], A[8])
→ swap(34, 66)



Heapify(A, 8)
(same as this).

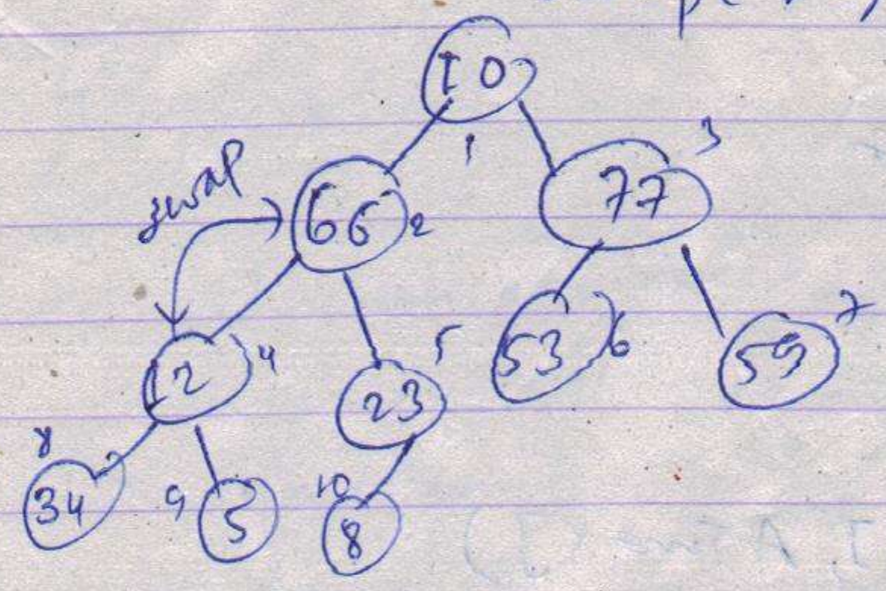
Heapify(A, 3) → swap(A[i], A[max])
→ swap(A[3], A[6])
→ swap(53, 77)



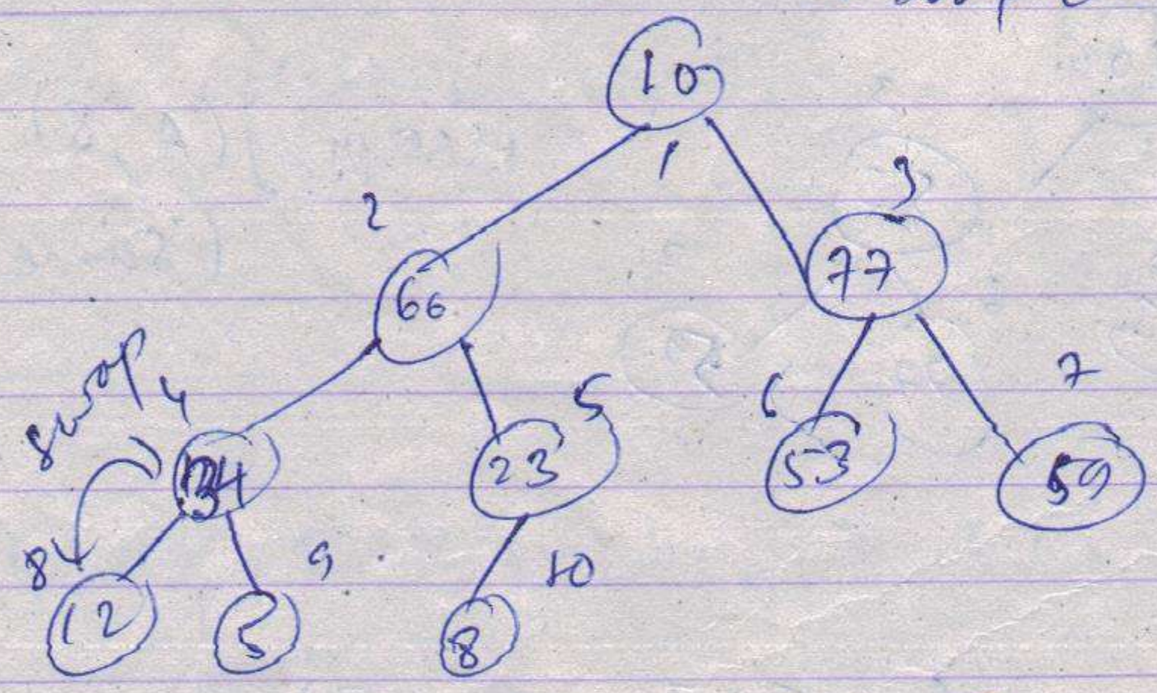
Heapify (A, 6)
(Same as before)

Downloaded from CSIT Tutor

Heapify(A, 2) → Swap(A[2], A[4])
→ swap(12, 66)

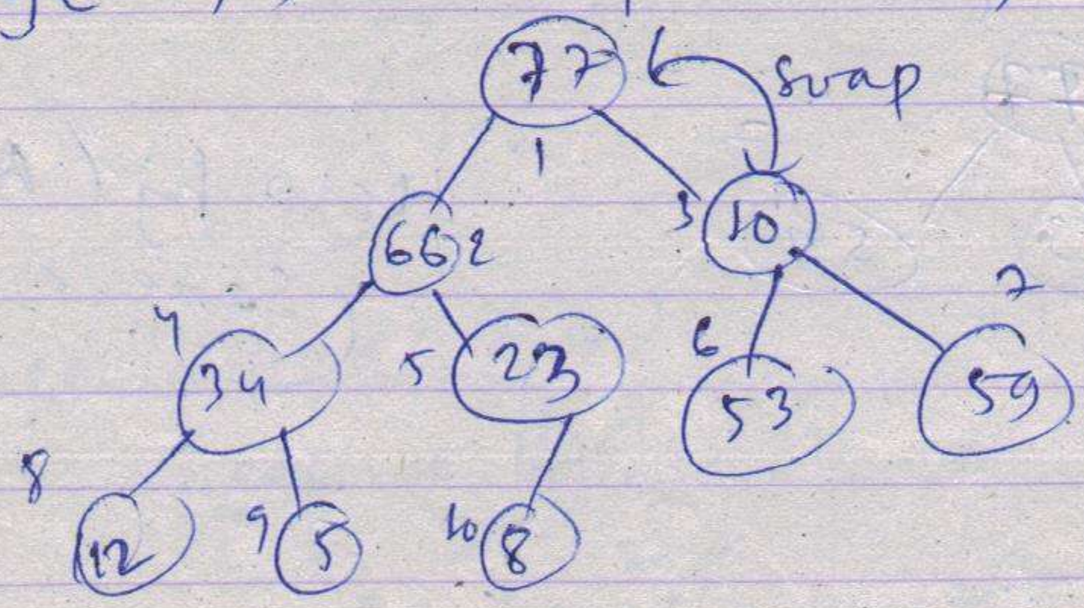


Heapify(A, 4) → swap[A[4], A[8]]
→ swap(12, 34)



Heapify(A, 8)
(same as above).

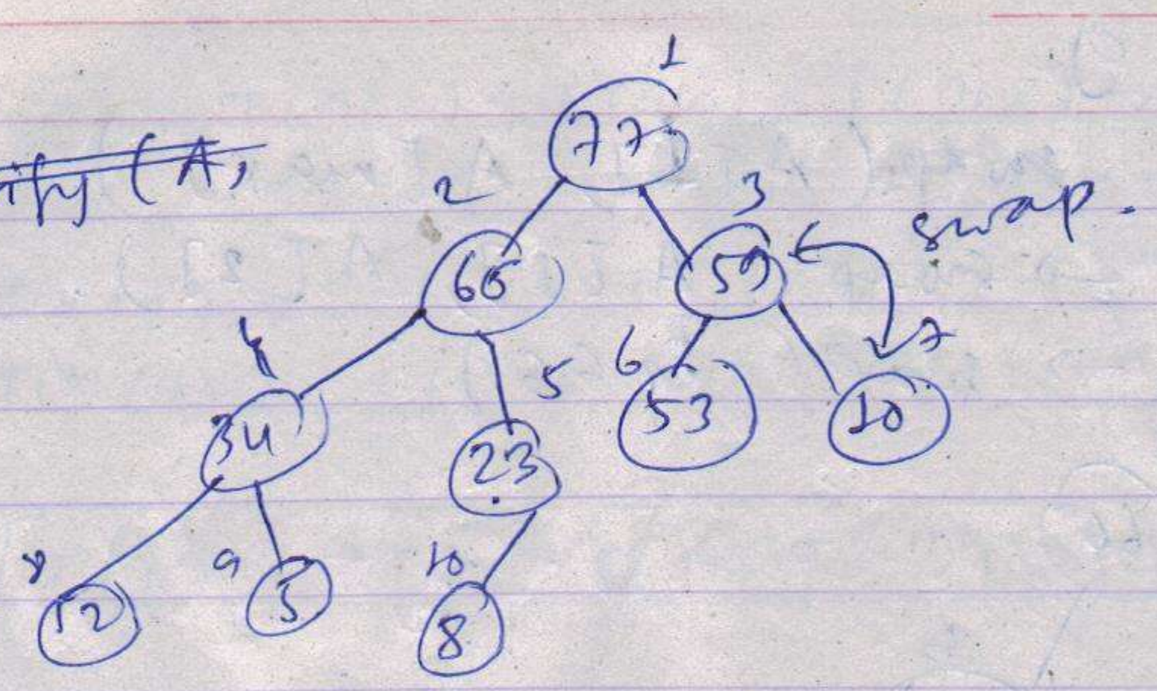
Heapify(A, 1) → swap(A[1], A[3]) → swap(10, 77)



Heapify(A, 3)
↳ swap(A[3], A
↳ swap(10, 59

Downloaded from CSIT Tutor

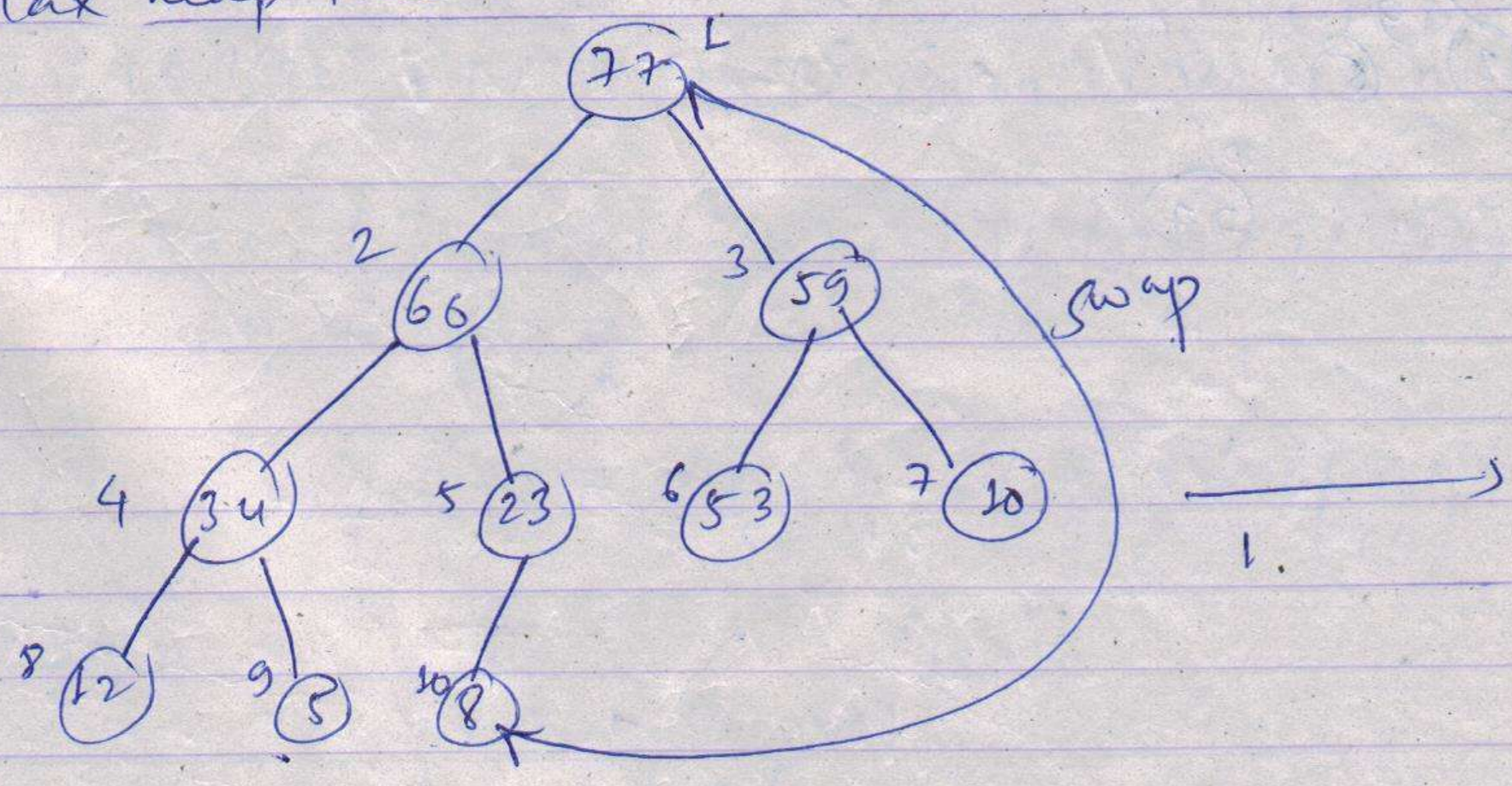
~~Heapify (A,~~



Heapify(A, 7)

Same as above.

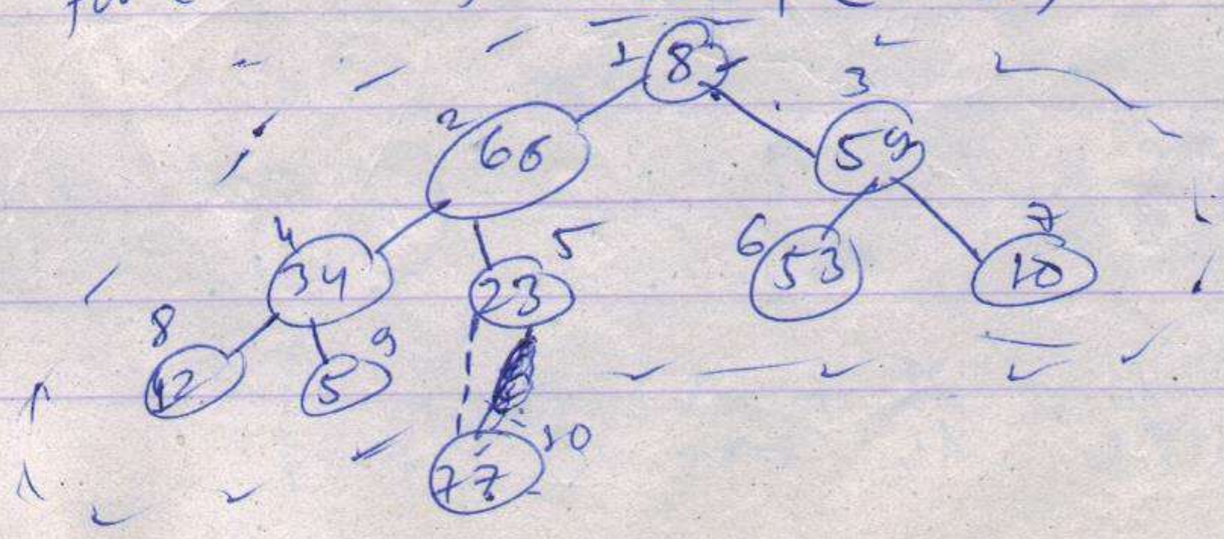
Max heap tree :-



Now,

m = length [A] = 10.

for (m = 10) → swap(A[1], A[10]) → swap(77, 8)



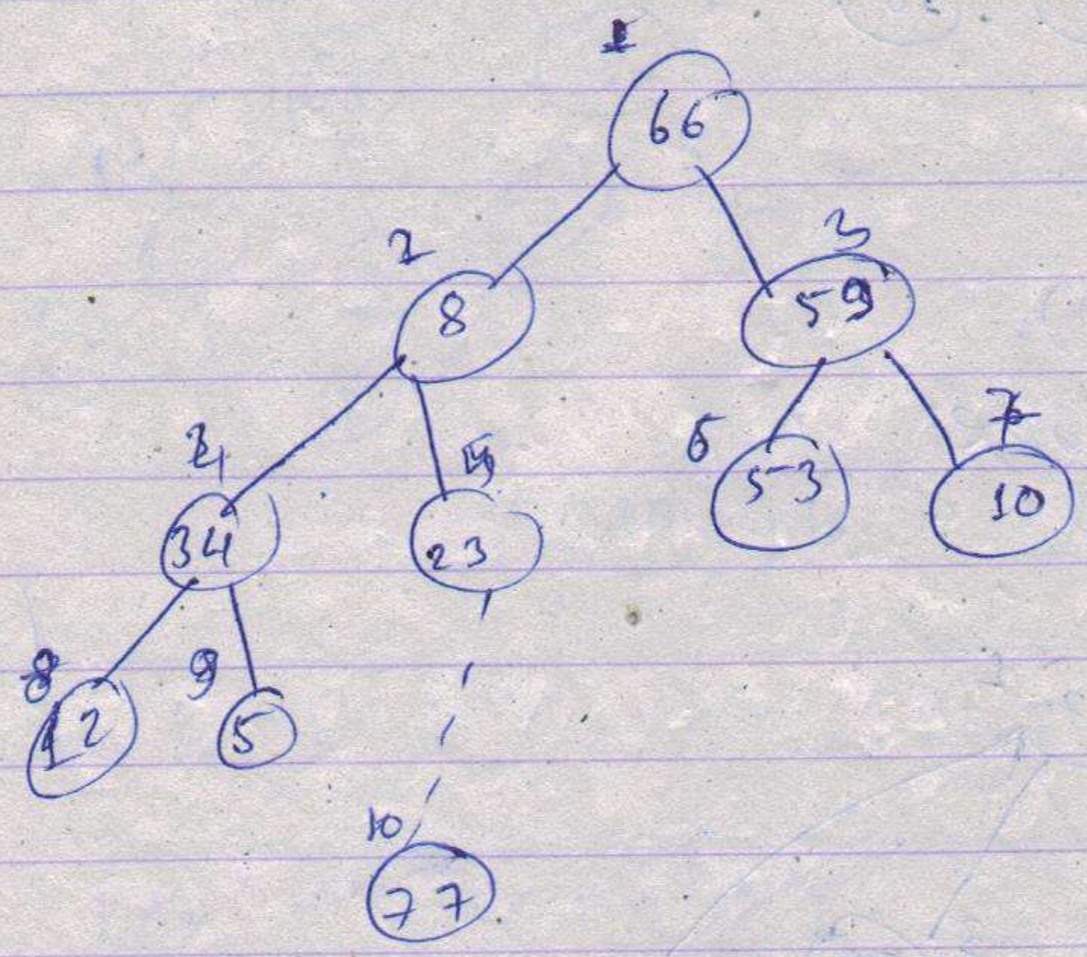
Downloaded from CSIT Tutor

$m = m - 1 = 9$

Heapify(A, 1) → swap(A[1], A[max])

⇒ swap(A[1], A[2]).

→ swap(8, 66)



Heapify(A,

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

Now,

$T(n/2) = T(n/4) + T(n/8) + T(n/16) + n$

$T(n/4) = T(n/16) + T(n/64) + T($

$= [T(n/4) + T(n/8) + T(n/16) + n/2] + [T(n/8) + T(n/16) + T(n/32) + n/4] + [T(n/16) + T(n/32) + T(n/64) + n/8] + n$

$= T(n/4) + T(n/8) + T(n/16) + T(n/8) + T(n/16) + T(n/32) + T(n/16) + T(n/32) + T(n/64) + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n$

$= T(n/4) + 2T(n/8) + 3T(n/16) + 2T(n/32) + T(n/64) + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n$

$= T\left(\frac{n}{4}\right) + 2T\left(\frac{n}{8}\right) + 3T\left(\frac{n}{16}\right) + 2T\left(\frac{n}{32}\right) + T\left(\frac{n}{64}\right) + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n.$

$= T\left(\frac{n}{2^2}\right) + 2T\left(\frac{n}{2^3}\right) + 3T\left(\frac{n}{2^4}\right) + 2T\left(\frac{n}{2^5}\right) + T\left(\frac{n}{2^6}\right) + \frac{n}{2^1} + \frac{n}{2^2} + \frac{n}{2^8} + \frac{n}{2^0}$

upto k-terms,

$T(n) = T\left(\frac{n}{2^k}\right) + 2T\left(\frac{n}{2^{k+1}}\right) + 3T\left(\frac{n}{2^{k+2}}\right) + 2T\left(\frac{n}{2^{k+3}}\right) + T\left(\frac{n}{2^{k+4}}\right) + \frac{n}{2^{k-1}} + \frac{n}{2^k} + \frac{n}{2^{k+1}} + \frac{n}{2^0}$

Downloaded from CSIT Tutor

$\frac{4+2+1}{8}$ $\left(\frac{7}{8}\right)^k n$

for $2^k = n$, $k = \log n$.

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

Sol^n:-

$T(n) = [T(n/4) + T(n/8) + T(n/16) + n/2] + [T(n/8) + T(n/16) + T(n/32) + n/4] + [T(n/16) + T(n/32) + T(n/64) + n/8] + n$

$= T(n/4) + 2T(n/8) + 3T(n/16) + 2T(n/32) + T(n/64) + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n$

$= \left(\frac{7}{8}\right)n + n$

$= [T(n/8) + T(n/16) + T(n/32) + n/4] + 2[2T(n/16) + 2T(n/32) + 2T(n/64) + \frac{n}{8}] + 3[3T(\frac{n}{32}) + 3T(\frac{n}{64}) + 3T(\frac{n}{128}) + \frac{n}{16}] + 2[2T(\frac{n}{64}) + 2T(\frac{n}{128}) + 2T(\frac{n}{256}) + \frac{n}{32}] + [T(n/128) + T(n/256) + T(n/512) + \frac{n}{64}] + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n.$

$= T(n/8) + 5T(n/16) + 14T(n/32) + 17T(n/64) + 14T(n/128) + 5T(n/256) + T(n/512) + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \frac{n}{4} + \frac{n}{8} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64} + n.$

$= T(\frac{n}{8}) + 5T(\frac{n}{16}) + 14T(\frac{n}{32}) + 17T(\frac{n}{64}) + 14T(\frac{n}{128}) + 5T(\frac{n}{256}) + T(\frac{n}{512}) + 2n + \frac{n}{4} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64}$

$\frac{n}{4} + \frac{n}{8} + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n = \frac{2n}{4} + \frac{2n}{8} + \frac{n}{2} + n = 2n + \frac{n}{4} +$

Downloaded from CSIT Tutor

(5) n^2

n^5

$$= \left[T\left(\frac{n}{8}\right)+T\left(\frac{n}{16}\right)+T\left(\frac{n}{32}\right)+\frac{n}{4}\right]+2\left[T\left(\frac{n}{16}\right)+T\left(\frac{n}{32}\right)+T\left(\frac{n}{64}\right)+\frac{n}{8}\right]$$

$$+3\left[T\left(\frac{n}{32}\right)+T\left(\frac{n}{64}\right)+T\left(\frac{n}{128}\right)+\frac{n}{16}\right]+2\left[T\left(\frac{n}{64}\right)+T\left(\frac{n}{128}\right)\right.$$

$$\left.T\left(\frac{n}{256}\right)+\frac{n}{32}\right]+\left[T\left(\frac{n}{128}\right)+T\left(\frac{n}{256}\right)+T\left(\frac{n}{512}\right)+\frac{n}{64}\right]+$$

$$\cancel{\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+n} \qquad \left(\frac{7}{8}\right)^1 n + \left(\frac{7}{8}\right)^0 n$$

$$= T\left(\frac{n}{8}\right)+3T\left(\frac{n}{16}\right)+6T\left(\frac{n}{32}\right)+7T\left(\frac{n}{64}\right)+6T\left(\frac{n}{128}\right)$$

$$+3T\left(\frac{n}{256}\right)+T\left(\frac{n}{512}\right)+2n+\frac{n}{4}+\frac{n}{16}+\frac{n}{32}+\frac{n}{64}$$

$$= T\left(\frac{n}{2^3}\right)+3T\left(\frac{n}{2^4}\right)+6T\left(\frac{n}{2^5}\right)+7T\left(\frac{n}{2^6}\right)+6T\left(\frac{n}{2^7}\right)+$$

$$3T\left(\frac{n}{2^8}\right)+T\left(\frac{n}{2^9}\right)+ \cancel{2n+\frac{n}{4}+\frac{n}{16}+\frac{n}{32}+\frac{n}{64}} \left(\frac{7}{8}\right)^k n +$$

$$- - - \qquad = \left(\frac{7}{8}\right)^1 n + \left(\frac{7}{8}\right)^0 n.$$

upto k-terms,

$$= T\left(\frac{n}{2^k}\right)+3T\left(\frac{n}{2^{k+1}}\right)+6T\left(\frac{n}{2^{k+2}}\right)+7T\left(\frac{n}{2^{k+3}}\right)+$$

$$6T\left(\frac{n}{2^{k+4}}\right)+3T\left(\frac{n}{2^{k+5}}\right)+T\left(\frac{n}{2^{k+6}}\right)+\cancel{\frac{n}{4}+\frac{n}{16}+\frac{n}{32}+}$$

$$\cancel{\frac{n}{64}+2n} \left(\frac{7}{8}\right)$$

For $2^k = n$, $T(n) = 1$.

$$= T(1)+3T\left(\frac{n}{2^{k+1}}\right)+ \cdots + 3T\left(\frac{n}{2^{k+5}}\right)+\frac{n}{4}+\frac{n}{16}+\frac{n}{32}$$

$$+\frac{n}{64}+2n.$$

Downloaded from CSIT Tutor

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

$\Rightarrow T(1) + 3T(1) + 6\,T\left(\frac{n}{2^{k+2}}\right) + \cdots + \frac{n}{4} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64} + 2n$

for $2^{k+2} = n$,

$\Rightarrow T(1) + 3T(1) + 6\,T(1) + 7T\left(\frac{n}{2^{k+3}}\right) + \cdots + \frac{n}{4} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64} + 2n$

for $2^{k+3} = n$,

$\Rightarrow 1 + 3 + 6 + 7\,T(1) + 6T\left(\frac{n}{2^{k+4}}\right) + \cdots + \frac{n}{4} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64} + 2n$

for $2^{k+4} = n$,

$\Rightarrow 1 + 3 + 6 + 7 + 6\,T(1) + 3T\left(\frac{n}{2^{k+5}}\right) + \cdots + \frac{n}{4} + \frac{n}{16} + \frac{n}{32} + \frac{n}{64} + n$

for $2^{k+5} = n$,

$\Rightarrow \left(1 + 3 + 6 + 7 + 6 + 3\,T(1)\right) + T\left(\frac{n}{2^{k+6}}\right)$ $\qquad \left(\frac{7}{8}\right)^i n + \left(\frac{7}{8}\right)^i n$

$= c + n \sum_{i=0}^{k} \left(\frac{7}{8}\right)^i$

Downloaded from CSIT Tutor

$n!$, 2^n, $4n^2$, n^5, $n^{2.02}$, $n^{1.2}$, $n^{0.5}$, $\log n$, $\log\log n$.

Q. Solve the recurrence relation using recurrence tree

$T(n) = T(n/4) + T(n/2) + n^3$; $n > 1$

$T(n) = 1$; $n = 1$

Q. Arrange the following expressions by growth rate from slowest to fastest.

$4n^2$, $\log n$, $\log n^2$, $\log\log n$, $n!$, n^5, 2^n, $n^{1.2}$, $n^{2.02}$, $n^{0.5}$.

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

Soln :-

Here, $T(n) = \left[T(n/4) + T(n/8) + T(n/16) + n/2\right] + \left[T(n/8) + T(n/16) + T(n/32) + n/4\right] + \left[T(n/16) + T(n/32) + T(n/64) + n/8\right] + n$

$= T(n/4) + 2T(n/8) + 3T(n/16) + 2T(n/32) + T(n/64) + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + n$

$= T(n/4) + 2T(n/8) + 3T(n/16) + 2T(n/32) + T(n/64) + 7/8\, n + n$

$= \left[T(n/8) + T(n/16) + T(n/32) + n/4\right] + 2\left[T(n/16) + T(n/32) + T(n/64) + n/8\right] + 3\left[T(n/32) + T(n/64) + T(n/128) + n/16\right] + 2\left[T(n/64) + T(n/128) + T(n/256) + n/32\right] + \left[T(n/128) + T(n/256) + T(n/512) + n/64\right] + \left(\frac{7}{8}\right)^1 n + \left(\frac{7}{8}\right)^0 n$

Downloaded from CSIT Tutor

$$= T\left(\frac{n}{8}\right) + 3T\left(\frac{n}{16}\right) + 6T\left(\frac{n}{32}\right) + 7T\left(\frac{n}{64}\right) + 6T\left(\frac{n}{128}\right) + 3T\left(\frac{n}{256}\right) + T\left(\frac{n}{512}\right) +$$

$$\left(\frac{7}{8}\right)^2 n + \left(\frac{7}{8}\right)^1 n + \left(\frac{7}{8}\right)^0 n$$

upto k-times,

$$= T\left(\frac{n}{2^k}\right) + 3T\left(\frac{n}{2^{k+1}}\right) + 6T\left(\frac{n}{2^{k+2}}\right) + 7T\left(\frac{n}{2^{k+3}}\right) + 6T\left(\frac{n}{2^{k+4}}\right)$$

$$+ 3T\left(\frac{n}{2^{k+5}}\right) + T\left(\frac{n}{2^{k+6}}\right) + \left(\frac{7}{8}\right)^{k-1} n + \cdots + \left(\frac{7}{8}\right)^0 n$$

for $2^k = n$,

$$= T(1) + 3T\left(n/2^{k+1}\right) + \cdots + n\left((7/8)^{k-1} + \cdots + (7/8)^0\right).$$

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

$$= 1 + 3T(1) + 6T\left(n/2^{k+2}\right) + \cdots + n\left[(7/8)^{k-1} + \cdots + (7/8)^0\right]$$

$$\cancel{= 1 + 3 +}$$

for $2^{k+2} = n$,

$$= 1 + 3 + 6T(1) + 7T\left(n/2^{k+3}\right) + \cdots + n\left[(7/8)^{k-1} + \cdots (7/8)^0\right]$$

for $2^{k+3} = n$,

$$= 1 + 3 + 6 + 7\,T(1) + 6T\left(n/2^{k+4}\right) + \cdots + n\left[(7/8)^{k-1} + \cdots + (7/8)^0\right]$$

$$\cancel{= 1 + 3 + 6 + 7 + 6T(1)}$$

for $2^{k+4} = n$,

$$= 1 + 3 + 6 + 7 + 6\,T(1) + 3T\left(n/2^{k+5}\right) + \cdots + n\left[(7/8)^{k-1} + \cdots + (7/8)^0\right]$$

for $2^{k+5} = n$,

$$= 1 + 3 + 6 + 7 + 6 + 3T(1) + T\left(n/2^{k+6}\right) + \cdots + n\left[(7/8)^{2k-1} + \cdots + (7/8)^0\right]$$

$$= 27 + n\left[(7/8)^{k-1} + \cdots (7/8)^0\right]$$

$$\Rightarrow 6 + n * \sum_{i=0}^{k-1} (7/8)^i$$

Downloaded from CSIT Tutor

$$\sum_{i=0}^{k-1} \left(\frac{7}{8}\right)^i = \frac{1}{1-a}, \quad 0 < a < 1$$

$$0 \leq \frac{7}{8} < 1$$

$$\Rightarrow \quad c + n * \frac{(7/8)^{k-1+1} - 1}{\left(\frac{7}{8} - 1\right)}$$

$$= c + n * \frac{(7/8)^k - 1}{(-1/8)}$$

$$= c - n * \left(\frac{7}{8}\right)^k - 1$$

$$c + n * \frac{1}{1 - 7/8}$$

$$c + n c$$

$$O(n)$$

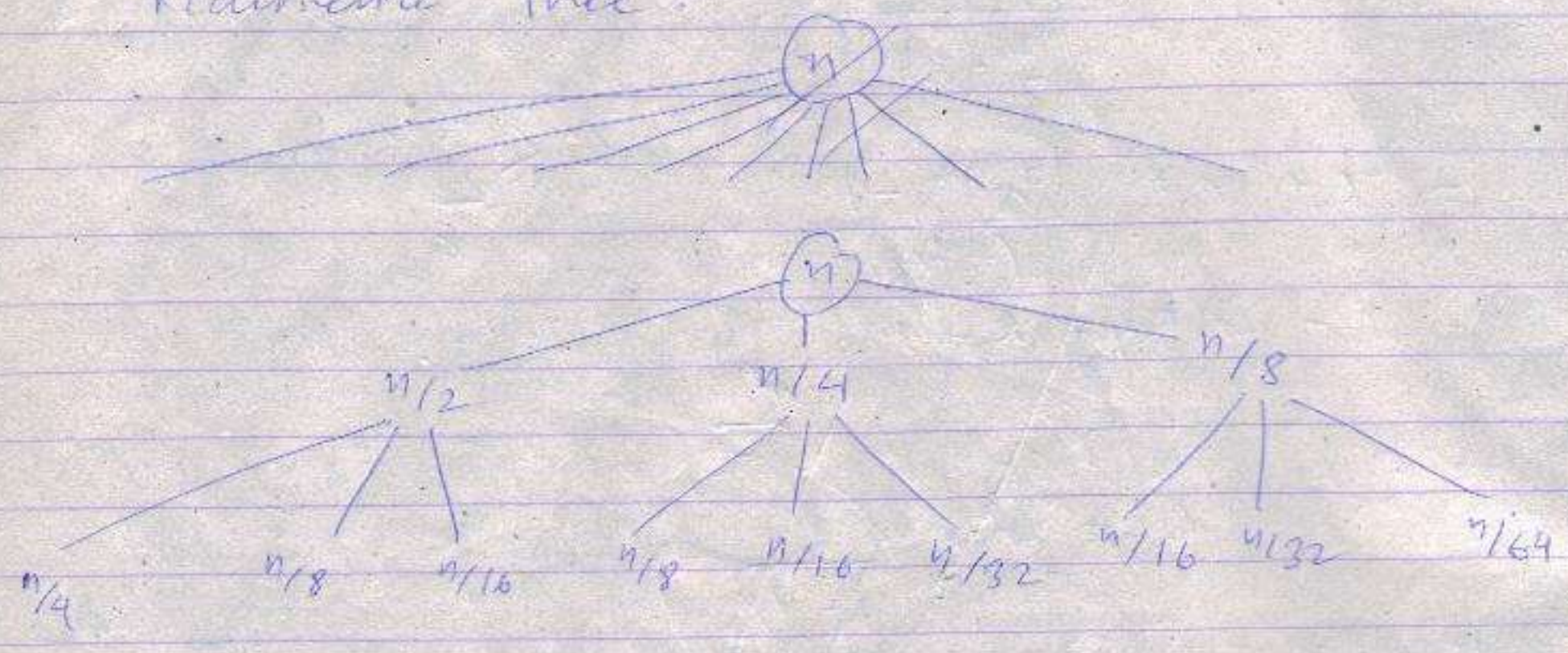
$$= c + n * \frac{1}{1 - \left(\frac{7}{8}\right)}$$

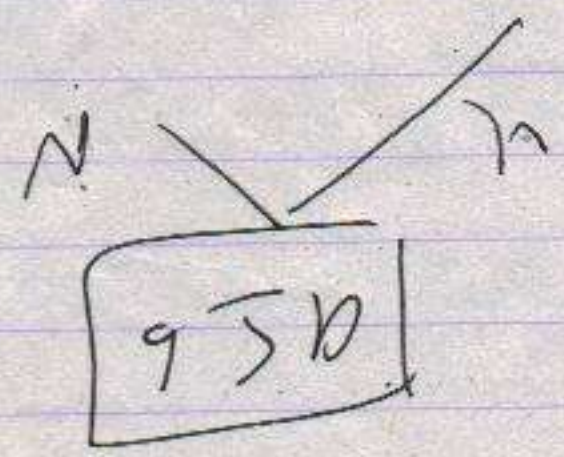
$$= c + n * c \quad \text{(for any } c\text{)}$$

$$= O(n).$$

Downloaded from CSIT Tutor

Recurrence Tree :-





Downloaded from CSIT Tutor

Analysis of Heap Sort :-

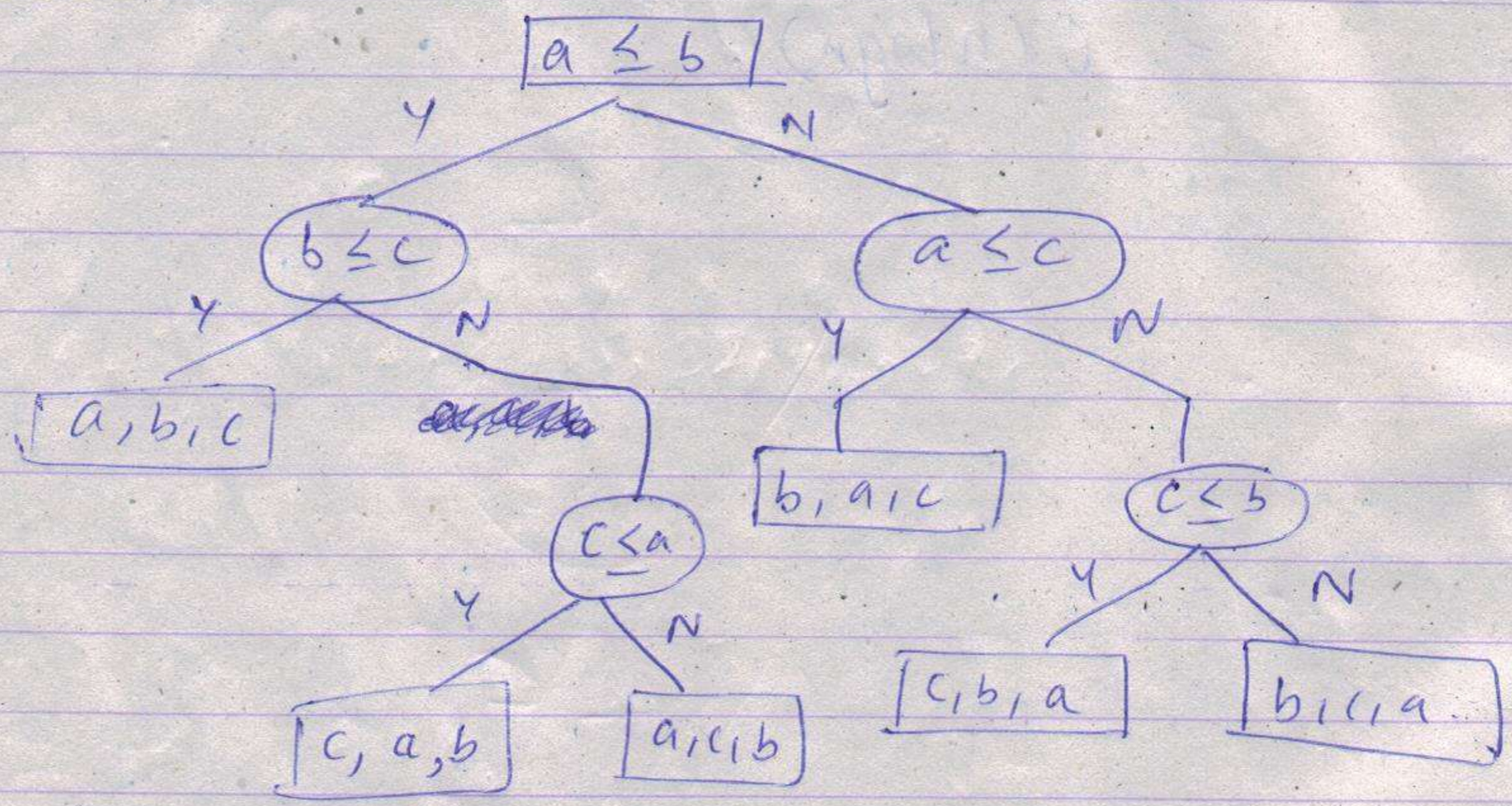
$$T(n) = O(n) + (n-1)(\log n)$$
$$= O(n) + n\log n - \log n$$
$$= O(n) + O(n\log n) + O(-\log n)$$
$$= O(n\log n).$$

Downloaded from CSIT Tutor

Lower bound for sorting :-

for 3 no.s a, b, c

Decision tree :-



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

This is the lower bound for sorting.

Downloaded from CSIT Tutor

Divide and Conquer Paradigms:-

Quick Sort:-

Algorithm:-

```
Quicksort(A, p, q)
{
    if (p<q)
    {
        r = Partition(A, p, q);
        Quicksort(A, p, r-1);
        Quicksort(A, r+1, q);
    }
}

                i   j
Partition(A, p, q)
{
    x = i-1; y = j+1 ; v = A[i];
    while (x < y)
    {
        do
        {
            x++;
        } while( A[x] <= v);
        do
        {
            y--;
        } while (A[y] >= v);
```

Downloaded from CSIT Tutor

```
if(x < y)
    swap (A[x], A[y]);
A[i] = A[y] ; A[y] = v;
return y ;
}
```

Example :-

	0	1	2	3	4	5	6	7	8	9	10
A[ ] =	16,	7,	6,	15,	14,	18,	25,	55,	32,	45,	37}

$i = 0, \quad j = 10$

$x = i - 1 = -1$

$y = j + 1 = 11.$

$v = A[i] = A[0] = 16.$

0	1	2	3	4	5	6	7	8	9	10
16	7	6	15	14	18	25	55	32	45	37

$p = 0, \quad q = 10, \quad v = A[i] = A[p] = A[0] = 16$

$x = p - 1 = -1, \quad y = 11.$

while	do1	while1	do2	while2	if(x<y)	swap	A[i]	A[y]
-1 < 11	x=0 ✓							
	x=1 ✓							
	x=2 ✓							
	x=3 ✓							
	x=4 ✓							
	x=5 ✗							
		y=10 ✓						
		y=9 ✓						

Downloaded from CSIT Tutor

while do1 while1 do2 while2 if(x<y) swap A[i] A[y] return y (pivot)

y=8 ✓

y=7 ✓

y=6 ✓

y=5 ✓

y=4 ✗

(5<4) (false) ✗ A[y] = A[4] = 14 ✓ = 16. 4

⇒ [14 7 6 15 16 18 25 55 32 45 37]

pivot

for [14, 7, 6, 15] ⇒ left.

0 1 2 3

p = 0, q = 3, v = A[p] = A[0] = 14

x = -1, y = 4.

while do1 while1 do2 while2 if(x<y) swap A[i] A[y] (return y) pivot

-1 < 4 x = 0 ✓

x = 1 ✓

x = 2 ✓

x = 3 ✗

y = 3 ✓

y = 2 ✗

(3<2) false ✗ A[y] = A[2] = 6 ✓ 2

⇒ [6, 7, 14, 15]

0 1 2 3

left 1 = [6, 7]

0 1

x = p - 1 = -1, y = 2

v = A[i] = 6. →

Downloaded from CSIT Tutor

while (-1 < 2) true

do 1 { x = 0 : } while 1 true

{ x = 1 } while 1 No.

do 2 { y = 1 } while 2 true

{ y = 0 } while 2 true

{ y = -1 } false.

x = 1, y = 0.

if (x < y)? No.

A[i] = A[y] = A[0] = 6

A[y] = v = 6

A[0] = A[y]

= A[0] = 6

A[y] = A[0] = v

= 6

⇒ [6, 7, 14, 15, 16, 18, 25, 35, 32, 45, 37]

sorted.

Right ⇒ [18, 25, 35, 32, 45, 37]

0 1 2 3 4 5

p = 0, q = 5, x = -1, y = 6, v = A[0] = 18

while do1 while1 do2 while2 if(x<y) swap A[i] A[y] return

(-1 < 6) ✓ x = 0 ✓

x = 1 ✗

y = 5 ✓

y = 4 ✓

y = 3 ✓

y = 2 ✓

y = 1 ✓

y = 0 ✓

y = -1 ✗ (1 < 0) ✗ ✗ A[0] = 18 0

Downloaded from CSIT Tutor

⇒ [18, 25 35 32 45 37]

Sorted ⇒ [6, 7, 14, 15, 16, 18].

Right half ⇒ [25, 35, 32, 45, 37]
0 1 2 3 4

p = 0, q = 4, x = -1, y = 5, v = 25

while	do1	while1	do2	while2	if (x < y) swap A[i] A[y] returning
(-1<5) ✓	x = 0	✓			
	x = 1	✗			
		y = 4	✓		
		y = 3	✓		
		y = 2	✓		
		y = 1	✓		
		y = 0	✓	(0 < 0)	✗ A[y] = A[0] = 25 — 0
		y = -1	✗		

⇒ [25, 55, 32, 45, 37]

Unsorted = [55, 32, 45, 37]
0 1 2 3

while (x < y) true, v = 55, x = -1, y = 4.

do { x = 0; true
x = 1; true
x = 2; false } ⇒ x = 1

do { y = 3; true
y = 2; true
y = 1; false } ⇒ y = 2.

if (x < y)? true
swap (A[1], A[2]) = (32, 45)
A[

while	do1 while1	do2 while2	if (x<y)	swap A[i] A[y]	returns
-1<4	x=0	✓			
	x=1	✓			
	x=2	✓			
	x=3	✓			

Analysis :-

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

Best Case :- $T(n) = 2T(n/2) + n$

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

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

$$= 2^2 [2T(n/2^3) + n/2^2] + 2^1 \cdot n/2^1 + 2^0 \cdot n/2^0$$

$$= 2^k T(n/2^k) + 2^{k-1} \cdot n/2^{k-1} + \cdots + 2^0 n/2^0$$

If $2^k = n$, $\Rightarrow$ $\log n = k \log_2 \Rightarrow k = \log n$.

$$= 2^k \cdot 1 + 2^{k-1} \cdot \frac{n}{2^{k-1}} + \cdots + 2^0 \cdot \frac{n}{2^0}$$

$$= n + \frac{n}{2} \cdot \frac{n}{n} \cdot 2 + n + n + \cdots + n$$

$$= kn = n \log n, \quad \therefore T(n) = \Omega(n \log n)$$

Downloaded from CSIT Tutor

$3 < 10$

$3 < 15$

$O(n)$

$\leq cn$

Worst Case :- $T(n) = T(n-1) + O(n)$.

Suppose $O(n) = n$

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

upto k-terms,

$$= T(n-k) + (n-k+1) + (n-k+2) \cdots + (n-0)$$

For $k = n-1$, $T(n) = 1$

$$= T(1) + (n-n+1+1) + (n-n+1+2) + \cdots (n-0)$$
$$= 1 + 2 + 3 + \cdots + n = \sum_{i=1}^{n} i$$
$$= \frac{n(n+1)}{2}$$
$$= \frac{n^2+n}{2}$$

Worst Case :- $T(n) = O(n^2)$.

Case when the partition is balanced but not as of best-case partition :-

Let us suppose that partitioning algorithm always produce 9:1 split of the array. For this situation, we can write recurrence relation as,

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + n.$$

Downloaded from CSIT Tutor