Name - Satvik Mittal

CST SPL - 2 (34)

Satv

Tutorial - 2

Sol 1. Values after execution

1st time $\Rightarrow$ $i = 1$

2nd time $\rightarrow$ $i = 1+2$

3rd time $\rightarrow$ $i = 1+2+3$

4th time $\rightarrow$ $i = 1+2+3+4$

for ith time $\rightarrow$ $i = (1+2+3+ \ldots\ i) < n$

$$\Rightarrow \frac{i(i+1)}{2} < n$$

$$\Rightarrow i^2 < n$$

$$i = \sqrt{n}$$

Time complexity $\Rightarrow O(\sqrt{n})$

Sol. 2.
```
int fib (int n)
{
    if (n<=1)
        return n;
    return fib((n-1))+ fib(n-2);
```

Satv

$\Rightarrow$ Recurrence Relation

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

let $T(n)$ denote the time complexity of $F(n)$

For $F(n-1)$ and $F(n-2)$, time will be $T(n-1)$ & $T(n-2)$. We have one more addition to sum results

For $n > 1$

$$T(n) = T(n-1) + T(n-2) + 1 \quad \text{-} (1)$$

For $n = 0$ & $n = 1$, no addition occurs

Page No.:

$T(0) = T(1) = 0$

Let $T(n-1) \approx T(n-2)$ — ②

Adding ② in ①

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

Using Backward Substitution

$\because T(n-1) = 2 \times T(n-2) + 1$
$T(n) = 2 \times [2 \times T(n-2) + 1] + 1$
$= 4T(n-2) + 3.$

We can substitute.

$T(n-2) = 2 \times T(n-3) + 1$
$\Rightarrow T(n) = 8 \times T(n-3) + 7$

General equation :-

$T(n) = 2^k \times T(n-k) + (2^k - 1)$ — ③

for $T(0)$
$n-k = 0 \Rightarrow k = n$

Substituting values in ③

Saho

Sol 3.

(i) $n(\log n)$

```
int sum = 0, n = 8;
for (int i = 1; i <= n; i++)
{
    for (int j = 1; j <= n; j *= 2)
    {
        sum += j;
    } }
```

(ii) n^3

```
int sum = 0 , n = 8;
for (int i = 1; i <= n; i++)
{
    for (int j = 1 ; j <= n ; j++)
    {
        for (int k = 0 ; k <= n ; k++)
        {
            sum ++;
        } } }
```

(iii) $\log(\log n)$

```
int sum = 0 , n = 8;
for (int i = 1 ; i <= n ; i *= 2)
{
    for (int j = 1; j <= n ; j *= 2)
    {
        sum += j;
    } }
```

Sahs

Sol. 4. $T(n) = T(n/4) + T(n/2) + cn^2$

Using Master's Theorom

We can assume $T(n/2) >= T(n/4)$

Equation can be rewritten as
$T(n) <= 2T(n/2) + cn^2$

$\Rightarrow T(n) <= O(n^2)$
$\Rightarrow T(n) = O(n^2)$
Also $T(n) >= cn^2 \Rightarrow T(n) >= O(n^2)$

$\Rightarrow T(n) = \Omega(n^2)$
$\because T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$

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

Sol. 5. for $i = 1$, inner loop is executed n times
for $i = 2$, " " " " $n/2$ "
for $i = 3$, " " " " $n/3$ times

It is forming a series :-

$\Rightarrow n + \frac{n}{2} + \frac{n}{3} + \ldots \frac{n}{n}$

$\Rightarrow n \times \sum_{K=1}^{n} \frac{1}{K}$

$\Rightarrow n \times \log n$
Time complexity $= O(n \log n)$

Sol. 6. for (int i = 2 ; i <= n ; i = pow(i, k))
{
O(1);
}

→ for 1st iterations = 2
for 2nd iteration = 2^k
for 3rd iteration = $(2^k)^k$

for n iteration → $2^{k \log(\log(n))}$

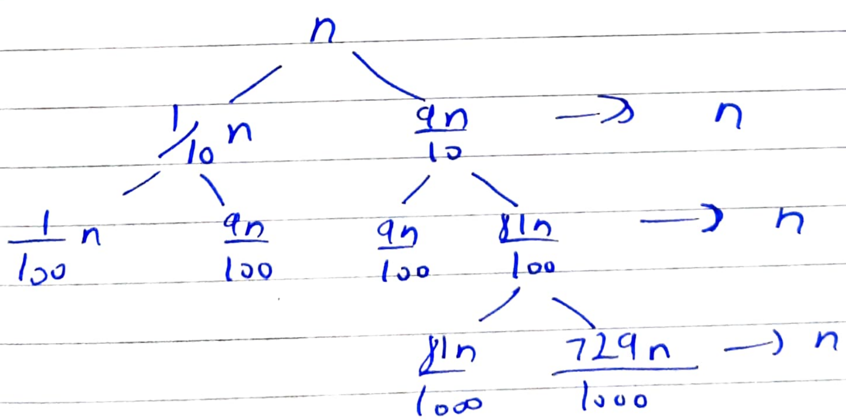
∵ Last term must be less than / equal to n

⇒ $2^{k \log_k(\log(n))} = 2^{\log n} = n$

Each iteration takes constant time
∴ Total iteration = $\log_k(\log(n))$

Time complexity = $O(\log(\log(n)))$

Sol 7.



If we split in this manner

Recurrence relation :-
$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + O(n)$$

- When first branch is of size $9n/10$ & second one is $n/10$

- Solving the above using recursion tree approach calculating values.

At 1st level, value $= n$
At 2nd level, value $= \frac{9n}{10} + \frac{n}{10} = n$

Value remains same at all levels.

Time complexity = Summation of values.

$= O(n \times \log_{10/9} n)$

$= \Omega(n \log_{10} n)$

$\Rightarrow O(n \log n)$

Sol. 8(a) $100 < \log(\log n) < \log(n) < \sqrt{n} < n < n\log n < \log^{\wedge} 2(n) < \log(\ln) < n^2 < 2^n < \ln < 4^n < 2^{2^n}$

(b) $1 < \log(\log(n)) < \sqrt{\log(n)} < \log(n) < 2\log(n) < \log(2n) < n < n\log(n) < \log(\sqrt{n}) < 2n < 4n < n^2 < 2(2^n)$

(c) $96 < \log_8(n) < n\log_6(n) < \log_2(n) < n\log_2(n) < \log(n!) < 5n < 8n^2 < 7n^3 < \ln < (8)^{2n}$

Satis