Nikhil Aswal Sec-f_17 2016877

Tutorial - 2

Ans.1. When while loop executes -

At first pass $i=1$

2nd pass $i=1+2$

3rd pass $i=1+2+3$

Similarly, 4th $i=1+2+3+4$

$n^{th}$ $i=1+2+3+\dots+n$

for $i^{th}$ time $i=(1+2+3+4+\dots i)<n$

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

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

ignoring $\frac{i}{2}$ & $\frac{1}{2}$

After neglecting, we left with

$$= i^2 < n$$

$$= i < \sqrt{n}$$

Hence the time complexity is $O(\sqrt{n})$

Ans.2.
```
int recfib (int n)
{
    if (n<=1)
        return n;
    else
        return recfib(n-1) + recfib(n-2);
}
```

time complexity :-

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

when $n=0$ & $n=1$

i.e. $T(0) = T(1) = 0$

for $T(n) = ?$

Here $T(n-2) \approx T(n-1)$

On substituting the value of $T(n-1) = T(n-2)$ in $T(n)$

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

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

On Substituting

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

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

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

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

$T(n) = 8 * T(n-3) + 7$

$\vdots$

$T(n) = 16 * T(n-4) + 15$

Similarly for $k^{th}$ term

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

$n-k=0$

$n=k$

Hence, $T(n) = 2^n * T(0) + (2^n - 1)$

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

So, the time complexity is $O(2^n)$.

Space Complexity :

Here n are the no. of entries in a stack & for each function call one.

So space complexity for each case is 1, i.e. $O(1)$

& for n. no. of case $<= n$

i.e. $O(n)$

Ans. 3. Which program have complexity —

— $n(\log n)$, $n^3$, $\log(\log n)$.

1. $n \log n$ — (Quick Sort)

```
void quicksort (int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition (arr, low, high);
        quicksort (arr, low, pi-high+1);
        quicksort (arr, low pi+1, high);
    }
}
int partition (int arr [], int low, int high)
{
    int pivot = arr[high];
    int i= low-1;
    for (int j= low; j<= high-1; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap (&arr[i], &arr[j]);
        }
    }
```

```
    swap(&arr[i+1], &arr[high]);
    return (i+1);
}
```

2. $n^3$ - (multiplication of 2 square matrix)

```
for(i=0; i<r1; i++)
{
    for (j=0; j<c2; j++)
        for (k=0; k<c1; k++)
        {
            res[i][j]+ = a[i][k] * b[k][j];
        }
}
```

3. log (log n)

```
for (i=2; i<n; i=i*i)
{
    count++;
}
```

Ans.4. Recurrence Relation -

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

On removing $T(n/4)$ as smaller term

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

on applying Master's theorem on RHS

$a=0, \quad b=2 \quad k=2, \quad p=0.$

$\log_b a = \log_2 0 = 0$

$0<2$ i.e. $\log_b a < k$

& $p \geq 0$

$\theta(n^k \log^p n)$

$O(n^2 \log^0 n)$

$O(n^2)$ Ans

Ans. 5. time complexity of function:

```
int fun( int n)
{
    for( int i=1;  i<=n ; i++)
    {
        for (int j=1; j<n; j++)
        {
            // Some O(1) task
        }
    }
}
```

→ for

| i | j |
|---|---|
| 1 | 1 |
| 2 | 1+3=5 |
| 3 | 1+4=7 |
| ⋮ | |
| n | |

$j = (n-1)/i$ times

$$\sum_{i=1}^{n} \left(\frac{n-1}{i}\right)$$

$$\therefore T(n) = \frac{(n-1)}{1} + \frac{(n-1)}{2} + \frac{(n-1)}{3} + \dots + \frac{(n-1)}{n}$$

$$T(n) = n\left[1 + \frac{1}{2} + \frac{1}{3} + \dots \frac{1}{n}\right] - 1 \times \left[1 + \frac{1}{2} + \frac{1}{3} + \dots \frac{1}{n}\right]$$

$$= n\log n - \log n$$

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

Ans. 6. What should be time complexity of

```
for ( int i=2; i<=n; i= pow (i,K))
{
    // Some O(1)
}
```

where K is constant

→ for $i$

$2^1$

$2^K$

$2^{K^2}$

$2^{K^3}$

⋮

$2^{K^m}$

where

$2^{K^m} <= n$

$K^m < \log_2 n$

$m = \log_K \cdot \log_2 n$

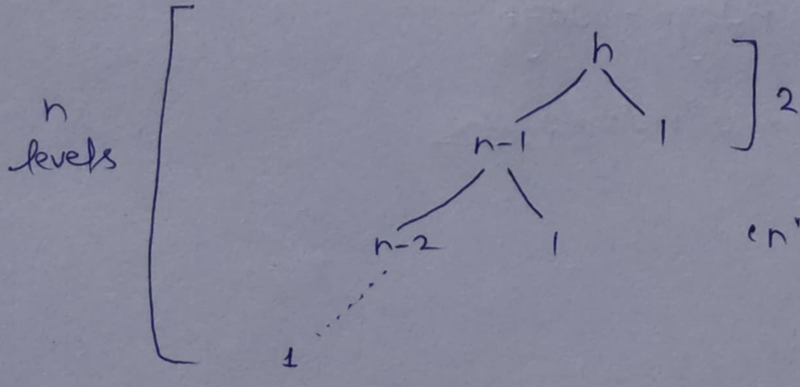
$\therefore \sum_{i=1}^{m} 1$

$1 + 1 + 1 + \ldots$ m times

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

Ans. 7. Given algorithm divides every array in 99% and 1% part

$\therefore T(n) = T(n-1) + O(1)$




'n' work is done at each level.

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

$= n \times n$

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

Lowest height = 2

highest height = n

$\therefore$ $\boxed{\text{difference} = n-2}$ $n>1$

The given algorithm produces linear result.

Ans. 8.

a. $n, n!, \log n, \log\log n, \text{root}(n), \log(n!), n\log n, \log^{2(n)}, 2^n,$
$2^{2^n}, 4^n, n^2, 100$

$\Rightarrow 100 < \log\log n < \log n < (\log n)^2 < \sqrt{n} < n < n\log n < \log(n!)$
$< n^2 < 2^n < 4^n < 2^{2^n}$

b. $2(2^n), 4n, 2n, 1, \log(n), \log(\log(n)), \sqrt{\log(n)}, \log 2n,$
$2\log(n), n, \log(n!), n!, n2, n\log(n)$

$\Rightarrow 1 < \log\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n$
$< 2n < 4n < \log(n!) < n^2 < n! < 2^{2^n}.$

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

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