Sravvat Shah

CST- SPL-1

Roll no. 39

DAA - Tutorial-2

1.) for $j = 1$ $\quad i = 1$

$j = 2$ $\quad i = 1+2$.

$j = 3$ $\quad i = 1+2+3$.

$\vdots$

$j = n$ $\quad i = 1+2+3+\dots+n$.

for while condition

$\therefore 1+2+3+\dots n < n$.

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

$$n^2 + n < 2n$$

$$n^2 < n.$$

$$n < \sqrt{n}. \quad n \approx \sqrt{n}$$

Using summation Method

$$\sum_{i=1}^{m} (1) \Rightarrow 1+1+1+\dots \sqrt{n} \text{ times}.$$

$$T(n) = O(\sqrt{n}).$$

2.) for fibonacci series:

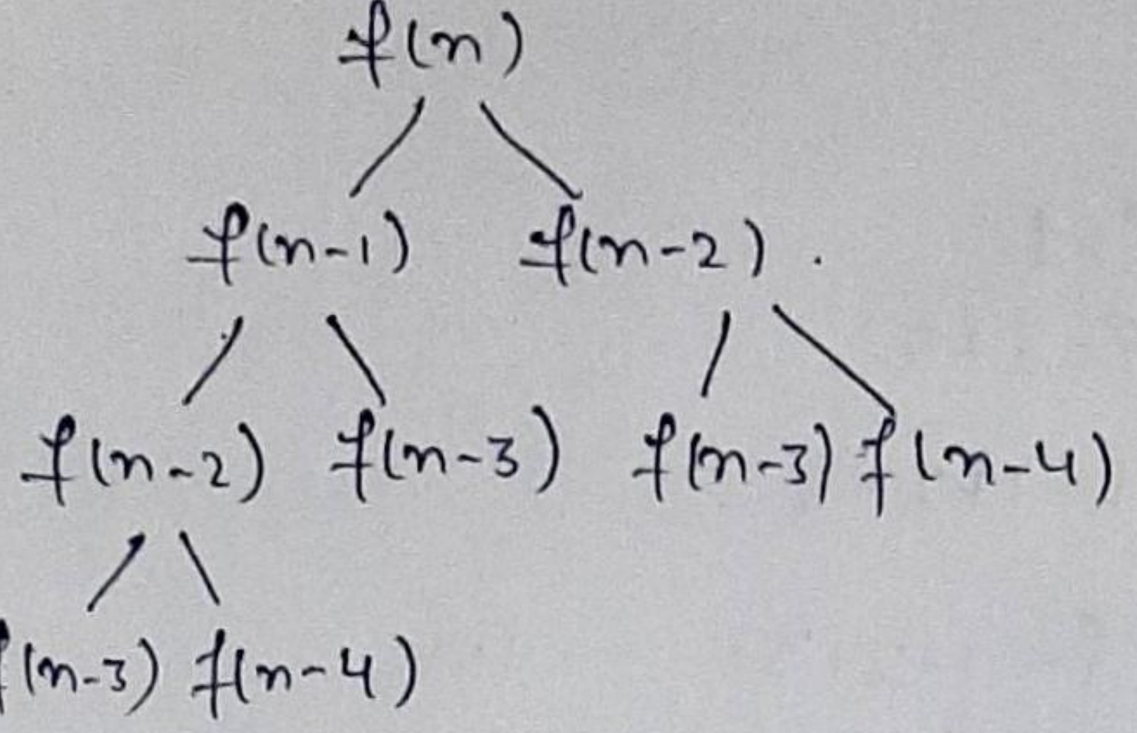
$$f(n) = f(n-1) + f(n-2).$$

$f(0) = 0$.

$f(1) = 1$.

Sravvat

Tree :



At every function call we get 2 function calls.
for 'n' levels.

We have, 2 * 2 * ---n times--- 2 times

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

Maximum Space complexity. O(n). for recursive call without recursive stack

$T(n) = O(1)$.

3.) ① $n \log n$.

Quick sort algorithm

```
void q_sort (int arr[], int low, int high).
{
    if (low < high).
    {
        int pi = partition (arr, low, high).
        q_sort (arr, low, pi-1);.
        q_sort (arr, pi+1, high);
    }
}.
```

Scanned by TapScanner

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

② n^3

Multiplication of two square matrix.

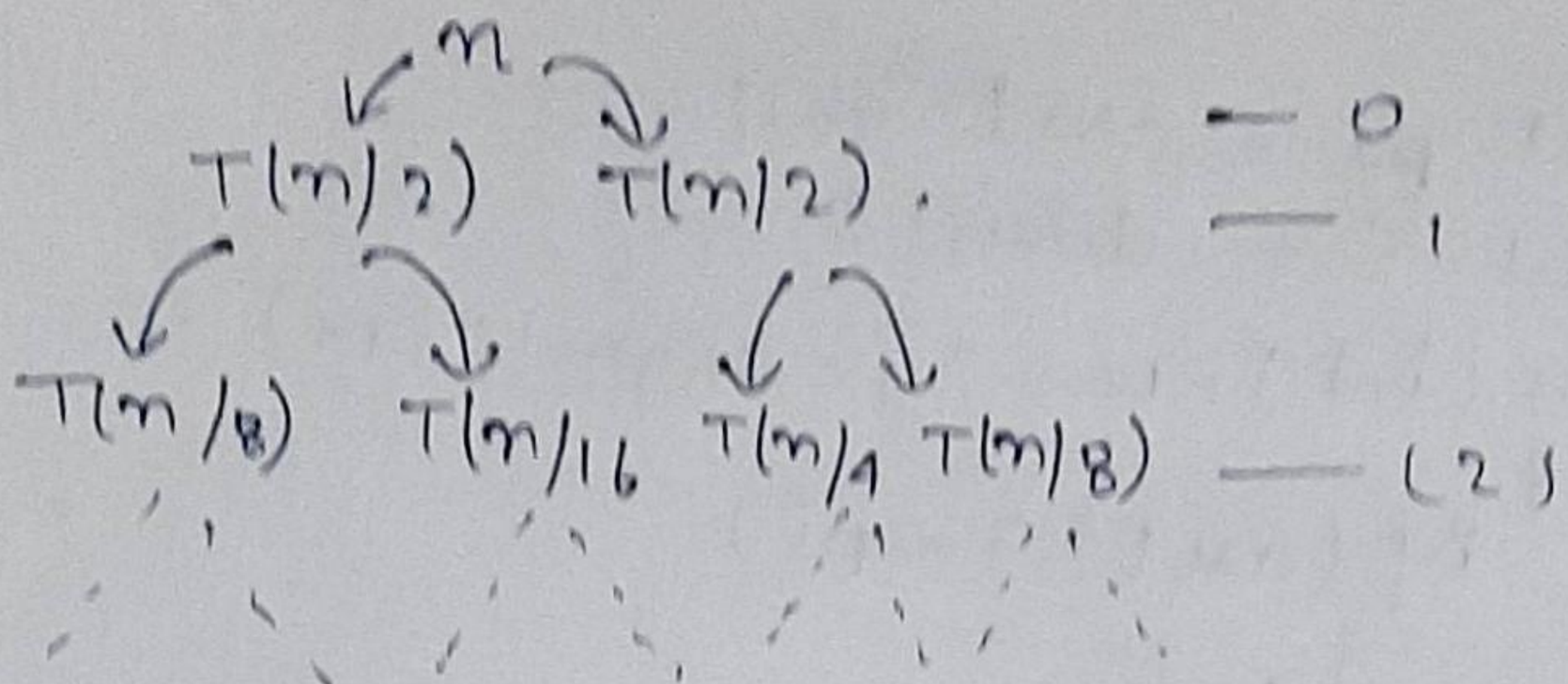
```
for ( i = 0; i < r1 ; i++ )
    for (j = 0 ; j < c2 ; j++ )
        for (k = 0 ; k < c1 ; k++ ).
          {
            res[i][j] = a[i][k] * b[k][j] ;
          }.
```

③ log(logn).

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

Scanned by TapScanner

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



— 0
— 1
— (2)

$0 \rightarrow cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{5cn^2}{16}$.

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 c$.

Max level $\Rightarrow \frac{n}{2^k} = 1$

$\Rightarrow k = \log_2 n$.

$\therefore T(n) = c\left(n^2 + (5/16)n^2 + (5/16)n^2 + \dots\right)$

$= c\left((5/n)^{\log n} n^2\right)$.

$T(n) = cn^2\left[(1) + \left(\frac{5}{16}\right)n^2 + \left(\frac{5}{16}\right)n^2 + \dots \left(\frac{5}{16}\right)^{\log n}\right)$.

$= cn^2\left(\frac{1 - (5/16)^{\log n}}{1 - 5/16}\right)$.

$= cn^2\left(\frac{11}{5}\right)\left(1 - \left(\frac{5}{16}\right)\right)^{\log n}$

$= O(n^2 c)$

$T(n) = \underline{O(cn^2)}$

Scanned by TapScanner

5° int fun(int n).

{

for(i=1; i<=n; i++).

for(j=1; j<n; j+=i).

{ //O(1)}.

i	j	
1	1	j = (n-1)/i times.
2	1+3+5	
3	1+4+7	

$\Rightarrow \sum_{i=1}^{n} \frac{(n-1)}{i} \Rightarrow T(n) = (n-1) + \frac{(n-1)}{2} + \cdots + \frac{(n-1)}{n}$.

$\Rightarrow T(n) = n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$.

$= n \log n$

$= O(n \log n)$.

6.) for(i=2; i<=n; i=(pow(i,k))){

//O(n)

}.

$i = 2^1$	$\therefore 2^{k^m} <= n$
$= 2^k$	$k^m = \log_2 n$
$= 2^{k^2}$	
$= 2^{k^3}$	$m = \log_k \log_2 n$

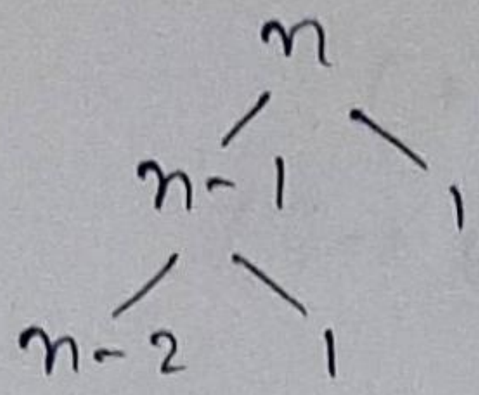
and O(1) → m times.

$\therefore T(n) = O(\log_k \log_2 n) + O(1)$

$= O(\log_k \log_2 n)$.

Srasvat

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



difference in height $= (n-2)$.

lowest height $= 2$
highest height $= n$.

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

$= n \times n = O(n^2)$.

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

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

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

Shashwat