Name- Jatin Jindal
Section- CST Spl2
Roll No.- 39

DAA (TCS-4

22-3-22

TUTORIAL-2

Date: / /

Q.1. What is the time complexity of below code and how?

```
void fun (int n)
{
    int j= 1, i=0;
    while( i <n)
    {
        i=i+j;
        j++;
    }
}
```

Ans.1. Now,

for $j = 1$, $i = 1$;
$j = 2$, $i = i + 2$
$j = 3$, $i = 1 + 2 + 3$
$j = 4$, $i = 1 + 2 + 3 + 4$.
. . .
. . .

} n terms

As $1 + 2 + 3 + 4 + \ldots \propto n$

$1 + 2 + 3 + \ldots + m \propto n$

$\frac{m(m+1)}{2} \propto n$

$m \approx \sqrt{n}$

Using Summation method

$$\sum_{i=1}^{n} 1 = 1 + 1 + 1 + \ldots + \sqrt{n} \text{ times}$$

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

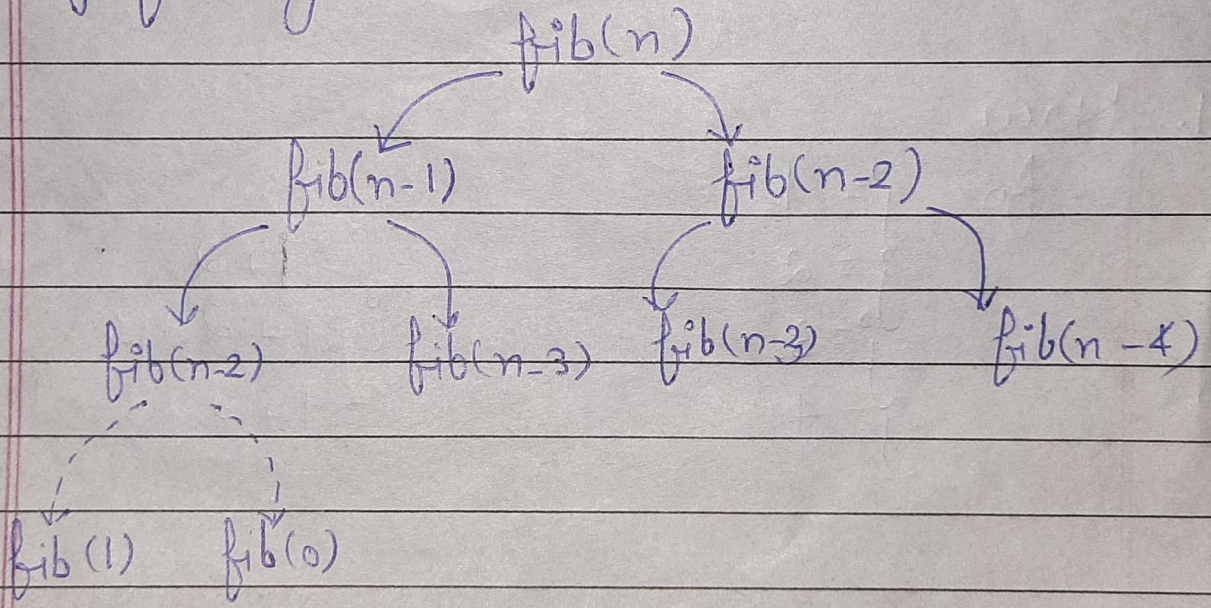
Ans 2) Fibonacci series —

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

$fib(0) = 0$

$fib(1) = 1$

by forming tree —



As at every fun call, we get 2 function calls.

So, for n levels —

we have — $2 \times 2 \times 2 \times \ldots \times n$ times

$T(n) = 2^n$

Maximum space—

considering recursive stack—

no. of calls maximum = n.

for each call we have space complexity o(1)

$\therefore \boxed{T(n) = O(n)}$

Without considering recursive stack—

at each call we have time complexity O(1).

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

Ans 3. i- n log n

Quick Sort

```
void quickSort(int a[], int l, int h)
{
    if(l < h)
    {
        int pi = partition(a, l, h);
        quickSort(a, l, pi-1);
        quickSort(a, pi+1, h);
    }
}
int partition(int a[], int l, int h)
{
    int pivotElement = a[h];
    int i = (l-1);

for(int j = l; j <= h-1; j++)
```

```
{
    if (a[i] < pivotElement)
    {
        i++;
        swap(&arr[i], &arr[j]);
    }
}
swap(&arr[i+1], &arr[h]);
return (i+1);
}
```

2- n^3

Ex-

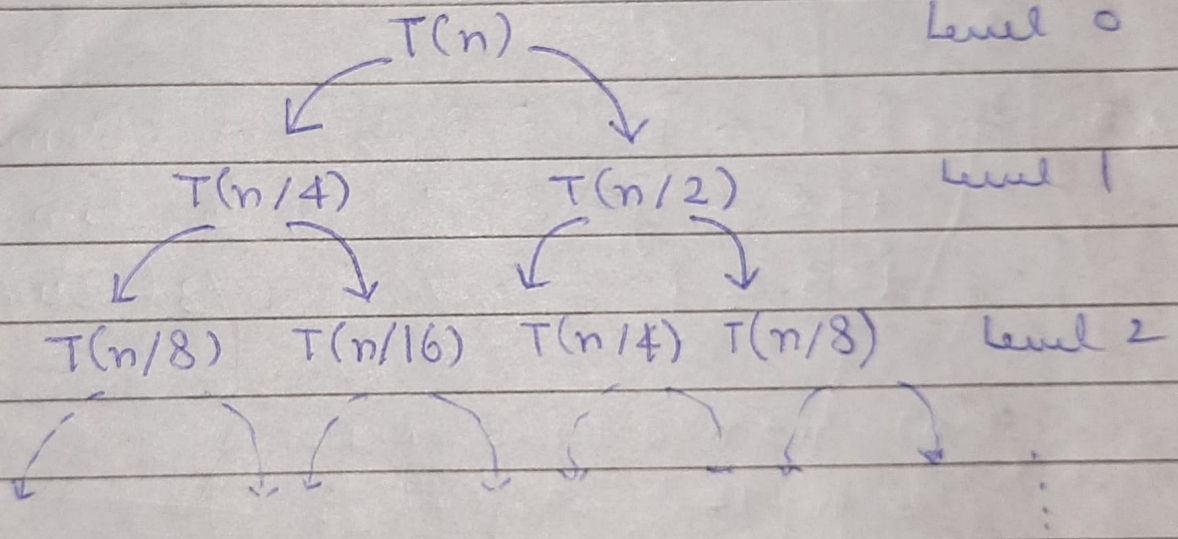
```
for(i=0; i<n; i++)
{
    for(j=0; j<c2; j++)
    {
        for(k=0; k<ci; k++)
        {
            res[i][j] += a[i][k] *
                         b[k][j];
        }
    }
}
```

3- $\log(\log n)$

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

Ans. 4. Given that –

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



At diff. levels –

$0 \rightarrow Cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{C5n^2}{16}$

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

⋮

max levels $= \frac{n}{2^k} = 1$

$\Rightarrow k = \log_2 n$

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

$= \left((5/16)^{\log n} n^2\right)$

$T(n) = Cn^2\left(1 + \left(\frac{5}{16}\right) + \left(5/16\right)^2 + \ldots + \left(5/16\right)^{\log n}\right)$

$T(n) = Cn^2 \times 1 \times \left(\frac{1 - (5/16)^{\log n}}{1 - 5/16}\right)$

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

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

Ans 5. for $i = 1 \rightarrow j = 1, 2, 3, 4, \ldots n$ (sum for n times)

$i = 2 \rightarrow j = 1, 3, 5, 7, \ldots n/2$ (sum for $n/2$ times)

$i = 3 \rightarrow j = 1, 4, 7, \ldots n/3$ (sum for $n/3$ times).

$T(n) = n + n/2 + n/3 + n/4 + \ldots$

$= n\left(1 + 1/2 + 1/3 + 1/4 + \ldots\right)$

$= n \int_1^n 1/x = n \int_1^n \frac{dx}{x} = [\log x]_1^n$

$= n \log n$.

Hence, the time complexity of given function is $n \log n$.

Date: / /

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

```
{
    O(1)
}
```

for i

2^1

2^k

2^{k^2}

2^{k^3}

$\vdots$

2^{k^m}

, where $2^{k^m} <= n$.

$k^m = \log_2 n$

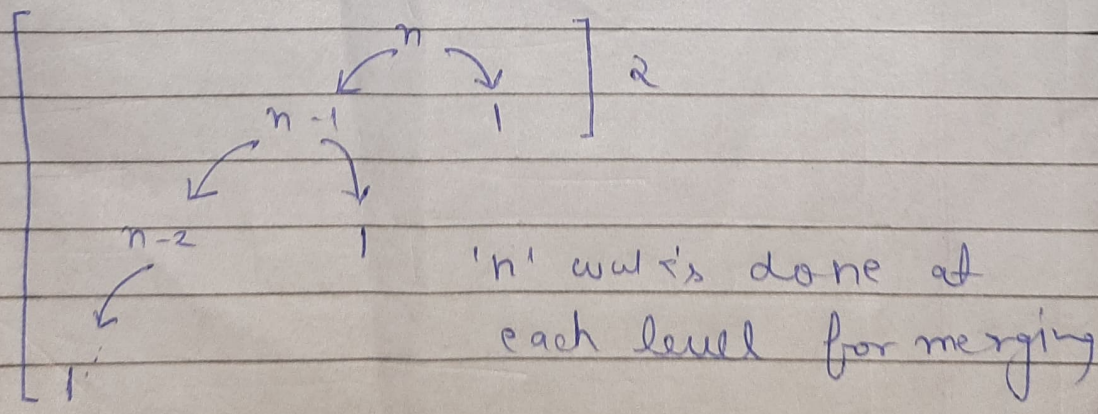
$m = \log_k \log_2 n$

$\because \sum_{i=1}^{m} 1 = 1 + 1 + 1 + \ldots + m \text{ times}$

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

Ans.7. The given algorithm divides array in eq.1 & 1/. part.1

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



'n' work is done at each level for merging

$T(n) = \left(T(n-1) + T(n-2) + \ldots + T(1) + O(1)\right) \times n$

$= n \times n$

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

Lowest height = 2

Highest height = n

$[\because \text{diff} = n-2], \quad n \geq 1$

given algorithm ~~proves~~ produces linear result.

Ans. 8. Considering for large values of ~~'n'~~ 'n':

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

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

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