TUTORIAL - 2 (DAA)

Geetansh Verma
2017494
CST - 41

Q 1.) Time complexity of code ::

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

Ans. In the iterations of while loop.

1st iteration , $i = 1$
2nd " , $i = 1+2$
3rd " , $i = 1+2+3$
4th " , $i = 1+2+3+4$

$\therefore$ so, for i times, $i = (1+2+3+4+ \cdots + i)$

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

now $i < n$, to exist.

so, $\Rightarrow \frac{i^2+i}{2} < n \Rightarrow i^2 < n$

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

So, $\boxed{\text{complexity} = O(\sqrt{n})}$ **Ans**

Q-2.) write recursive relation for recursive fibonacci series function. Solve the recurrence relation to get time complexity. what will be space compl. & why?

Sol^n recursive func. can be written as:-

```
int fibo(int n)
{   if n<=1;
        return n;          —①
    else
        return fibo(n-1) + fibo(n-2);    —②
}
```

Gv

from ① and ②

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

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

So, $T(n) = 2T(n-1) + n$ — ③

Now in ③ $n = \frac{n}{2}$ substituting.

$T(n) = 4T(n-2) + 3$ — ④

Same $\Rightarrow T(n) = 8T(n-3) + 7$ — ⑤

So, common eq for $k \Rightarrow T(n) = 2^k T(n-k) + (2^k - 1)$

$n - k = 0 \Rightarrow k = n$

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

$= 2^n - 1$ (discarding $T(0)$ since its of lower order)

$\boxed{T(n) = O(2^n)}$ Ans

Space Complexity:- It is given by:-

① no of stack frames × memory per stack frame

② the order of max depth of the binary tree for the func^n

Thus for recursive fibonacci func. $\boxed{\text{space compl.} = O(n)}$ Ans

Q-3) Write programs which have following complexities:

① $n(\log n)$

```
void fun(int n) {
    for(int i=1; i<=n; i++) {
        for(int j=1; j<n; j+=i) {
            ---
            ---
        }
    }
}
```

CS

② n^3:

```
void fun(int n){
    for(int i=0; i<n; i++){
        for(int j=i; j<n; j++){
            for(int k=j; k<n; k++){
                ---
            }
        }
    }
}
```

③ $\log(\log n)$

```
void fun(int n) {
    for(int i=2; i<=n; i*=i)
    { ---
    }
}
```

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

Ans we can assume that $T(n/4) \le T(n/2)$

$\therefore T(n) = 2T(n/2) + cn^2$

$\Rightarrow$ applying MASTERS METHOD

$a=2, b=2, c=\log_2 2 = 1$

$n^c = n^1 = n$

comparing n with $f(n)$

$\therefore n < n^2$

$\boxed{\text{complexity} = O(n^2)}$ Ans

Q-5) Time complexity =?

```
int fun(int n){
    for(int i=1; i<=n; i++) {
        for(int j=1; j<=n; j+=i) { ——— ①
            ---
        }
    }
}
```

Sol^n for $i = 1$

j (inner loop) $= 1 + 2 + 3 + \ldots$ n times

$i = 2$

$j = 1 + 3 - 5 - 7 \ldots \frac{n}{2}$ times

$i = 3$

$j = 1 - 4 - 7 - 10 - \frac{n}{3}$ times

$i = n$

$j =$ n times

$\therefore$ so, $\rightarrow$ $\Sigma = \{n + \frac{n}{2} + \frac{n}{3} + \ldots n\}$

$= n(1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{n})$

$= n(\log n)$

so, $\boxed{\text{Time complexity} = O(n \log n)}$ Ans

Q-6.) Time complexity = ?

```
for(int i=2; i<=n; i=pow(i,k))
{ ---
}
```

k => constant

Sol^n

1st iteration : $i = 2^1$

2nd " : $i = 2^k$

3rd " : $i = (2^k)^k = 2^{k^2}$

so, for nth " : $i = 2^{k^{\log_k(\log(n))}} = 2^{\log(n)} = n$

so, total iterations $\log_k(\log(n))$

$\boxed{\text{so, time complexity} = O(\log \log n)}$ Ans

Q-8) Arrange the time complexities in ↑sing order :-

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

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

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

<u>Ans</u> → $1 < \log\log n < \sqrt{\log n} < \log n < \log(2n) < 2\log(n) < n < n\log(n) < 2n < 4n < \log(n!) < n^2 < n! < 2^{2^n}$.

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

<u>Ans</u> → $96 < \log_8 n < \log_2 n < 5n < n\log_8(n) < n\log_2(n) < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$.