Tutorial-2

Name:- Harshit Mehra
Section:- CST.
Semester:- IV
Roll no:- 31

Design & Analysis of Algorithms

Tutorial - 2

Ques-1. What is the time complexity of below code?

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

Sol

j	i
1	0
1	1
2	3
3	6
4	10
5	15

$s = 0+1+3+6+10+15+ \dots T_k$ — ①

also $s = 0+1+3+6+10+15+ \dots T_{k-1} + T_k$ — ②

from 1 - 2

$\Rightarrow 0 = 1+2+3+4+ \dots + k - T_k$

$\Rightarrow T_k = 1+2+3+4+ \dots + k$

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

$\Rightarrow$ for k iteration

$1+2+3+6+ \dots k < n$.

$$\Rightarrow \frac{k(k+1)}{2} \leq n$$

$$\Rightarrow \frac{k^2+k}{2} < n.$$

$$\Rightarrow \sqrt{\frac{k^2}{2}+\frac{k}{2}} < \sqrt{n}$$

$$\Rightarrow k \cong O(\sqrt{n})$$

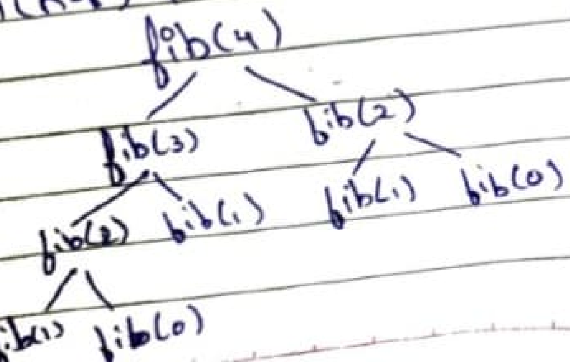
$$\Rightarrow T(n) \leq O(\sqrt{n})$$

Q-2 Write recurrence relation for recursive fn that prints fibonacci series. Solve the recurrence relation to get time complexity of the program. what will be the space complexity of this program and why?

0 1 1 2 3 5 . . . . n.

```
int fib(int n)
{ if(n<=1)                        — O(1)
    return n;
  return fib(n-1) + fib(n-2)      — T(n-1) + T(n-2)
}
```

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



— 1

— 2

— 4

— 8

$T(n) = 1+2+4+8+ \cdots +n.$

$$S(n) = \frac{a(r^{terms}-1)}{r-1}$$

$$\Rightarrow \frac{1(2^{n+1}-1)}{1}$$

$$\therefore T(n) = [2^n \cdot 2 - 1]$$

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

Space Complexity $O(1)$.

as recursive implementation doesn't store any values for and calculates every value from scratch.
so as complexity of 1 call is $O(1)$.
$\therefore$ total space complexity $= O(1)$

Ques-3 Program which have complexity:-

1) $n(\log n)$.

```
for(i=1; i<=n; i=i*2)        // log n times
   for(j=1; j<=n; j++)       // n times
   { int s=1;
   }
```

$\Rightarrow O(n\log n)$

x) n^3

```
for(i=0; i<=n; ++i)
  for(j=0; j<=n; ++j)
    for(k=0; k<=n; ++k)
      { cout<<"Hi"; }
```

n times
n times
n times

$\Rightarrow O(n^3)$

xi) log(log n)

```
for(int i=2; i<n; i=pow(i,2))
  { cout<<"Hi"; }
```

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

by neglecting lower order term $T(n/4)$.

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

$a = 1, \quad b = 2,$

$\Rightarrow c = \log_2 1$

$= 0$

~~for~~ $n^c = n^0 = 1 < cn^2$

$\therefore \Rightarrow T(n) = \Theta(n^2)$.

Q-5
```
int fn(int n)
{ for(int i=1 ; i<=n ; i++)
    for(int j=1; j<n; j+=i)
    { cout<<"hi";
    }
}
```

for $i=1$, $j = 1+2+3+4+5+6 \cdots n$

for $i=2$; $j = \cancel{2+4+6+8+ \cdots n}$ $1,3,5,7,\cdots n$

for $i=3$ ~~&~~, $j = 1,4,7,\cdots n$.

$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots,$

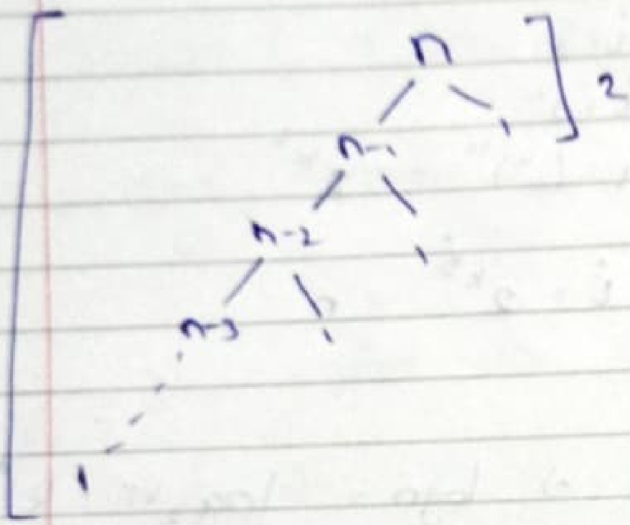
$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n}\right)$

$= n\int_1^n 1/x$

$= O(n \cdot \log n)$

Ques-7 given algo divides array in 99:1 to 1:1.

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



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

$T(n) = n^2$

lowest height $= 2$

highest height $= n$.

$\therefore$ difference $= n-2$

$\Rightarrow n > 1$

the given algo provides linear result

DOMS

Q-6 & Time complexity :-> for (int i = 2 ; i <= n ; i = pow(i,k)) { }

where k is constant

for first iteration $i = 2$.
second $i = 2^k$
third $i = (2^k)^k = 2^{k^2}$
:
nth iteration $i = 2^{k^i} = n$

$n = 2^{k^i}$

applying log => $\log n = \log_2 2^{k^i} = k^i$
applying log again
$i => \log_k \log(n)$

$=> T(n) = \log_k \log_2(n)$

Q-8 a) $n, n!, \log n, \log\log n, \text{root}(n), \log(n!), n\log n, \log^2(n), 2^n, 2^{2^n}$
$4^n, n^2, 100$

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

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