PAGE
DATE

Name Ananya Jhaldyal
Class CST-SPL-1
Roll no 54.

TUTORIAL-2

1. What is the time complexity of given code?

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

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

$S = 0+1+3+6+10+15+ \dots\dots T_k$ ——— (1)

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

(1) − (2)

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

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

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

for k iterations

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

~~$T(n) = 1+2+4+8+ \dots\dots k+n$~~

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

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

taking roots both sides

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

$$k \simeq (\sqrt{n})$$

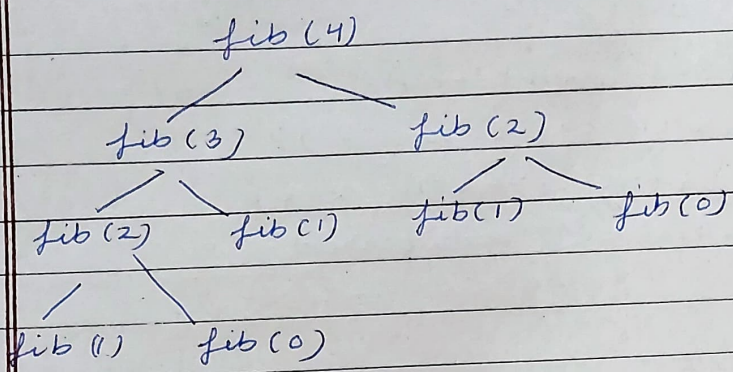
$$T(n) = O(\sqrt{n})$$

2. Write recurrance relation for recursive function that prints fibonacci series. Solve recurrance relation to get time complexity of the program what will be the space complexity of this program & why?

Fibonacci series 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)
}
```

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



Recursive Tree

Total no. of recursive calls for n terms

$1+2+4+8+\dots$

$a=1,\ r=2$

$$\frac{a\,(r^{\text{terms}}-1)}{r-1}$$

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

$2^n \cdot 2 \Rightarrow O(2^n)$

Space complexity for the program will be $O(1)$

As recursive, As in recursive approach, it doesn't store any values and for every value it has to start from beginning so complexity of call is $O(1)$

3. Write programs which will have complexity

1.) $n(\log n)$

```
for (i = 1; i <= n; i *= 2)
    for (j = 1; j <= n; j++)
        sum = sum + j;
```

i	j
1	1
2	2
4	3
⋮	⋮
$\log n$	n

$\Rightarrow O(\log n * n)$

$O(n \log n)$

ii) $O(n^3)$

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

i	j	k
0	0	0
1	1	1
2	2	2
3	3	3
...	...	...
n	n	n

$\Rightarrow O(n * n * n)$
$= O(n^3)$

iii) log (log n)

```
for (int i = 2; i < n; j = pow(i, 10))
 {
 s = s+1;
 }
```

$\Rightarrow O(\log(\log n))$

~~what is this~~

q-4) Solve the following recurrance relation

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

neglecting $T(n/4)$ as it is lower order term

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

$a = 1, b = 2$

$c = \log_2 1$

$= 0$

$n^0 = n^0 = 1 < cn^2$

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

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 ..... n
for i = 2 j = 1, 3, 5, 7 ... n
for i = 3 j = 1, 4, 7 ... n

$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \dots 1$

$n\left(1 + \frac{1}{2} + \frac{1}{4} + \dots \frac{1}{n}\right)$

$n\int_{1}^{n} 1/n \Rightarrow O(n \cdot \log n)$

6. Time complexity - for(int i=2; i<=n; i=pow(i,k))
{ }
where k is a constant

First iteration $i = 2$
Second iteration $i = 2^k$
Third iteration $i = (2^k)^k = 2^{k^2}$

n^{th} iteration $i = 2^{k^t} = n$
$n = 2^{k^t}$

taking $\log_2$ both sides
$\log n = \log_2 2^{k^t}$
$\log n = k^t \log_2 2$
$\log n = k^t$

Taking log with base k

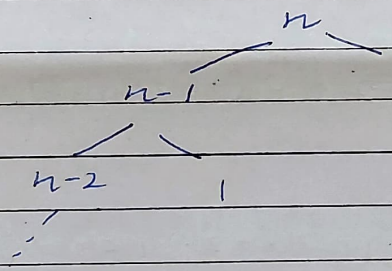
$\log_k \log^n = i$

$T(n) = \log_k \log(n)$

7) Write a recurrance relation when quick sort repeatedly divides the array in two parts of 99% & 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity and find difference in heights of both the extreme parts. What do you understand by this analysis

99% & 1% array part

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



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

$= n \times n$

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

Lowest ~~order~~ height = 2

highest height = n

$\therefore$ diff = $n-2 \quad n > 1$

By analysing the problem we can conclude that the given algorithm provides linear result.

8) Arrange the following in increasing order of given growth rate

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

b) $1 < \log\cdot\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < m < \log(n!) < n^2 < n! < 2^{2^n}$

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