Name: T Rishita Reddy
Roll no: 47
Sec: I

Tutorial - 2

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

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

Soln:

$i = \overbrace{0, 1, 3, 16, 10, 15, 21, \ldots}^{k\text{-terms}} n$

Let the sum of above k terms is S_k

$S_k = 1+3+6+10+15+21, \ldots\ldots + T_k$ — ①

$S_{k-1} = 1+3+6+10+15+21+\ldots\ldots + T_{k-1}$ — ②

Subtracting ② from ①

$T_k = S_k - S_{k-1} = 1+2+3+4+5+6+\ldots\ldots+k$

we have $T_k = n$

$\therefore 1+2+3+4+5+\ldots\ldots+k = n$

$\frac{k(k+1)}{2} = n \Rightarrow k^2+k-2n = 0$

$\Rightarrow k = \frac{-1 \pm \sqrt{8n+1}}{2}$

Taking only positive value we get total no. of times the loop runs for $i = k+1 = \frac{\sqrt{8n+1}}{2}$

$\therefore$ TC, $T(n) = O\left(\frac{\sqrt{8n+1}}{2}\right) = O(\sqrt{n})$

Q.2

Ans→ Recursive function:

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

Recurrance Relation, $T(n) = T(n-1) + T(n-2) + c$

$$T(n-1) \simeq T(n-2)$$

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

$$T(n-2) = 2^* (2T(n-2-2) + c) + c$$

$$= 4T(n-2) + 3c$$

$$T(n-4) = 2^* (4T(n-2) + 3c) + c$$

$$= 8T(n-3) + 7c$$

Generalising $= 2^k T(n-k) + (2^k - 1)c$

Put $n-k=0$

$n=k$

Put $n=k$

$$T(n) = 2^n * T(0) + (2^n - 1)c$$

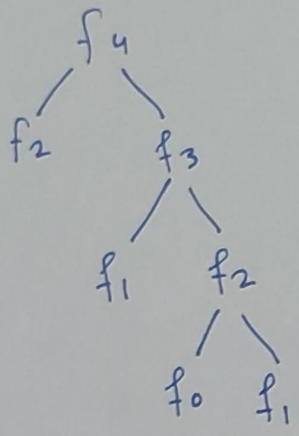
$$= 2^n * 1 + 2^n c - c$$

$$= 2^n (1+c) - c$$

$$= 2^n$$

Time Complexity $= O(2^n)$

Space Complexity: Space is proportional to the maximum depth of the recursion tree.



Hence space complexity of fibonacci recursion is $O(N)$

Q.3 Write programs which have complexity.

Soln:

1. n(logn)

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

2. n^3

```
for (i=0 ; i<n ; i++)
 {  for (j=0 ; j<n; j++)
   {  for(k=0; K < n; k++)
      {
        Sum = Sum + k;
      }
   }
}
```

3. logn(logn)

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

Q.4 Solve the Recurrance Relation $T(n) = T\left(\frac{n}{4}\right) = T\left(\frac{n}{2}\right) + Cn^2$

Soln :→ $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + Cn^2$

$\therefore T\left(\frac{n}{4}\right) \simeq T\left(\frac{n}{2}\right)$

$\Rightarrow T(n) = 2T\left(\frac{n}{2}\right) + Cn^2$

As $a \geq 1$ & $b \geq 1$

$\therefore$ using Master's Method.

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$$c = \log_b a$$

$$c = \log_2 2 = 1$$

$$f(n) > n^c$$

$$\therefore T(n) = O(f(n))$$

$$= O(n^2)$$

Q.5

Soln : for $i=1$, j is $1, 2, 3, 4 \ldots\ldots$ run for n-times

for $i=2$, j is $1, 3, 5 \ldots\ldots$ upto $n/2$ times

for $i=3$, j is $1, 4, 7 \ldots\ldots$ run for $n/3$ times.

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

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

$= n\int_1^n dx/x$

$= [\log x]_1^n$

$\Rightarrow TC = n\log n$

Q.6

Ans⇒ for first iteration $i = 2$

Second iteration $i = 2^k$

third iteration $i = (2^k)^k = 2^{k^2}$

⋮

n^{th} iteration $i = 2^k$ loop ends at $2^i = n$

apply $\log n = \log_2 k^i$

$k^i = \log n$

$i = \log_c(\log n)$

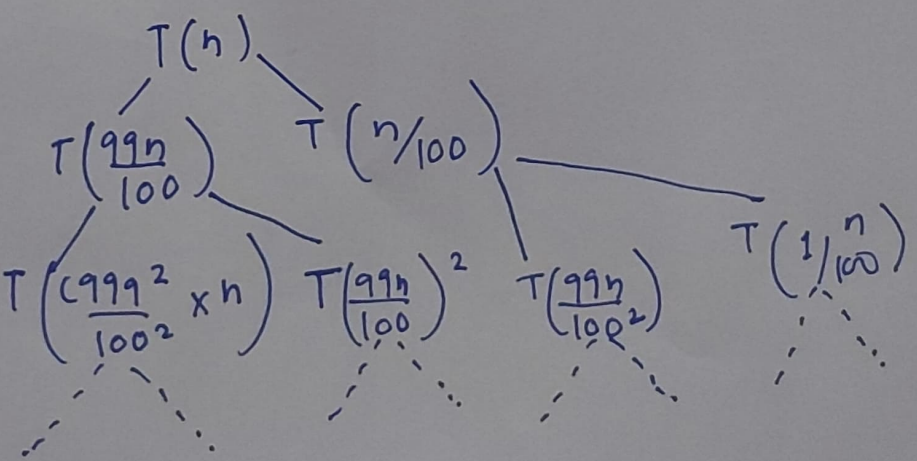
Q.7

Ans⇒ 99 to 1 in quick sort when pivot is where from front or end always.

So,

$T(n) = T(99n/100) + T(n^1/100) + O(n)$

$T(n) = T(99n/100) + T(n/100) + O(n)$



$n = (99/100)^K$

$\log n = K \log 99/100$

$K = \log_n \frac{100}{99}$

$\therefore T.C = n * \log_{\frac{100}{99}}(n)$

Q8

Ans=

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

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