TUTORIAL -2

Q1 What is the time complexity of below

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

Ans1 i=0, 1, 3, 6, 10 let say K times

So given form would be

$$\frac{K(K+1)}{2}$$

K^{th} term $h = \frac{K(K+1)}{2} = n$

$$K^2 + K = 2n$$

$$K^2 = n$$

$$K = \sqrt{n}$$

Time complexity $h = O(\sqrt{n})$

Q2 Write recurrence relation for the recursion function that prints fibonacci series. Solve the recurrence relation to get the time complexity of this program and why.

Ans2

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

Recurrence relation $T(n) = T(n-1) + T(n-2) + c$

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

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

By backward substitution

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

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

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

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

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

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

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

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

generalizing: $2^k T(n-k) + (2^k - 1)c$

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

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

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

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

$= 2^n$

Time complexity $= O(2^n)$

Space complexity:

For fibonaci space required is directly $\propto$ to maximum depth of recursion tree.

Since max. depth is dic to no of elements.

Q3 Write a program which have complexity $n(\log n)$

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

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 $\log n(\log n)$

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

Q4 Solve the recurrence relation

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

Ans4 $T(n/4) = T(n/2)$

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

$a \geq 1$ and $b \geq 1$

By using master's method

$T(n) = T(n/b) + f(n)$

$L = \log_2 a \Rightarrow 1$

$f(n) > n^c \Rightarrow cn^2 > n^c$

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

$O(n^2)$

Q5 What is the time complexity of following from ().

```
int fun (int n)
    for (int i=1; i<=n; i++) {
        for (int j=1; j<n; j+=6) {
            Some O(1) task
        }
    }
```

Ans5 for $i=1 \Rightarrow 1+2+3+\cdots(n+1)=n$

for $i=2 \Rightarrow 1+3+5+\cdots n \Rightarrow n/2$

for $i=3 \Rightarrow 1+4+7+\cdots n \Rightarrow n/3$

$$n+\frac{n}{2}+\frac{n}{3}+\cdots+)$$

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

Now we know

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

$$n\left(1+\frac{1}{2}+\frac{1}{3}+\cdots\frac{1}{n}\right) \leq n(1+0.5\cdots)$$

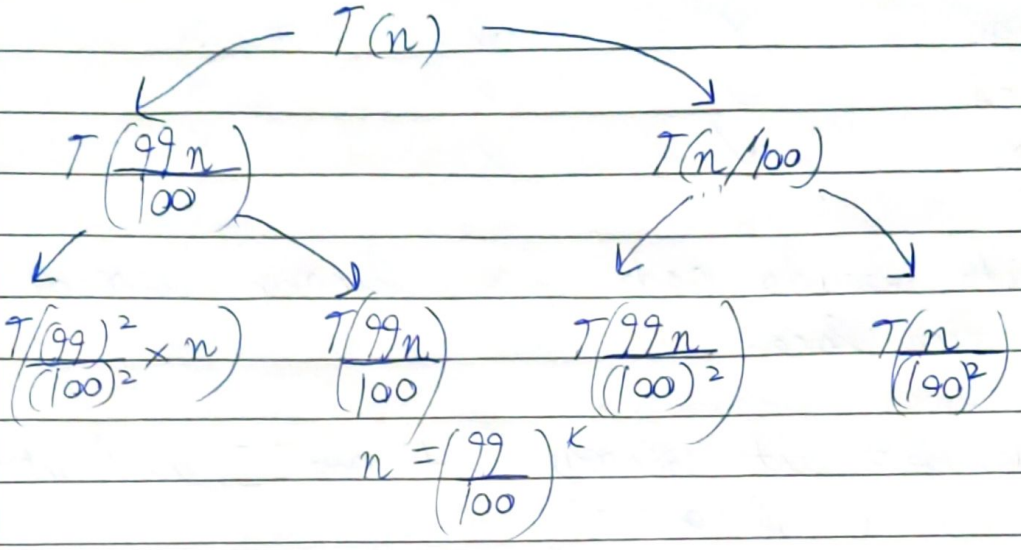
$O(n \log n)$

Q7 Write a recurrence relation color quick sort repeatedly divides the array to show the recurrence the time complexity the difference. in light of both the extreme cases what do you understand by the analysis.

Ans7 49 to 1 in quick sort

when pivot is close from front or end always.

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



$$n = \left(\frac{99}{100}\right)^k$$

$$\log n = k \log \frac{99}{100}$$

$$K = \log n \frac{100}{99}$$

Time complexity = $n \log$

Q8 Arrange the following in increasing order of rate of growth

Ans 8 $n!, n!, \log n, \log \log n, \sqrt{n}, \log(n!)$
$n \log n, 2^n, 2^{2n}, 4^n, n^2, 100$

Ans 8 $100 < \log(\log n) < \log n < \log^2 n < \sqrt{n} < n < \log n!$
$< n \log n < \log^{2n} < n^2 < 2^n < 4^4 < 2^{2n} < n!$