Q1.) What is the time complexity of the below code.

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

Ans

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

$\therefore$ 1st time $i=1$

2nd time $i=3$ $(i=1+2)$

3rd time $i=6$ $(i=1+2+3)$

$\vdots$

n^{th} time $i = \dfrac{k(k+1)}{\alpha} = n^2 < n$

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

Q2.) Write recurrence relation for the recursive function that prints fibonacci series. Solve the recurrence relation to get complexity of the program. What will be the space complexity of this program & why?

Ans

$*$ $fib(n) = fib(n-1) + fib(n-2)$

```
fib(n)
{ if n<=1
    return 1;
```

return $(fib(n-1) + fib(n-2))$;

Time complexity :

Let $T(0) = 1$

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

$= 2^*(n-2) + c$ (Let $T(n-1) \approx T(n-2)$)

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

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

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

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

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

$= 2^k \times T(n-k) + (2^k - 1)c$
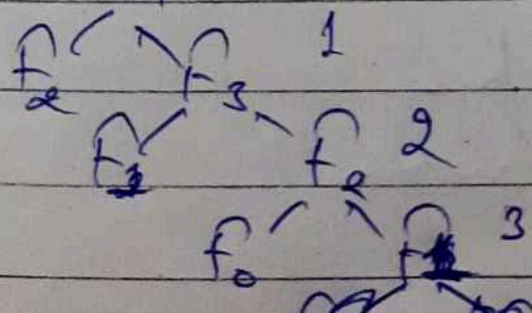
⇒ $n - k = 0$ 2) $n = k$

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

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

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

$\approx 2^n$ // Constants can be ignored

Space complexity : The space is proportional to the maximum depth of the recursion tree.

for e.g : for f_4

∴ space complexity of fibonacci series = $O(N)$.

(Q3.) Write programs which have complexity - $n(\log n)$, $n^2 \log(\log n)$

Ans

Merge sort - $n \log n$

• For time complexity = n^3.

We can use three nested loops - $O(n^3)$

```
for (int i=0; i<n; i++)
{
    for (int j=0; j<n; j++)
    {
        for (int k=0; k<n; k++)
        {
            // Some O(1) expressions
        }
    }
}
```

⇒ For time complexity = $\log(\log n)$

We can use the following function

```
for (int i=2; i<n; i=pow(i,k))
{
    // some O(1) expression
}
```

where k is constant

⇒ For time complexity = $n \log n$

We can use the following function

```
int fun (int n)
{
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j+=i)
        // some O(1) expressions } } }
```

(Q4.)

Ans

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

Using master's method $T(n) = aT(n/b) + f(n)$

$a \geq 1, \ b > 1, \ c = \log_b a$ (comparing n^c & $f(n)$)

We get $c = \log_2 2 = 1$

$\therefore f(n) > n^c \quad \therefore n^2 > 1$

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

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

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

```
int fun (int n)
{
    for (int i=1; i <= n; i++)
    {
        for (int j=1; j < i; j += i)
        {
            // some O(1) expression
        }
    }
}
```

Ans

for $i = 1 \rightarrow j = 1, 2, 3, 4, \dots n$ (sum for n terms)

for $i = 2 \rightarrow j = 1, 3, 5, 7 \dots$ (sum for $n/2$ terms)

for $i = 3 \rightarrow j = 1, 4, 7, \dots$ (sum for $n/3$ terms)

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

$$= n(1 + 1/2 + 1/3 + 1/4 + \dots)$$

$$= n \int_1^n 1/n = n \int_1^n dn/n = \log n \int_1^n dn = n \log n$$

$\therefore$ The time complexity is $n \log n$

Q6.) What should be the time complexity of following func...

```
for(int i=2; i<a; i=pow(i,k)){
{
    // Some O(1) expressions or statement
}
```

where k is a constant.

Ans for first iteration $i=2$

2nd iteration $i=2^k$

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

$\vdots$

n^{th} iteration $i=(2^k)$. Info, ends it $u = 2^{k}$

$\therefore$ applying $\log$, $\log_k = \log 2^{k} \Rightarrow \log_n = k^i b$

Applying $\log$ again, $\log(k^i) = \log()_n \Rightarrow i = \log_y(\log_2)$

Q7.) Write a recurrence relation when quick sort repeatedly divides the array into two parts of 99% & 1%. Derive the time complexity & find the difference in heights of both the extreme parts. What do you understand by "spill"?

Ans

$\frac{1}{10}n \quad \frac{9}{10}n \qquad \to n$

$\frac{1}{100}n \quad \frac{9}{100}n \quad \frac{9n}{100} \quad \frac{81n}{100} \quad \to n$

$\frac{81n}{1000} \quad \frac{729n}{100} \to n$
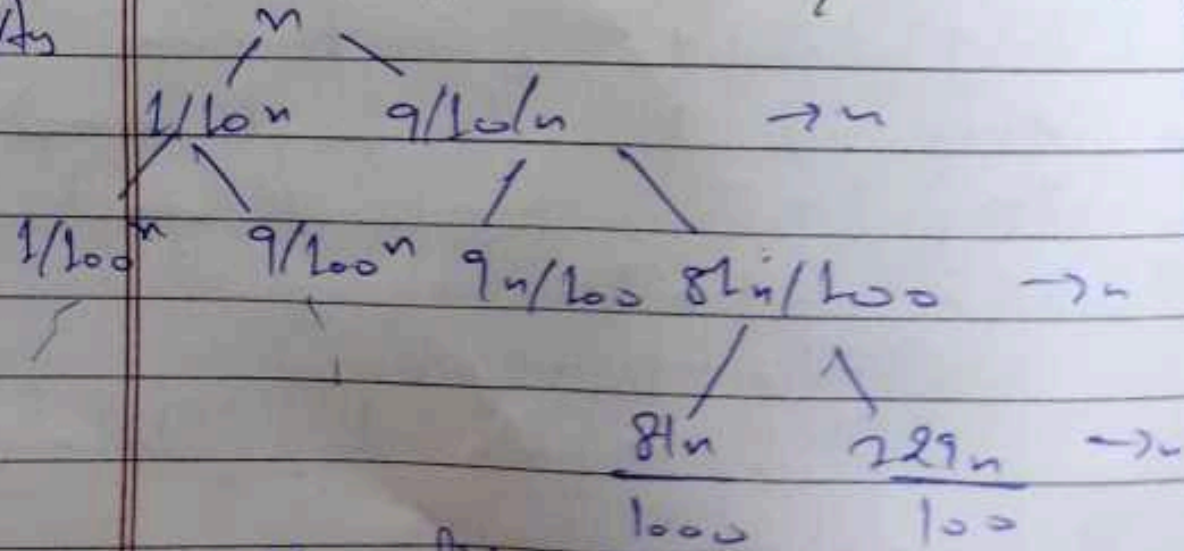
$\therefore$ If x split is the minimum

Recurrence relation: $T(n) = T(n/10) + T(n/10) + O(n)$

where first branch is of $9n/10$ & second is of $n/10$

Solving the above using recursion-tree approach calculating value

At 1st level, value = n

At 2nd level, value = $\dfrac{9n}{10} + \dfrac{n}{10} = n$

Values remains same at all levels of n.

Time complexity = Summation of values

$$= O(n \times \log_{10/9} n) \quad \text{(upper bound)}$$

$$= \Omega(n \log_{10} n) \quad \text{(lower bound)}$$

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

Q6) Arrange the following in increasing order of rate of growth:

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

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

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

A $1 < \log(\log n) < \sqrt{\log n} < \log n < \log^2 n < 2\log n < n < 2n <$
$4n < n\log n < n^2 < \log(n!) < n! < 2(2^n)$

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

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