TUTORIAL-2

Q.1) What is the time complexity of below code and how?

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

$j = 1$	$i = 1$	m-level
$j = 2$	$i = 1+2$	
$j = 3$	$i = 1+2+3$	

for (i)

$\because 1 + 2 + 3 + \dots < n$

$1 + 2 + 3 + m < n$

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

$m \approx \sqrt{n}$

By summation method,

$\sum_{k=1}^{m} 1 \Rightarrow 1 + 1 + \dots \sqrt{n}$ times

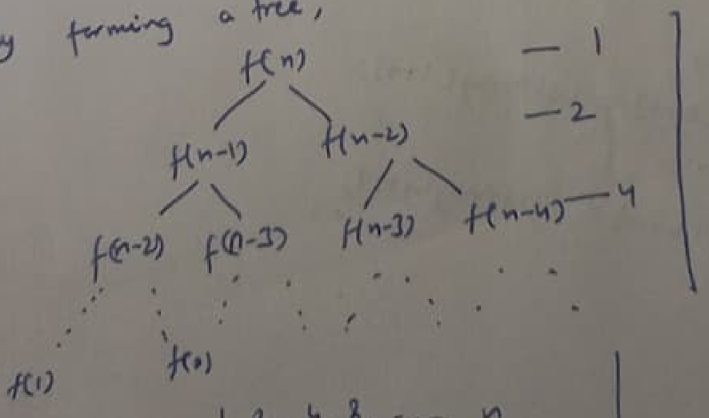
$T(n) = \sqrt{n}$

Q.2) What recurrence relation for ~~relation~~ function that prints Fib. series. Solve it to get the time complexity. What will be the space complexity and why?

For fibonacci series,

$f(n) = f(n-1) + f(n-2)$ $\quad f(0) = 0$, $f(1) = 1$

By forming a tree,



1, 2, 4, 8, ... n

$a = 1, r = 2$

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

$T.C. = O(2^n)$

n2

Space Complexity :-

Recursive:- $T(n) = O(n)$

Iterative: $T(n) = O(1)$

Q.3) Write programs which have time complexity:

$n \log n$, n^3, $\log(\log n)$

① $n\log n$ - ~~Q~~ Mergesort

```
void mergesort (int array[], const int low, const int high)
{
   if ( low >= high) return;
   int mid = low + (high-low)/2;
   mergesort (array, low, mid);
   mergesort (array, mid+1, high);
   merge (array, low, mid, high);
}

void merge ( int array[ ], const int low, int mid, int high)
{
   int i, j, k;
   int n1 = mid - low +1;
   int n2 = high - mid;
   int leftArray[n1] , rightArray[n2];
   for (~~int~~ i=0; i < n1; i++)
       leftArray[i] = array[low+i];
   for (~~int~~ j=0; j < n2; j++)
       rightArray[j] = array[j +mid +1];
   i=0, j=0, k= low;
   while ( i < n1 && j < n2)
   { if ( leftArray[i] <= rightArray [j])
          array[k] = leftArray [i];
     else
          array [k] = rightArray [j];
   }
   while (i<n1)
       array [k++] = leftArray [ i++];
   while (j<n2)
       array [k++] = rightArray [j++];
}
```

(ii) $\underline{n^3}$ - Multiplication of two matrices

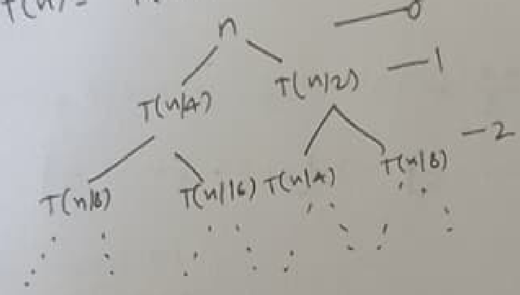
```
for(i = 0; i < n1; i++)
  for(j=0; j<n2; j++)
    for (k=0; k<n3; k++)
      res[i][j] += a[i][k] * b[k][j];
```

(iii) $\underline{\log(\log(n))}$ -

```
for(i = 2; i<n; i = i*i)
    count ++;
```

Q.4) Solve the following recurrence relation:-

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



At level,

$0 \rightarrow cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{c5n^2}{16}$

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 c$

$\vdots$

max level $= \frac{n}{2^k} = 1$

$= k = \log_2 n$

$$T(n) = c\left(n^2 + \frac{5}{16}n^2 + \frac{5}{16}^2 n^2 + \dots \left(\frac{5}{16}\right)^{\log_2 n} n^2\right)$$

$$T(n) = cn^2\left[1 + \frac{5}{16} + \frac{5}{16}^2 + \dots + \frac{5}{16}^{\log n}\right]$$

$$= cn^2 \times 1 \times \left[\frac{1 - (5/16)^{\log n}}{1 - 5/16}\right]$$

$$= cn^2 \times \frac{11}{5} \times \left[1 - \left(\frac{5}{16}\right)^{\log n}\right]$$

$T(n) = O(n^2 c)$

$O(cn^2)$

Q.5) What is the time complexity of following func() ?

```
int fun (int n)
  for ( int i = 1; i <= n; i ++)
     for (int j = 1; j < n; j += i)

          some O(1)
```

for	i	j
	1	1
	2	1+3+5 ~~[illegible]~~
	3	1 + 4+7 ~~[illegible]~~
	:	
	n	

$j = (n-1)/i$ times

$\sum_{i=1}^{n} \frac{n-1}{i}$

$\because T(n) = \frac{n-1}{1} + \frac{n-1}{2} + \frac{n-1}{3} + \cdots \frac{n+1}{n}$

$T(n) = n[1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n}] - 1\,[1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n}]$

$= n \log n - \log n$

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

Q.6) What should be time complexity of

```
for (int i=2; i<=n; i= pow (i,k))      k → const.
   // some O(1)
```

i

2^1

2^k

2^{k^2}

2^{k^3}

$\vdots$

2^{k^m}

where

$2^{k^m} <= n$

$k^m = \log_2 n$

$m = \log k \log_2 n$

$\therefore \sum_{i=1}^{m} 1 = 1 + 1 + 1 + \cdots$ m times

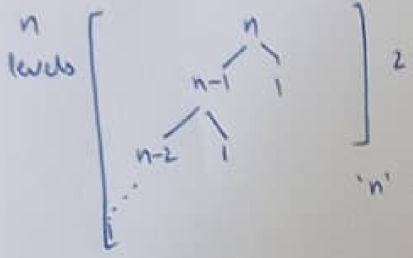
$T(n) = O(\log_k \log n)$

int fun (int n)

Q.7) Write a recurrence relation when quick sort repeatedly divide array into 2 parts of 99% and 1%. Derive time complexity in this case. Show the recurrence time while deriving time complexity and find difference in heights of both extreme parts. What do you understand by this analysis?

Given algorithm divides array in 99% and 1% part.

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



'n' work is done at each level

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

$= n \times n$

$\boxed{\therefore T(n) = n^2}$

Lowest height $= 2$

highest height $= n$

$\therefore \boxed{\text{difference} = n-2} \; n > 1$

The given algorithm produces linear result.

Q.8) Arrange following in increasing order of rate of growth:

a) $n, n!, \log n, \log\log n,$ ~~[illegible]~~ $\sqrt{n}, \log(n!),$ ~~[illegible]~~ $n\log n, \log^2(n),$ ~~[illegible]~~ $2^n,$
$2^{2^n}, 4^n, n^2, 100.$

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

'n!),

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

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

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

$96 < \log_2 n < \log 2n < 5n < n\log_6 n < n\log_2 n < \log(n!) < 8n^2 < 7n^3$
$< n! <$ ~~[illegible]~~ 8^{2n}