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

Time complexity → O(sqrt n).

1st time = i = 1
2nd time = i = 3 (i=1+2).
3rd time i = 6 (i=1+2+3).
:
nth time = i = $i\frac{(i+1)}{2} = x^2 < n$

$x = \text{sqrt}(n)$.

2.) Let T(0) = 1.

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

fib(n):
if n<=1
return 1
return fib(n-1) + fib(n-2).

Time complexity :-

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

$T(n-2) = 2*(T2(n-2-2)+C)+C$.
$= 2*(2T(n-2)+C)+C$
$= 4T(n-2)+3C$.

$T(n-4) = 2*(4T(n-2)+3C)+C$.
$= 8T(n-3)+7C$
$= 2^K \times T(n-K) + (2^K-1)C$

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

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

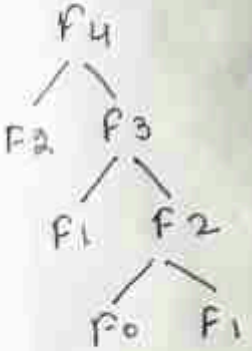
$= 2^n * 1 + 2^n c - c$

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

$= 2^n.$

$= O(2^n).$

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



Hence the space complexity of Fibonacci recursive is O(N).

Merge sort - $n \log n$.

for time complexity:- n^3

We can use three nested loops

```
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 - $\log(\log n)$.

```
for (int i = 2; i < n; i = power(i, j))
  { // some O(1) expression
  }.
```

where k is constant.

for time complexity $n \log n$

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

Q:-4.

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

using master's method

$T(n) = aT(n/b) + fn.$

$a \geq 1, b \geq 1, c = \log_b^a$

$c = \log_2^2 = 1$

$f(n) > n^c$

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

$\Rightarrow \Theta(n^2)$.

Q:-5

for $i = 1 \rightarrow j = 1, 2, 3, 4 - - - - n$ (runs for n times)

for $i = 2 \rightarrow j = 1, 3, 5 - - - - -$ (runs for $n/2$ times).

for $i = 3 \rightarrow j = 1, 4, 7 - - - - -$ (runs for $n/3$ times).

$T(n) = n + n/2 + n/3 + n/4 + - - - .$

$n(1 + 1/2 + 1/3 + 1/4 + - - -).$

$n \int_1^n 1/x \Rightarrow n \int_1^n dx/x \Rightarrow \log x]_1^n$

$T.C^1 = n \log n$

Q:-6.

for first iteration $i = 2$

second iteration $i = 2^K$

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

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

apply $\log n = \log 2^{K^i} = K^i = \log n \Rightarrow i = \log_2(\log n)$.

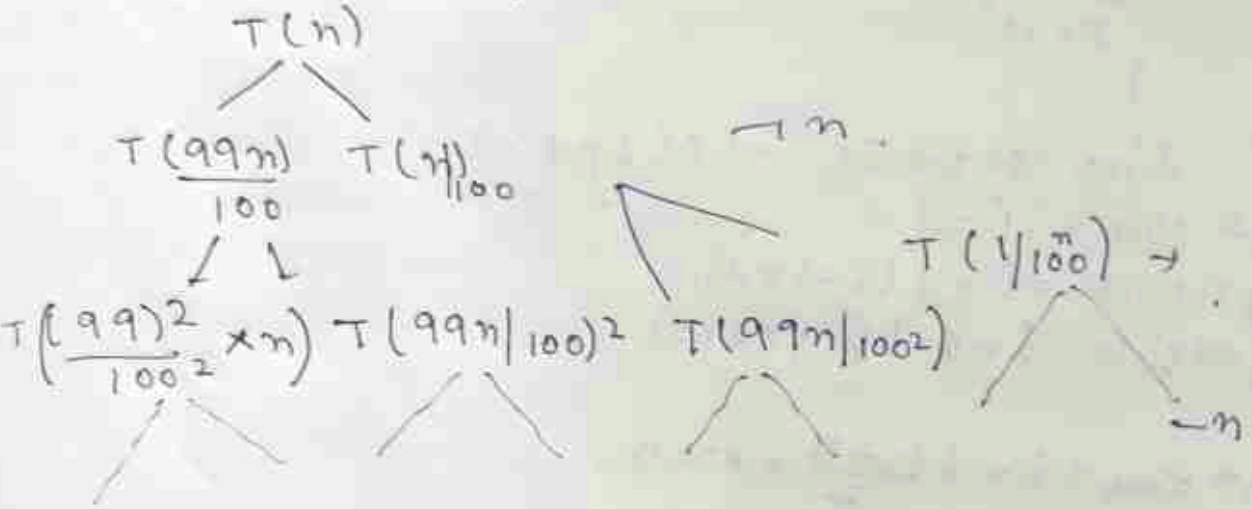
Q:-7.

99 to 1 in Quick Sort

When pivot is either from front or end always

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

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



$$\frac{n}{(99/100)^k} = 1.$$

$$n = (99/100)^k$$

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

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

$\therefore TC = n * \log_{100/99}(n)$.

Ques:- 8

a.) $100 < \log\log(b) < \log^2 n < \log n < \log n! < n < n \log n < n^2 < 2^n < +^n < 2^n (2^n) < n!$

b.)