Name - Saumya Negi
Section - I
Roll No - 38

classmate
Date
Page

TCS-409
Tutorial 2

Q1 void fun(int n)

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

$i = 0, 1, 3, 6, 10, 15, 21 --- n$

Let sum of K terms be S_K

$S_K = 1 + 3 + 6 + 10 + 15 + 21 + --- + T_K$ — (1)

$S_{K-1} = 1 + 3 + 6 + 10 + 15 + 21 + --- + T_{K-1}$ — (2)

Subtracting (2) from (1)

$T_K = S_K - S_{K-1} = 1 + 2 + 3 + 4 + 5 + 6 + --- + K$

we have $T_K = n$

$\therefore 1 + 2 + 3 + 4 + 5 + --- + K = n$

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

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

loop runs for $i = K+1 = \sqrt{8n+1}$

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

Q2 Recursive function:

int fib(int n)

{ if (n <= 1) $\rightarrow O(1) = C$

return n;

return fib(n-1) + f(n-2)

$\longrightarrow T(n-1) + T(n-2)$

}

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

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

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

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

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

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

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

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

Put $n-K = 0$

$n = K$

Put $n = K$

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

$= 2^n \cdot 1 + 2^n c - c$

$= 2^n$

Time complexity $= O(2^n)$

Q3 1) $n(\log n)$

```
for(i=1; i<n; i++)
{
  for(j=1; j<2n; 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) $\log n(\log n)$

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

Qu. 4: Solve the Recurrance Relation $T(n) = T(\frac{n}{4}) + T(n/2) + cn^2$

Sol^n

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + cn^2$$

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

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

As $a \geq 1$ and $b > 1$

$\therefore$ Using master's Method.

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

$$c = \log_b a$$

$$c = \log_2 2 = 1$$

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

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

$$= O(n^2)$$

Qu. 5: 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<n; j+=i)
            { some O(1) task
}}}
```

Solⁿ:

for $i = 1$, j is $1, 2, 3, 4$ - - - - - - - - run for n - times
for $i = 2$, j is $1, 3, 5,$ - - - - - - - - upto $n/2$ times
for $i = 3$, j is $1, 4, 7,$ - - - - - - - run for $n/3$ times

$$T(n) = n + n/2 + n/3 + n/4 + \cdots$$
$$n\left(1 + 1/2 + 1/3 + 1/4 + \cdots\right)$$
$$= n\int_1^n dx/x$$
$$= [\log x]_1^n$$

$\Rightarrow$ Time complexity $= n \log n$.

Qu-6: What should be the time complexity of

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

where k is a constant.

Solⁿ: for first iteration $i = 2$
second iteration $i = 2^k$
third iteration $i = (2^k)^k = 2^{k^2}$
|
|
mth iteration, $i = 2^k$ loop ends at $2^i = n$

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

$$k^i = \log n$$
$$i = \log_k(\log n)$$

Que-7 Write a recurrence relation when quick sort repeatedly divides the array in to two parts of 99% and 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity and find the difference in heights of both the extreme parts. What do you understand by this analysis.

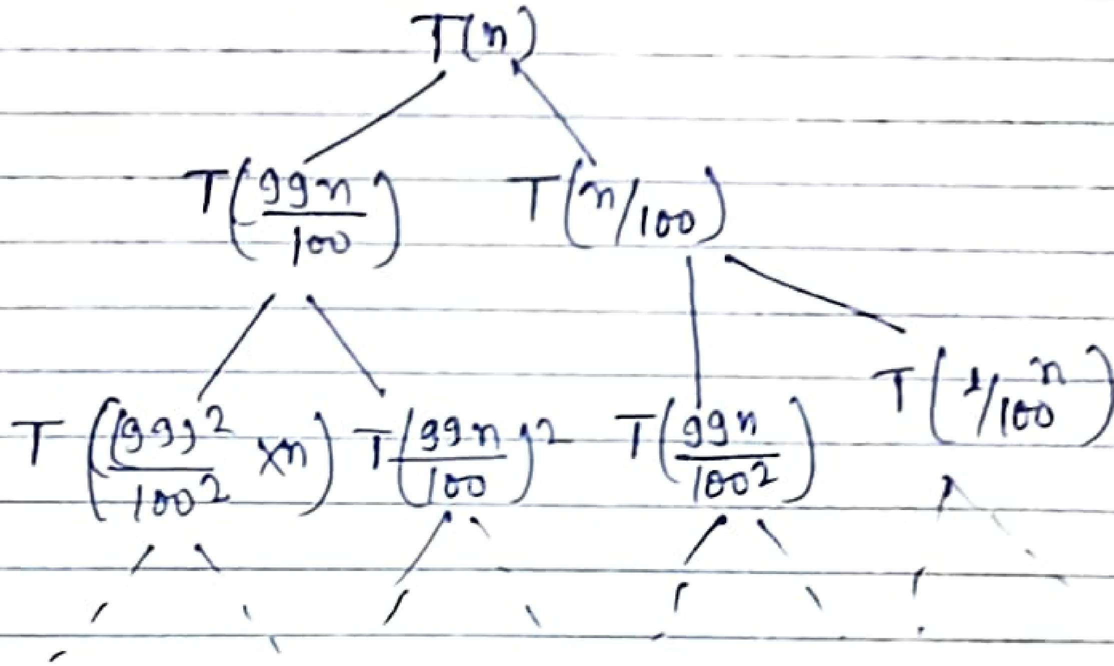
Sol^n

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

$$n = \left(99/100\right)^K$$

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

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

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

Teacher's Sign

Que. 8. Arrange the following in increasing order of rate of growth.

Solⁿ.

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