Name :- Aman Sharma
Sec. ⇒ CS-D
Roll no. ⇒ 10

Date
Page No.

Tutorial - 2

Q.1 What is the time complexity of below code & how `void fun(int n)`

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

Sol:

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 $i = \frac{i(i+1)}{2} = i^2 < n$

$n = \sqrt{n}$

Q.2 Write recurrence relation for the recursive function that prints fibonacci series. Solve the recurrence relation to get complexity of the program. What will the space complexity of this program and why

Sol:

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

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

Date.
Page No.

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

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

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

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

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

$T(n-4) = 2^k (4 \cdot T(n-2) + 6c) + c$

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

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

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

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

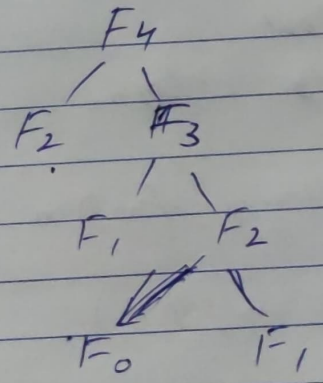
$2^n * 1 + 2^n c - c$

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

$= 2^n$

$O(2^n)$

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



Here, the space complexity of Fibonacci recursive is $O(n)$

Date.
Page No.

Q.3- Write programs which have complexity
$n(\log n)$, n^3, $\log(\log n)$

Sol 3) ⇒ 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, c))
{
    // Some O(1) expressions;
}
```

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) expression
        }
    }
}
```

Date.
Page No.

Q.4 Solve the following recurrence relation.

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

Sol(4): $T(n) = 2T(n/2) + cn^2$ $\left(T(n/2) \geq T(n/4)\right)$

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

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

We get,

$c \log_2 2 = 1$

$f(n) > n^c$

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

$\Rightarrow \Theta(n^2)$

Q.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=j*i)
        {
            // some O(1) task
        }
    }
}
```

Sol(5):- for $i=1 \rightarrow j = 1, 2, 3, 4 \ldots n$ (run for n times)

for $i=2 \rightarrow j = 1, 3, 5 \ldots$ (run for $n/2$ times)

for $i=3 \rightarrow j = 1, 4, 7, \ldots$ (run for $n/3$ time)

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

$= n\left(1 + 1/2 + 1/3 + 1/4 + \ldots\right)$

$= n \int_1^n \frac{1}{n} \Rightarrow n \int_1^n \frac{dn}{n} \Rightarrow \log n \Big]_1^n$

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

Date.
Page No.

Q.6 What should be the time complexity of following function?

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

where k is a constant

Sol(6): for first iteration, $i = 2$

2^{nd} iteration $i = 2^k$

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

$\vdots$

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

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

again apply log $\log(k^i) = \log n \Rightarrow i = \log(\log n)$

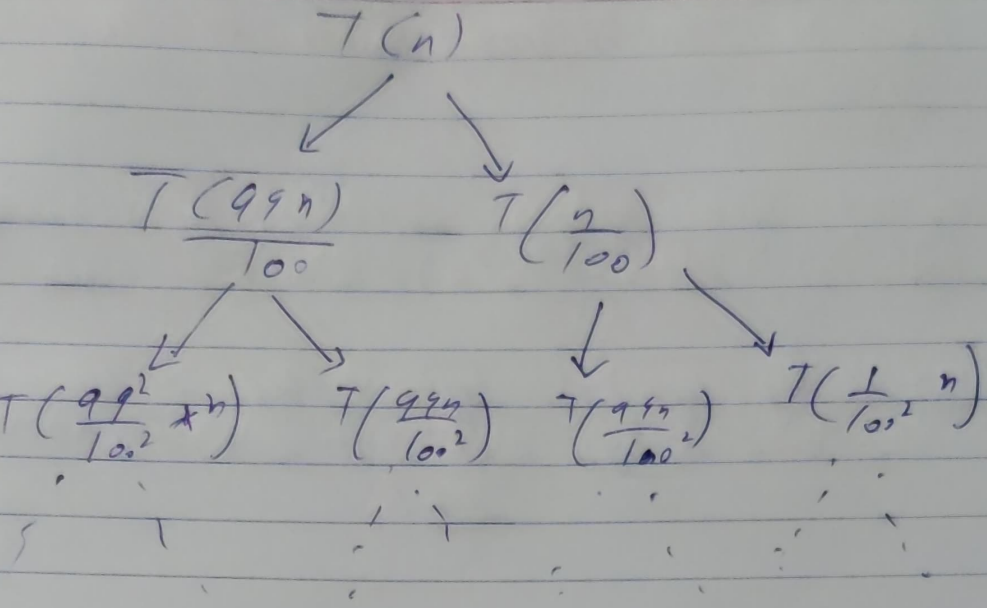
Q.7 Write a recurrence relation when quick sort repeatedly divides the array into two parts of 99% and 1%. derive the time complexity of this case

Sol(7): 99 to 1 in Quick sort possibly when pivot last from either from front or end

So, $T(n) = T\left(\frac{99n}{100}\right) + T\left(\frac{1}{100}\right) + O(n)$

$T(n) = T\left(\frac{99n}{100}\right) + T\left(\frac{n}{100}\right) + O(n)$

Date.
Page No.



$$n \times \left(\frac{99}{100}\right)^k = 1$$

$$\frac{n}{\left(\frac{100}{99}\right)^k} = 1$$

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

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

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

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

Date
Page No.

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

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

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