Name: Nishtha Jain
Section: H
Roll No: 39

Design And Analysis Of Algorithms (Tutorial-2)

Q1 What is the time complexity of below code and how?

```
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)

⋮

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

$\Rightarrow x = \sqrt{n}$

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

⇒ 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:

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

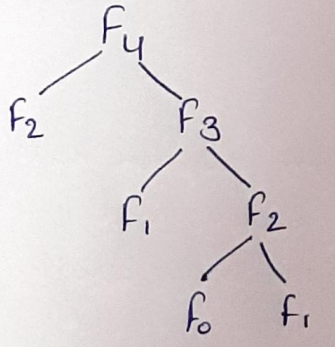
$$T(n-2) = 2*(T2(n-2-2)+C)+C$$
$$= 2*(2T(n-2)+C)+C$$
$$= 4T(n-2)+3C$$

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

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

$$T(n) = 2^n * T(0) + (2^n-1)C$$
$$= 2^n * 1 + 2^n C - C$$
$$= 2^n(1+C) - C \Rightarrow 2^n$$
$$\Rightarrow \underline{O(2^n)}.$$

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



Hence, Space complexity of fibonacci recursive is $\underline{O(n)}$.

Q3 Write programs which have complexity - $n(\log n)$, n^3, $\log(\log n)$

⇒ Complexity: $n(\log n)$ ⇒ Merge Sort

For time complexity: 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
        }
    }
}
```

⇒ Complexity: log(logn)

```
for(int i=2; i<n; i=pow(i,j))
{
    Some O(1) expression
}
```

(Where k is constant)

⇒ Complexity: nlogn)

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

Q4 Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$

⇒ $T(n) = 2T(n/2) + cn^2$

Using Master's Method.

$T(n) = aT(n/b) + f_n$

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

$c = \log_2 2 = 1 \quad f(n) > n^c$

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

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

Q5 What is the time complexity of following function fun()?

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

⇒ for i=1 ⇒ j = 1, 2, 3, ... n (run for n times)
for i=2 ⇒ j = 1, 3, 5, ... n (run for n/2 times)
for i=3 ⇒ j = 1, 4, 7, ... n (run for n/3 times)

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

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

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

T.C. = $n \log n$

Q6 What should be the time complexity of

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

where k is a constant

⇒ for 1st iteration ⇒ $i = 2$
2nd iteration ⇒ $i = 2^k$
3rd iteration ⇒ $i = (2^k)^k = 2^{k^2}$
i^{th} iteration ⇒ $i = 2^{k^i}$ loop ends at $2^{k^i} = n$

Apply $\log$: $\log n = \log 2^{k^i} = k^i = \log n \Rightarrow i = \log_k(\log n)$.

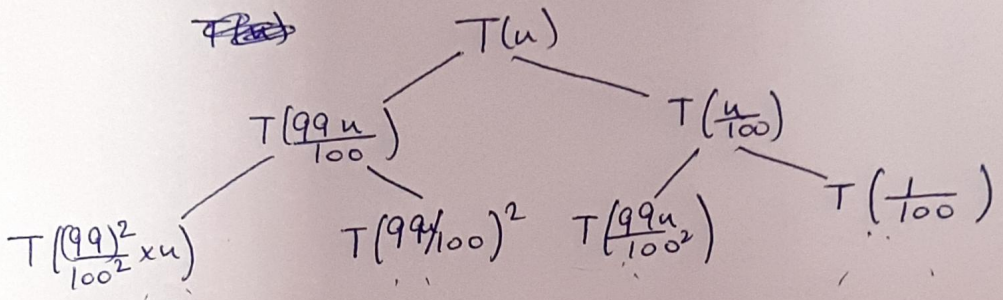
Q7 Write a recurrence relation when quick sort repeatedly divides the array in to two parts of 99% and 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity and find the difference in heights of both the extreme parts. What do you understand by this analysis? (5)

⇒ 99 to 1 in Quick Sort

When pivot is either from front or end always.

Qs, $T(n) = T(99n/100) + T(1n/100) + O(n)$

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



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

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

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

⇒ $TC = n^* \log_{\frac{100}{99}} (n)$

Q8 Arrange the following in increasing order of rate of growth.

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

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

c) $96 < \log n < \log 2n < 8n < n(\log 6n) < n(\log_2 n) < \log(n!) < 8n^2 < 7n^2 < n! < 8^{2n}$