SHREYA SINGH RAGHUVANSHI

61

CST_SPL1

TCS 409

20022399

25|03|22

Design and Analysis of Algorithms
Tutorial- 2

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

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

Sol-

j	~~i~~	i
1	~~0~~	1
2	~~1+0~~	1+2
3		1+2+3
4		1+2+3+4

Let be k times

Sum of k terms < n, So that the loop works

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

$$k^2 < n$$

$$k \approx \sqrt{n}$$

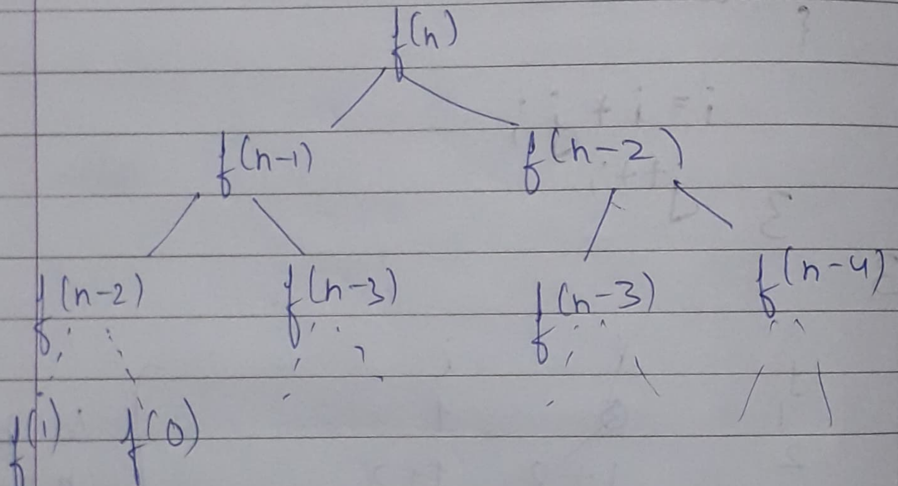
$$\boxed{T(n) = O\sqrt{n}}$$

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 the program & why?

Sol - For fibonacci series:

$f(n) = f(n-1) + f(n-2)$

where $f(0) = 0$, $f(1) = 1$



Every function call generates two fun. calls

For n levels: $= 2 \times 2 \ldots n$ times

$= 2^n$ times

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

Acc to recursion tree, max. height of tree is n

space complexity = $O(n)$

Q3 Write programs which have complexity

i) n(log n)

→
```
for (i=1; i<=n; i*=2)       // log_2 n times
{
    for (j=1; j<=n; j++)    // n times
    {
        sum = sum + i;
    }
}
```

ii) n^3

→
```
for (i=1; i<=n; i++)                  // n times
{
    for (j=1; j<=n; j=j+[illegible])  // (n+1)/3 times
    {
        for (k=1; k<=n; k++)          // n times
        {                             // ((n+1)/3) * n
            // O(1) operations
        }
    }
}
```

iii) log(log n)

→
```
for (i= 2; i<=n; i= pow(i, c))
{
    // Some O(1) op.
}
```

Q4- Solve the following recurrence relation

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

S1- as $T\left(\frac{n}{2}\right) > T\left(\frac{n}{4}\right)$

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

$a = 2,$ $b = 2$ $f(n) = cn^2$

$c = \log_2 2 = 1$

$n^c = n^1 = n$

$f(n) > n^c$

$cn^2 > n$

$\therefore$ $T(n) = \theta(cn^2)$

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

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

```
int fun (int n)
{
    for (int i = 1 ; i<=n; i++)          // n times
    {
        for (int j=1 ; j<n ; j+=i) // n-1/i times
        {
            // some O(1) operation
        }
    }
}
```

N8

Sl-

i	j
1	n
2	n/2
3	n/3
⋮	
n	n/n

$$\text{Time Compl.} = \left(n + \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n}\right)$$

$$= n * \underbrace{\left(1 + \frac{1}{2} + \frac{1}{3} + \dots \frac{1}{n}\right)}_{\text{harmonic series}}$$

$$= n * (\log n)$$

$$\boxed{T(n) = n(\log n)}$$

Q6- What should be the time complexity of
for (int i = 2; i <= n; i = pow(i, k))
{
// Some O(1) expressions or statements
}

Sol-

i

2

2^k — Let this be m times

$(2^k)^k$

2^{k^2}

⋮

~~$2^{k^{\log_k \log(n)}}$~~

⋮

2^{k^m}

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

$$k^m = \log_2 n$$

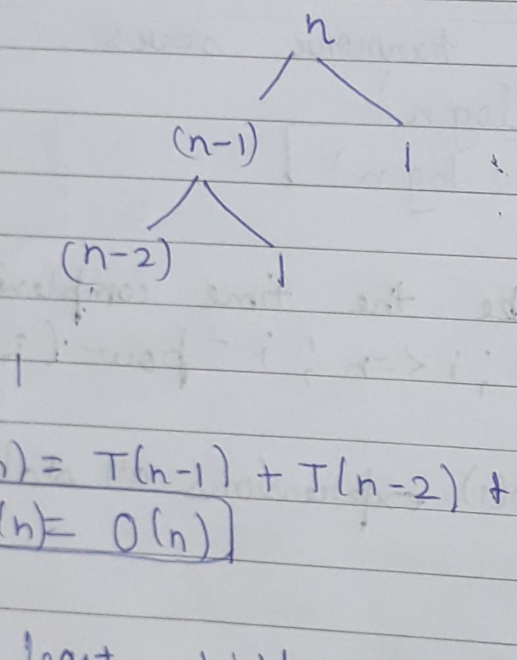
Taking log of base k

$$\log_k k^m = \log_k \log_2 n$$

$$T(n) = \boxed{m = \log_k \log_2 n}$$

Q7- Write a recurrence ------ this analysis?

Sol-



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

$$\boxed{T(n) = O(n)}$$

Least height = 2

Greatest height = n

Difference = n-2

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

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

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