· Section → Harshita Mishra
CST-SPL-1 (08)
Uni. Roll no → 2017498

TUTORIAL-2

Ques1 → What is the time Complexity of below code by → How?

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

Ans → 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 $i = x^2 < n$

$\Rightarrow x = \sqrt{n}$

Ques 2 → Write recurrence relation for the recursive function that prints Fibonacci Series. Solve the recurrence relation to get complexity of the program. What will be the space complexity of this program & why?

Ans → $fib(n) = fib(n-1) + fib(n-2)$ Let $T(0) = 1$

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

Time Complexity →

$$T(n) = T(n-1) + T(n-2) + C$$
$$= 2T(n-2) + C \quad (T(n-1) \cong T(n-2))$$

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

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

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

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

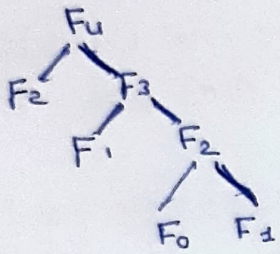
$= 2^n * 1 + 2^n C - C$

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

$\cong 2^n$ // constant can be ignored

$O(2^n)$

Space Complexity → The space is proportional to the maximum depth of the recursion tree.



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

Ques 3 → Write programs which have complexity $n(\log n)$, n^3, $\log(\log n)$?

Ans; $n \log n$

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

n^3

```
int arr[n1][n2][n3];
for (int i=0; i<n1; i++)
{
  for (int j=0; j<n2; j++)
  { for (int k=0; k<n3; k++)
    {
      printf (arr[i][j][k]);
    }
  }
}
```

$\log(\log n)$

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

Ques 4→ Solve the following recurrance relation $T(n)=T(n/4)+T(n/2)+$

Ans→ $T(n) = 2T(n/2) + cn^2$

Using Master's Method, $T(n) = aT(n/b) + f(n)$

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

we get, $c = \log_2^2 = 1$

$f(n) > n^c$

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

$\Rightarrow \theta(n^2)$

Ques 5→ What is the time Complexity of the following function?

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

Ans→ when $i = 1$ inner loop (j) will run n times

when $i = 2$ inner loop(j) will run $n/2$ times

when $i = 3$ inner loop (j) will run $n/3$ times

Similarly, when $i = n$ inner loop(j) will run n/n time

So, $n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$

$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots\right)$

$= O(n \log n)$

Ques6 → What should be the time Complexity of following fun C)?

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

where k is a constant.

Ans → as, 1st → $i = 2$, 2^k, 2^{k^2}, 2^{k^3}, ... 2^{k^i}

$\Rightarrow 2^{k^i} = n$

taking log on both sides

$\log_2 2^{k^i} = \log n \Rightarrow k^i = \log n$

Take again log on both sides with base k

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

So, time Complexity = $\log(\log n) * O(1) = \boxed{\log(\log n)}$

Ques7 → Write a recurrence relation when Quick sort repeatedly divides the array in to two parts of 99% & 1%. Derive the time complexity in this case. Show the recursion tree while deriving time complexity & find the difference in heights of both the extreme parts. what do you understand by this analysis?

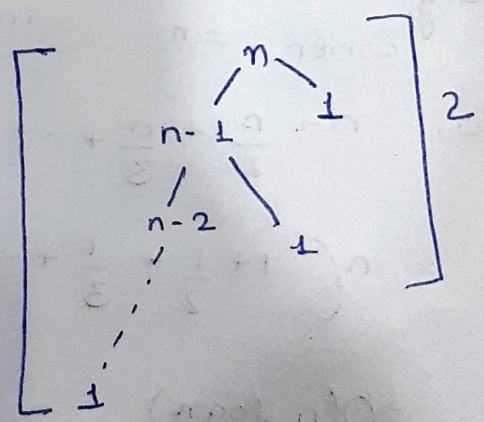
Ans → array is divided into 99% & 1%

$\because T(n) = T(n-1) + O(1)$

$T(n) = (T(n-1) + T(n-2) + \dots + T(1) + O(1)) \times n$

$= n \times n$

$\therefore T(n) = O(n^2)$



lowest height = 2

Heighest Height = n

$\therefore \boxed{diff = n-2}$ $n>1$

The given algorithm provides linear results.

Ques 8 → Arrange the following in increasing order of rate of growth.

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

Ans → $100 < \log\log n < \log^2(n) < \log(n) < \log(n!) < n\log n < \text{root}\,n < n < n! < 2^n < n^2 < 4^n < 2^{2n}$

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

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

(c) $\theta^\wedge(2n), \log_2(n), n\log_6(n), n\log_2(n), \log(n!), n!, \log_\theta(n), 96, \theta n^2, 7n^3, 5n$

Ans → $96 < \log\theta^n < \log_2^n < \log(n!) < n\log_6 n < n\log_2^n < 5n < \theta n^2 < 7n^3 < n! < \theta^\wedge(2n)$