Name: Nikita Kumari

Roll : 19

Section: CST SPL-1.

Date :

Page :

Tutorial- 2

Q1.

$j = 1$ $\qquad$ $i = 1$

$j = 2$ $\qquad$ $i = 1 + 2 = 3$

$j = 3$ $\qquad$ $i = 3 + 3 = 1 + 2 + 3$

⋮

$j = K$ $\qquad$ $i = 1 + 2 + 3 + 4 + \ldots k$

sum of K consecutive integers $= \frac{K(K+1)}{2}$

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

$\Rightarrow \frac{K^2 + K}{2} < n$

$K^2 < n$ (ignoring constants & small terms)

$\therefore K < \sqrt{n}$

Hence, Time complexity $= O(\sqrt{n})$. Ans

Q2. Recursive relation for Fibonacci series:

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

$T(n)$ — 1

$T(n-1)$ $\quad$ $T(n-2)$ — 2

$T(n-2)$ $\quad$ $T(n-3)$ $\quad$ $T(n-3)$ $\quad$ $T(n-4)$ — 4

⋮

$\Rightarrow 1 + 2 + 4 + 8 \ldots$

Here, $a = 1$ $\qquad$ $r = 2$

So,

$$\frac{a(r^{terms}-1)}{r-1} = \frac{2^{terms}-1}{1}$$

$$= 2^n \cdot 2 = O(2^n).$$ Ans.

Space complexity : O(1)
Since no any extra space is required during the execution.

Q3. 1. n(logn)

```
void quick_sort(int a[], int lb, int ub)
{
  int i = lb, j = ub;
  int key = a[lb];
  int t = 0;
  if( lb >= ub)
  return;
    while(i < j) {
    while (key >= a[i] && i < j)
      i++;
    while ( key < a[j])
      j--;
    if (i < j) {
      t = a[i];
      a[i] = a[j];
      a[j] = t;
    }
  }
  a[ lb] = a[j];
  a[j] = key;
```

```
    quick_sort(a, 0, j-1);
    quick_sort(a, j+1, ub);
}
```

(b) $O(n^3)$

→
```
for(int i=0; i<n; i++)
    // some O(1)
for(int i=0; i<n; i++)
{
    // O(1)
    for(int j=0; j<n; j++)
    {
        // O(1)
    }
}
```

(c) $O(\log(\log n))$

→
```
int p=0;
for(int i=1; i<n; i=i*2)
    p++;
for(j=1; j<p; j=j*2)
{
    // O(1)
}
```

Q4. $T(n) = T(n/4) + T(n/2) + cn^2.$

$= 2T\left(n/2\right) + cn^2.$

Using Master's method $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a \geq 1, \quad b > 1, \quad c = \log_b a.$

$C = \log_2 2 = 1$

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

$T(n) = (f(n))$

$\therefore \quad O(n^2)$. Ans.

Q5.

i	j	
1	1, 2, 3, ...	n times
2	1, 3, 5, 7 ...	n/2 times
3	1, 4, 7, 11, ...	n/3 times
4	⋮	
⋮	⋮	
n	J = 1 ...	n, n/2, n/3 ... times

$T(n) = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \dots + 1$

$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots \frac{1}{n}\right)$

So, $T(n) = n(\log n)$. Ans.

Q6. $T(n) = 2, 2^k, 2k^2, 2^{k^3} \dots 2^{k \log k(\log n)}$

So, $2^{k \log k(\log n)} = 2^{\log n} = n$

So, Total Time complexity $= T_n = O(\log k(\log n))$ Ans.

Date : / /
Page :

Q8. a. $100 < \log(\log n) < \log(n) < \log^2 n <$
$\sqrt{n} < n < n \log n < n^2 < 2^n < 4^n <$
$2^{2^n} < \log(n!) < n!$

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

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