NAME → Avanika Singh
ROLL NO. → 01
SECTION → CST SPL 1

PAGE No.
DATE

TUTORIAL-2

Q1. find time complexity

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

j = 1	i = 1
j = 2	i = 1+2 = 3
j = 3	i = 3+3 = 1+2+3
⋮	⋮
j = k	i = 1+2+3+ ... +k

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

$\frac{k^2+k}{2} < n$

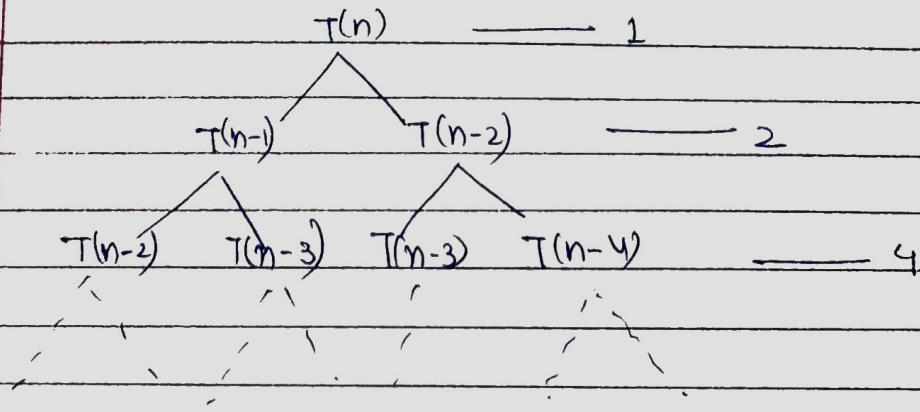
$k^2 < n$ (ignoring constants)

$k < \sqrt{n}$

time complexity = $\boxed{O(\sqrt{n})}$

Q2. Recursive Relation for fibonacci series:

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



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

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

So,

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

$$= 2^n - 1$$

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

Q3. i) $n(\log n)$

```
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(0, j+1, ub);
}
```

(ii) $O(n^3)$

```
for (int i=0; i<n; i++)
{
    for (int j=0; j<n; j++)
    {
        for (int k=0; k<n; k++)
        {
            sum+=k;
        }
    }
}
```

iii) $O(\log(\log n))$

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

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

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

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

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

$c = \log_2 2 = 1$

$\because f(n) > n^c$, $\boxed{T(n) = f(n) = O(n^2)}$ Ans

PAGE No.
DATE

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

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

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

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

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

So,

Total time complexity $\boxed{T(n) = O(\log_k(\log n))}$ Ans

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

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

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