Complexity Analysis of Iterative Algo.

Q.1. A( )

```
{ int i;
for (i=1 to n)
  pf ("Ravi");
}
```

Complexity.

$O(n)$.

Q.2 A( )

```
{ int i, j;

for (i=1 to n)
  for (j=1 to n)
    pf ("Ravi");
}
```

$O(n^2)$.

```
Q. A( )
{ i=1, s=1;
while(s<=n)
{ i++;
s=s+i;
pf("Ravi");}
}
```

s	1	3	6	10	15	21	.....	(n).1
i	1	2	3	4	5	6	.....	K.

Here s value is sum of 1st n natural no. of i. i.e. S = 3 ⇒ 1+2.
= 6 ⇒ 1+2+3.
= 10 ⇒ 1+2+3+4.

i 2 3 4 5
2
3 5

S = sum of 1st K natural no.
= K(K+1)/2

$\frac{k(k+1)}{2} > n$ if loop has to be stop.

$k^2 > n \Rightarrow (k = 0\sqrt{n})$

```
}
```

Q. A( )

```
{ int i,J,K,n;

for (i=1; i<=n ; i++)
{ for(J=1; J<=i, J++)
  { for (K=1; K<=100; K++)
    { Pf ("Raj");
  }
}
}
```

i=1 J=1 K=100

i=2 J=2 K=2×100

i=3 J=3 3×100

i=4 J=4 K=4×100,

i=n J=n. K=n×100.

$$K = 100 + 2\times100 + 3\times100 + \dots n(\times100)$$

$$= 100(1+2+3\dots n)$$

$$= 100\left(\frac{n(n+1)}{2}\right)$$

$$= 100(n^2)$$

$$= O(n^2)$$ ✓

A( )

{ int i, j, k, n;

for(i=1 ; i<=n; i++)

{ for (j=1; j ≤ i^2 ; j++)

{ for (k=1 ; k ≤ n/2 ; k++)

{ pf ("Rahul");

} }

}

i=1	i=2	3
j=1	j=4	9
k=n/2	k=n/2 *4	n/2 x 9

..... n.

n^2.

$\frac{n}{2} \times n^2$.

$$k = \frac{n}{2} + \frac{n}{2} \times 4 + \frac{n}{2} \times 9 \cdots \frac{n}{2} \times n^2$$

$$= \frac{n}{2} \left[1+4+9 \cdots n^2 \right]$$

$$= n/2 \left[\frac{n(n+1)(2n+1)}{6} \right]$$

$$= O(n^4)$$ ✓.

for (i=1, i<n, i*2) ⇒ $O(\log_2 n)$

for (i=1 ; i<n ; i*3) ⇒ $O(\log_3 n)$

↳ (no need to unroll as j value dsn't depend on

for (i=n/2 ; i≤n ; i++) → n/2 times. I)

for (j=1 ; j≤n/2 ; j++) → n/2 times.

for (k=1 ; k≤n k=k*2) → $(\log_2 n)$

$= O(n/2 \times n/2 \times \log_2 n) \Rightarrow O(n^2 \log_2 n)$

for (i=n/2 ; i<=n ; i++) → n/2.

for (j=1 ; j<=n ; j=2*j) → $\log_2 n$.

for (k=1 ; k<=n ; k=k*2) → $\log_2 n$.

pf ("Ravi");

}

$\Rightarrow n/2(\log_2 n)^2$

$\Rightarrow O[n(\log_2 n)^2]$ ✓

Q. Whenever inner loop does not depend on outer loop, no need to unroll it, just multiply how many no. of times each loop runs.

⇒ (If inner loop depends on outer loop, unroll it.)

Ex.
```
for (i=1 ; i<=n ; i++)
    for (J=1 ; J<=n; J=J+i)
        Pf("Rani");
{
```

(depend on i.) so

i=1 : J=1 to n — n times

i=2 : J=1 to n — n/2 times

i=n : J=1 to n — n/n times.

⇒ $n + n/2 + n/3 + \dots n/n \Rightarrow n(1 + 1/2 + 1/3 + \dots 1/n)$.

How to write Rewrrence for Rewrsion.

1)

```
f (int n)
{
  if (n==1) return;

  for (int i=1 ; i<=n ; i++)
      for (int j=1 ; j<=n ; j++)
          printf (" * ");          ——— n^2.
  fun(n-3);
}
```

Ans:

$$T(n) = c \cdot n^2 + T(n-3).$$

```
f(int n)
{ if (n<=1) return;
  for (int i=1 ; i<=3 ; i++)
    f(n-1).
}
```

$$T(n) = 3T(n-1) + c.$$

⑨.

```
for (i to n)
{ t = t + 1 ;
}
f (n/2);
}
```

$\Rightarrow T(n) = T(n/2) + n$;

⑩.

```
void f (int n)
{ if (n<=1) return;
  if (n>1)
  {
    Printf ("*");
    f (n/2);
    f (n/2);
  }
}
```

→ constant.

$\Rightarrow T(n) = T(n/2) + T(n/2) + 1$;

→ 1.

+ n/2

+ n/2

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

$= 1$ if $n=1$.

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

$T(n-2) = n-2 + T(n-3)$

K 1

$\Rightarrow T(n) = n + [(n-1) + T(n-2)]$.

$= n + (n-1) + (n-2) + T(n-3) \cdots$

$= n + (n-1) + (n-2) + (n-3) + T(n-4)$

$\vdots$

K.

let's suppose $\rightarrow$

$= n + (n-1) + (n-2) + \cdots + (n-K) + T(n-(K+1))$

$= n + (n-1) + (n-2) + \cdots n-K + T(n-(K+1))$

$n-2$.

$\Rightarrow n-(K+1) = 1$ $\quad n-(n-2+1)$

$\Rightarrow n-K-1 = 1$ $\quad n-K+2-1) = (1)$.

$n-K = 2$.

$K = n-2$.

$\Rightarrow n + (n-1) + \cdots n - (n-2) + 1$.

$\Rightarrow n + (n-1) + (n-2) + \cdots 2 + 1$.

$\Rightarrow \frac{n(n+1)}{2} = O(n^2)$

$T(n) = n * T(n-1)$

$= 1$ if $n \leq 1$.

$T(n-1) = n-1 \times T(n-2)$.

$T(n-2) = n-2 \times T(n-3)$ . . .

$T(n) = n \times [\, n-1 \times T(n-2)$

$= n \times n-1 \times n-2 \times T(n-3)$ . . .

$= n \times (n-1) \times (n-2) \times \cdots (n-k) \times T(n-(k$

(k steps)

$T(1) = 1$.

$n - (k+1) = 1$

$n - k - 1 = 1$

$n - k = 2$

$\boxed{k = n-2}$

$= n \times (n-1) \times (n-2) \times \cdots (n-(n-2)) \times 1$

$= \underbrace{n * n-1 * n-2 * \cdots 2 \times 1}_{n!}$.

$= n! = \boxed{O(n^n)}$

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

$= 1$ if $n = 1$.

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

$T(n-2) = 1 + 2T(n-3)$

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

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

$= 1 + 2 + 2^2[(1 + 2T(n-2))]$

$= 1 + 2 + 2^2 + 2^3 T(n-3)$

$\Rightarrow$ $\vdots$ k.

$2^k T(n-k)$

$= 1$

$n = k+1$

2^{n-1}.

$k+1-k$

$= 1 + 2 + 2^2 + \cdots 2^k T(n-(k+1))$

$\Rightarrow n-(k+1) = 1$.

$\Rightarrow n-k-1 = 1$

$n-k = 2$

$\boxed{k = n-2}$

$\Rightarrow 1 + 2 + 2^2 + \cdots 2^{n-2} \cdot T(n-(n-2+1))$

$T(n-(n-1))$

$T(1)$.

$\Rightarrow 1 + 2 + 2^2 + \cdots 2^{n-2}$.

$= O(2^n)$

Recursive Model.

① $T(n) = T(n-1) + 1$.

Ans.

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

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

$= 1 + T(n-3)$

$T(n-3) = 1 + T[(n-3)-1]$

$= 1 + T(n-4)$.

place these values →

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

$= 1 + 1 + 1 + T(n-3)$

$= 1 + 1 + 1 + 1 + T(n-4)$

$= 4 + T(n-4)$

:

To stop this recursion $T(n-k) = 1$.

$= k + T(n-k)$. ⇒ so $n-k = 1$.

$(k = n-1)$

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

$= n-1 + T(1)$

$= n-1+1$.

$= O(n)$.