Time Complexity - 2

T.C ?

Prob $\rightarrow C_1 \longrightarrow O(n)$

$C_2 \longrightarrow O(n^2)$

$C_3 \longrightarrow O(\log_2^n)$

compare & decide which is best

C_3 is better

$$O(1) < O(\log_2^n) < O(\sqrt{n}) < O(n) < O(n \cdot \log_2^n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

best — worst

TC

$O(1)$: if, else, break, continue, return, All arithmetic operation etc..

print ✓

Code -> Time Complexity ?

Note:- when we are finding the T.C of any program
we can take approximate values, exact values are not required

1 Template1 :-

$n * 1 = n$ ✓

```
for(i=1;i<=n;i=i+1)
{
        print("*");
}
```

→ $\frac{n}{1}$ n times ↑ O(n)

print("*"); → O(1)

↳ constant. time

$n = 10 \rightarrow 5 (*)$

```
for(i=1;i<=n;i=i+2)
{
        print("*");
}
```

→ $\frac{n}{2}$ times $= \cancel{\frac{1}{2}} \cdot n \Rightarrow O(n)$

we ignore constant's.

# of times loop runs.

```
for(i=1;i<=n;i=i+a)
{
        print("*");
}
```

→ $\frac{n}{a}$ times.

```
for(i=1;i<=n;i=i+3)
{
        print("*");
}
```

→ $\frac{n}{3}$ times O(n)

⟹

```
for(i=1;i<=n;i=i+n)
{
        print("*");
}
```

$\rightarrow \frac{n}{n} = 1 \Rightarrow O(1)$

whenever we find the T.C
always see how many times loop runs

based on that decide O()

1st template.

```
for(i=1;i<=n;i=i+a)
{
        print("*")
}
```

$\rightarrow \frac{n}{a}$ times.

$a=1$ ✓	$a=n$ ✓
$O(n)$	$O(1)$

Remember

Note :-

if( ), else, break, continue, return, print } $O(1)$ ✓

constant time

Template2

```
for(i=1;i<n;i=i*2)
{
        print("*")
}
```

(P.) Loop runs

$n = \cancel{10}$ ✓ → 100 times

size of input

$\cancel{5}$ → 25 "

6 → 36 "

$O(n^2)$

(P) $n = 1\cancel{4}4$ → 12 times

$\cancel{4}9$ → 7 "

625 → 25 "

$O(\sqrt{n})$

(P) if $n = 2^{10}$ → 10 times

$n = 2^5$ → 5 "

$n = 2^{100}$ → 100 "

$\because \log_2 2^{10} = 10$ ✓

$\therefore O(\log_2 n)$

$* \quad \log_a a^m = m \cdot \underbrace{\log_a a}_{1} = m$

in T.C, we focus on how many times loop runs

$\frac{n}{2} \rightarrow O(n)$

Assume

```
for(i=1;i<n;i=i*2)
{
    print("*")
}
```

→ $\underline{K}$ times ✓ ✓ → $O(K)$

```
for(i=1;i<=n;i=i*2)
{
    print("*")
}
```

→ $\underline{K+1 \text{ times}}$ ⇒ $\underline{O(K)}$

```
for(i=1;i<=n;i=i*2)
{
    print("*")
}
```

Loop runs k times

1st	2nd	3rd	4th	5	6		k
$i = 1$	2	4	8	16	32	......	
*	*	*	*	*	*		

$2^{k-1} > n$ (stop)

$4 \rightarrow 2^3$

$5 \rightarrow 2^4$

$6 \rightarrow 2^5$

$\vdots$

$k \rightarrow 2^{k-1}$

;i<=n

$2^{k-1} > n$

$2^{k-1} = n$

to make cal. easy.

$2^k = n$

Apply $\log_2$ on both sides

$\log_2 2^k = \log_2 n$

$k \cdot \underbrace{\log_2 2}_{1} = \log_2 n \Rightarrow k = \log_2 n$

$\therefore O(\log_2 n)$

* Remember:-

$i = 1 \xrightarrow{*2} 2 \xrightarrow{*2} 4 \xrightarrow{*2} 8 \longrightarrow \cdots \xrightarrow{*2} n$

$O(\log_2 n)$

```
for(i=n;i>=1;i=i/2)
{
    print("*")
}
```

$\rightarrow O(\log_2 n)$

$n \rightarrow n/2 \rightarrow n/4 \rightarrow \cdots \; 1$

```
for(i=1;i<n;i=i*3)
{
      print("*")
}
```

$\rightarrow \log_3^n$

$3 \to a \hookrightarrow \log_a^n$

$\rightarrow$ Generalaized example

```
for(i=1;i<=n;i=i+a)
{
      print("*")
}
```

$\rightarrow \frac{n}{a}$ time

```
for(i=1;i<=n;i=i*a)
{
      print("*")
}
```

$\rightarrow \log_a^n$ times

$a \to 2 \hookrightarrow \log_2^n$

Qn)

```
OL for(i=1;i<=5;i++) → 5
   {
     IL for(j=1;j<=5;j++) → 5
        {
              print("masai")
        }
   }
```

$5 \times 5 = \underbrace{25 \text{ times}}$

Un-rolling process

$i = 1$
$j = 1\ 2\ 3 \cdot \cdot\ 5$
m, m, m
5 (M)

+

$i = 2$
$j = 1\ 2\ 3 \cdot \cdot\ 5$
m, m, m
5 (M)

+

$i = 3$
$j = 1\ 2\ 3 \cdot \cdot\ 5$
m, m, m
5 (M)

+

$i = 4$
$j = 1\ 2\ 3 \cdot \cdot\ 5$
m, m, m
5 (M)

+

$i = 5$
$j = 1\ 2\ 3 \cdot \cdot\ 5$
m, m, m
5 (M)

$5 + 5 + 5 + 5 + 5 = 25$ times

Nested Loop

No - Dependency

C1

OL
```
for(i=1; i<=n;i++)
{
    for(j=1;j<=n;j++)
    {

        c=c+1

    }
}
```
IL

dependency

C2

OL
```
for(i=1; i<=n;i++)
{
    for(j=1;j<=i;j++)
    {

        c=c+1

    }
}
```
IL

number of times inner loop runs depends on outer loop variable

No - Dep

$n * n \Rightarrow O(n^2)$

```
i for(i=1; i<=n;i++)        --n times-->  OL
  {
     j for(j=1;j<=n;j++)    --n times-->  IL
       {

          c=c+1  -> O(1)

       }
  }
```

Qn) No-dep

```
i for(i=1;i<=n;i++)        ⟶ n times
  {                              ×
  j for(j=1;j<=n/4;j++)    ⟶ n/4 times
    {                              ×
    k for(k=1;k<=n;k++)    ⟶ n times
      {

          print("*")  → O(1)
      }
    }
  }
```

$\frac{n^3}{4} \Rightarrow O(n^3)$

Q) No-Dep ✓

```
i for(i=1;i<=n;i++)  → n
{
    j for(j=1;j<=n/4;j++)  → n/4
    {
        for(k=1;k<=n;k++)  → run's only once
        {
            print("*")
            break; ✓
        }
    }
}
```

$$n * \frac{n}{4} * 1 = \frac{n^2}{4}$$

$$= O(n^2)$$

Qn) No-dep

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j++)
    {
        for(k=n/2; k<=n; k=k+n/2)
        {

            c=c+1
        }
    }
}
```

✓ n

✓ n

$n * n * 2 = 2 \cdot n^2 \Rightarrow O(n^2)$

$K = \frac{n}{2} + \frac{n}{2} = n + \frac{n}{2} = \frac{3n}{2} = 1.5 \cdot n \leq n$ ✗

2 times (cosnst)

```
i=1
while(i<n)
{
    i=i*2
    print(*)
}
```

$i = 1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \; . \; . \; . \; . \rightarrow \frac{n}{2} \rightarrow n$

$O(\log_2 n)$

Q) No-dep

```
i for(i=1; i<=n; i++)     → n times ✓
  {
    j for(j=1; j<n; j=2*j)     → log₂ⁿ times.
      {
          c=c+1
      }
  }
```

$O(n\log_2^n)$

Q) dep ✓ ⟶ un-rolling process

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=i; j++)  → i times
    {
        c=c+1
        print(*)
    }
}
```

\# of times IL Runs (i times)

$i=1 \rightarrow 1$
$+$
$2 \rightarrow 2$
$+$
$3 \rightarrow 3$
$\vdots$
$+$
$n \rightarrow n$

$$1+2+3+\dots+n = \frac{n(n+1)}{2} \Rightarrow O(n^2)$$

```
for(j=1; j<=i; j++)  → i/1 = i times
{
    c=c+1
    print(*)
}
```

```
for(i=1;i<=n;i=i+a)  → n/a times
{
    print("*")
}
```

Qn) dep → un. rolling process

```
i for(i=1; i<=n; i++)
  {
      for(j=1; j<=n; j=j+i) → n/i times
      {
          c=c+1
      }
  }
```

\# of times IL Runs ($\frac{n}{i}$ times)

$i=1 \rightarrow \frac{n}{1} = n$

$+$

$2 \rightarrow \frac{n}{2}$

$+$

$3 \rightarrow \frac{n}{3}$

$+$

$4 \rightarrow \frac{n}{4}$

$\vdots$

$+$

$n \rightarrow \frac{n}{n} = 1$

$$= n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + 1$$

$$= n \Big[\underbrace{1 + 1/2 + 1/3 + 1/4 + \cdots + 1/n}_{\log n} \Big]$$

$= O(n \log n)$ ✓

```
for(j=1; j<=n; j=j+i) → n/i times
{
    c=c+1
}
```

Q) dep. → un. rolling

$1+1+1+\cdots+1 = 10$ (10 times)

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

→ n−i times

$i = 1 \rightarrow n-1$

$2 \rightarrow n-2$

$3 \rightarrow n-3$

$\vdots$

$n \rightarrow 0$ $(n-n=0)$

$n-1 + n-2 + n-3 + n-4 + \cdots + 1 + 0$

$= \underbrace{n+n+n+\cdots+n}_{n*n} + (-1-2\cdots-n)$

$=$

$= O(n^2)$

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

$\frac{n}{1} - i$

```
for(i=1;i<=n;i=i+a)
{
    o(1)
}
```

$\rightarrow \frac{n}{a}$ times -1

```
for(i=10;i<=n;i=i+a)
{
    o(1)
}
```

$\rightarrow \frac{n}{a}$ times -10

```
for(i=1;i<=n;i++)
{
    for(j=1;j<=sqrt(n);j++)
    {
        // O(1)
    }
    for(k=1;k<=n;k=k*2)
    {
        //O(1)
    }
}
```

$\rightsquigarrow n$ times

$\rightsquigarrow \sqrt{n}$ times

$\rightsquigarrow \log_2 n$ times

$O(n \cdot \sqrt{n})$

```
L1 for(i=1;i<=n;i++)      ~~> n times ✓
   {
      L2 for(j=1;j<=sqrt(n);j++)   ~~> √n times
         {
               // O(1)
         }
      L3 for(k=1;k<=n;k=k*2)   ~~> log2 n times
         {
               //O(1)
         }
   }
```

$O(n \cdot \sqrt{n} * \underbrace{\log_2^n}_{\text{NO.}})$

Sig. term

$O(\ )$

$L_1 * [L_2 + L_3]$

max

$n [\sqrt{n} + \log_2^n]$

max = ?

Ans

$n[\sqrt{n}] = O(n \cdot \sqrt{n})$

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

H/W

```
function fun(n)
{
    for(i=1;i<=n;i++)
    {
        p=0
        for(j=n; j>1; j=j/2)
        {
            ++p
        }

        for(k=1; k<p; k=k*2 )
        {
            ++q
        }
    }
}
```