Starts @ 10:15 am.

ACTIVITY:-

Guess whose logo is it

1

Starbucks

2

Malibu

3

Atari

4

5

CBS

‘

Unilever

7

Kodak

8

BiC

٩

beats

10

You Tube

Constant Operations – includes

O(1)

- print statements
- if-else condition
- break
- increment/decrement statement
- continue
- return

Theirs time-complexity is always O(1)

for (int i = 1; i <= N; i++)
{
if (i == 2)
print( )
}
→ O(1)

[1, N] = N

O(N)

- Time complexities can be represented by any of the mathematical functions we studied in the last class.

Sno	Function Name	Function Expression
1	Constant	1
2	Logarithmic	log(n)
3	Square root	√n
4	Linear	n
5	$\boxed{\text{Linearithmic}}$	n.log(n)
6	Quadratic	n^2
7	Cubic	n^3
8	Exponential	[redacted] x^n
9	Factorial	n!

$\boxed{\text{Linearithmetic} \downarrow n \log n}$

$\boxed{\text{10 mins break}}$

```
for i in range(n):          [0, n-1]
    for j in range(n/4)
        for k in range(n):
            print("*")
```

TC ⇒ ? $\boxed{O(n^3)}$

$O(n^3)$

i	j	k
1	1	n
	2	n
	3	n
	⋮	n
	n/4	
		n
2	1	
		n
	2	
⋮	⋮	n
	n/4	
		n
n	1	n
	2	

$$\begin{array}{ccccc} i & & j & & k \\ n & * & n/4 & * & n \end{array}$$

$$\Rightarrow \frac{n^3}{4}$$

$(n+1)(\log_2^n)$

$= n\log_2^n + \cancel{\log_2^n}$

i	j
1	$\log_2^n$
2	$\log_2^n$
3	$\log_2^n$
4	$\log_2^n$
⋮	
n+1	$\log_2^n$

$\Rightarrow \boxed{O(n \log_2^n)}$

```
n=5
c=0                    [1, n+1]
for i in range(1,n+1):
  j=1
  while(j<n):
    c+=1
    print(c)
    j*=2
```

$\log_2^n$

```
j = 1
while (j<n) {
    c+=1
    print(c)
    j *=2
}
```

iteration	j value after every iteration
1	$j*=2 \Rightarrow 2 \rightarrow 2^1$
2	$j=2*2 \Rightarrow 4 \rightarrow 2^2$
3	$j=2*4 \Rightarrow 8 \rightarrow 2^3$
4	$j=8*2 \Rightarrow 16 \rightarrow 2^4$
⋮	⋮
k	$\rightarrow 2^k$

After k iterations, stop.

$2^k = n \Rightarrow \log_2 2^k = \log_2^n$

$\boxed{k = \log_2^n}$

```
for (int i= 0; i<n; i++){
    int j = 1
    while (j<=n){
        j*=2;
    }
    for(int k=0; k<n; k++)
        print( * )
    }
}
```

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

```python
n=2
for i in range(n):
  j=1
  while(j<=n):
    j*=2
  for k in range(n):
    print("*")
```

Obs:-

When 2 loops are independent of eachother, then take the maximum one.

i	j	k	max of j, k
1	($\log_2 n$	n)	n
2	($\log_2 n$)	n)	n
⋮			
n	($\log_2 n$	n)	n
			+

$n + n + n$ — - — n times.

$\Rightarrow n * n \Rightarrow n^2$

```python
c=0
for i in range(n):
  for j in range(n):
    for k in range((n/2),n+1,2):
      c+=1
```

for (int k = n/2 ; k < (n+1) ; k += 2)

```
for (int i=0; i<n; i++) {
    for( int j=0; j<n; j++) {
        for (int k=n/2 ; k<n+1 ; k*=2) {
            print( ___ ).
        }
    }
}
```

2times.

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

i	j	k
0	0	2
	1	2
	2	2
	⋮	
	n	2
1	0	2
	1	2
	⋮	
	n	2
⋮		
n	0	2
	1	2
	⋮	
	n	2

~~$O(n^2)$~~ ~~$O(n \log n)$~~

$\frac{n}{2}$ $\quad \frac{n}{2} * 2 = n \quad n * 2 = 2n$

i j * k

$n * n * 2 \Rightarrow \cancel{2} n^2 \Rightarrow \boxed{O(n^2)}$

```python
c=0
for i in range(n):
  for j in range(n):
    for k in range((n/2),n+1,(n/2)):
      c+=1
      print(c)
```

sum n

$2^k = n$

$\log_2 2^k = \log_2 n$ $\Rightarrow$ $\boxed{k = \log_2 n}$

i	After every iteration
1	$i *= 2 \Rightarrow 2 \rightarrow 2^1$
2	$i = 4 \rightarrow 2^2$
3	$i = 8 \rightarrow 2^3$
4	$i = 16 \rightarrow 2^4$
k	$n \rightarrow 2^k$

```
n=6
i=1
while(i<n):
  print("*")
  i=i*2
```

$\boxed{T.C \Rightarrow O(\log_2 n)}$

p.s int (int n)
{
i=1
while(i<n)
print
i*=2
}

```
n=5
c=0
for i in range(1,n+1):
    j=1
    while(j<n):
        c+=1
        print(c)
        j*=2
```

$\rightarrow (n+1)(\log_2^n)$

$\rightarrow \log_2^n$

$\rightarrow$ T.C $\rightarrow O(n \log^n)$

```python
n=2
for i in range(n):
  for j in range(n**2):
    print("*")
  for k in range(n**3):
    print("*")
```

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

$\rightarrow O(n^4)$

i	j	k	max
1	n^2	n^3	n^3
2	n^2	n^3	n^3
⋮			
n	n^2	n^3	n^3

$n^3 + n^3 + n^3 + \dots$ + n times

$n * (n^3)$

$\boxed{n^4}$

```python
n=2
for i in range(n):
    j=1
    while(j<=n):
        j*=2
    for k in range(n):
        print("*")
```

$\rightarrow n$

$\rightarrow \log_2 n$

$\rightarrow n$

n^2

```python
def fun(n):
  count=0
  for i in range(int(n/2),n+1):
    for j in range(1,n+1,2):
      count+=1
  print(count)

fun(5)
```

```
n=5
i=n
x=0
while(i>=1):
  for j in range(1,n+1):
    x+=1
  i=int(i/2)
print(x)
```

Time complexity

Problem: Find TC and SC → space complexity

```
def fun(a,b,c):
    p=a
    q=b
    r=c
    print(p+q+r)
    return a+b+c
x=fun(3,4,5)
print(x)
```

O(1)
O(1)
O(1)
→ O(1)
→ O(1)
→ O(1)
→ O(1)

T.C. ⇒ O(1)

S.C. ⇒ O(1)

```
count = 0;
for (i = 0;i<N;i++){
    count++;
}
j = N;
while(j >= 0){
    count++;
    j--;
}
```

```
count = 0;
for (i = 0;i<N;i++){
    count++;
}
for (j = 0;j<N;j++){
    count++;
}
```
How many times the above code will run?

```
sum = 0;                                    //1
for (i=0;i<N;i++){                          //2
    j = 0;                                  //3
    while(j < N){                           //4
       sum += i;                            //5
       j++;                                  //6
    }                                       //7
}                                           //8
```

How many times will the above code run?

A N B N^2 C N(LogN) D Log(N)

```
count = 0;
for (i = 0;i<N;i++){
    for (j = 0;j<N;j++){
        count++;
    }
}
```

How many times will the above code run?

A N B N^2 C Log(N) D None Of These