DOMS | Page No.
Date / /

1 Dry Run & Analyze : Time & space complexity

1. n = 4

```
void PrintTriangle (int n){
    for(int i = 0; i<n; i++)
        for (int j = 0 j<=i; j++)
            System.out.Print (" * ")
```

1st iteration: print
i = 0 & j = 0 *

For 2nd iteration
i = 1 & j = 0, 1 * *

for 3rd iteration
i = 2 & j = 0, 1, 2 * * *

for 4th iteration
i = 3 & j = 3 * * * *

* Time complexity :-

i	j	No. of iteration
0	0	0
1	0, 1	1
2	0, 1, 2	2
3	0, 1, 2, 3	3
4	0, 1, 2, 3	4

∴ Sum of First n natural numbers

$$S = \frac{n(n+1)}{2}$$

$$f(n) = \frac{n^2+n}{2}$$

① ignore constant
② take bigger
∴ $O(n^2)$

2) Dry run for n = 8. what's the number of iterations? Time complexity?

→ 1st iteration

		Print
i = 1	j = 0	1, 0
i = 1	j = 1	1, 1
i = 1	j = 2	1, 2
i = 1	j = 3	1, 3
i = 1	j = 4	1, 4
i = 1	j = 5	1, 5
i = 1	j = 6	1, 6
i = 1	j = 7	1, 7

2nd iteration.

i = 2	j = 0	2, 0
i = 2	j = 1	2, 1
i = 2	j = 2	2, 2
i = 2	j = 3	2, 3
i = 2	j = 4	2, 4
i = 2	j = 5	2, 5
i = 2	j = 6	2, 6
i = 2	j = 7	2, 7

Code: void PrintPattern (int n) {
for (int i=1; i<=n; i*=2) {
for (int j=0; j<n; j++) {
System.out.println (i+","+j); } }

3rd iteration

i=4,	j=0	4,0
i=4	j=1	4,1
i=4	j=2	4,2
i=4	j=3	4,3
i=4	j=4	4,4
i=4	j=5	4,5
i=4	j=6	4,6
i=4	j=7	4,7

4th iteration.

i=8,	j=0	8,0
i=8	j=1	8,1
i=8	j=2	8,2
i=8	j=3	8,3
i=8	j=4	8,4
i=8	j=5	8,5
i=8	j=6	8,6
i=8	j=7	8,7

Time complexity :-

outer loop = O(log n) ∴ $i = 2^k$

inner loop = O(n)

O(n) × O(log n) :

O(n log n)

DOMS | Page No.
Date / /

9) Dry run for n = 20 How many recursive calls & what values are printed.

```
void recHalf (int n) {
    if (n<=0) return;
    System.out.println(n+" ");
    recHalf (n/2);
}
```

→ first check n = 20
⇒ 20<=0 false
function call
recHalf (20/2) = 10

∴ 2nd call
n = 10
check, 10<=0, false
functioncall
rechalf(10/2) = 5

∴ 3rd call
n = 5
check 5<=0 false.
functioncall
recHalf (5/2) = 2

4th call
n = 2
check 2<=0 False
∴ recersive call
recHalf (2/2) = 1

5th call
n = 1
check 1<=0 false
recursive call
recHalf(1/2); = 0

6th call
n = 0 true
~~print~~
returns to previous call

Print		Print
	20 > 0	20
	10 > 0	10
	5 > 0	5
	2 > 0	2
	1 > 0	1
	0 <= 0	

∴ Total Recursive call = 6
it ~~Printin~~ ~~ting~~ Printing - 20, 10, 5, 2, 1

4) Dry run For n = 3. How many total call are made? what's the time complexity?

```
void fun (int n){
    if (n == 0) return
    fun (n-1);
    fun(n-1);
}
```

→ first check fun(3)
n = 3
(3 == 0) false

recursive call
fun(3-1) = 2
call fun(2)
call fun(2)
∴ 2nd call fun(2)
n = 2
if (2 == 0) false
fun(2-1) = 1
call fun(1)
call fun(1)
∴ 3rd call fun(1)
n = 1
if (1 == 0) false
call fun(1-1) = 0
call fun(0)
4th call
n = 0
if (0 == 0) true
call fun(0), call fun(0)

Total number of function call fun(n) is $2^{(n+1)} - 1$

$n = 3$
$= 2^{(3+1)} - 1$
$= 2^4 - 1$
$= 16 - 1$
$= 15$

Time Complexity:

2^n

$1 + 2 + 4 + 8 \cdots + 2^n$

$2^{n+1} - 1$

Time complexity = $O(2^n)$

5) Dry run for n=3. How many total iterations? Time complexity?

```
void tripleNested (int n){
    for(int i=0; i<n; i++)
        for (int j=0; j<n; j++)
            for( int k=0; k<n; k++)
                System.out.println(i+j+k);
}
```

→ 1st iteration.

i	j	k	Print
0	0	0	0

~~2nd iteration~~

i	j	k.	
0	0	1	1
~~3rd~~ 0	0	2	2.
~~4th~~ 0	1	0	1
0	1	1	2
0	1	2	3
0	2	0	2
0	2	1	3
0	2	2	4

2nd iteration

~~i=1~~

2nd iteration.

i	j	k	Print
1	0	0	1
1	0	1	2
1	0	2	3
1	1	0	2
1	1	1	3
1	1	2	4
1	2	0	3
1	2	1	4
1	2	2	5
2	0	0	2
2	0	1	3
2	0	2	4
2	1	0	3
2	1	1	4
2	1	2	5
2	2	0	4
2	2	1	5
2	2	2	6

Time complexity:

Each loop runs n time

$n = 3 \quad 3^3 = 27$ times.

$O(n^3)$