

Problem – 5: Dry Run & Analyse: Time and Space Complexity

1. Dry run the code for $n = 4$. How many times is `*` printed? What is the time complexity?

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

Ans: -

Dry Run:

```
i=0, j=0  
*  
i=0, j=1 //false  
*  
i=1, j=0  
*  
*  
i=1, j=1  
*  
**  
i=1, j=2 //false  
*  
**  
i=2, j=0  
*  
**  
*  
i=2, j=1  
*  
**  
**  
i=2, j=2  
*  
**  
***
```

```
i=2, j=3 //false  
*  
**  
***  
i=3, j=0  
*  
**  
***  
*  
i=3, j=1  
*  
**  
***  
**  
i=3, j=2  
*  
**  
***  
***  
i=3, j=3  
*  
**  
***  
****  
i=3, j=4 //false  
*  
**  
***  
****
```

Stars Printed: 10 stars are printed as inner loop runs $(n(n+1))/2$ times across all iterations.

Time Complexity:

Outer loop runs n times and inner loop runs $(n(n+1))/2$ across all iterations times which lead to time complexity of **$O(n^2)$** .

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

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

Ans: -

Dry run:

Outer loop iterations	Inner loop iterations
1, i=1	j=0, j=1, j=2, j=3, j=4, j=5, j=6, j=7
2, i=2	j=0, j=1, j=2, j=3, j=4, j=5, j=6, j=7
3, i=4	j=0, j=1, j=2, j=3, j=4, j=5, j=6, j=7
4, i=8	j=0, j=1, j=2, j=3, j=4, j=5, j=6, j=7
5, i=16	//False

Number of Iterations: 32

Time Complexity: The number of iterations of the outer loop is approximately $\log_2 n + 1$ times. For each iteration of the outer loop, the inner loop runs exactly n times. Therefore, time complexity of this program: **$O(n \log n)$** .

3. Dry run for $n = 20$. How many recursive calls? What values are printed?

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

Ans: -

Dry Run:

```
n=20 20  
n=10 20 10  
n=5 20 10 5  
n=2 20 10 5 2  
n=1 20 10 5 2 1  
n=0 //False  
  
Final Output:  
20 10 5 2 1
```

Number of recursive calls: 6

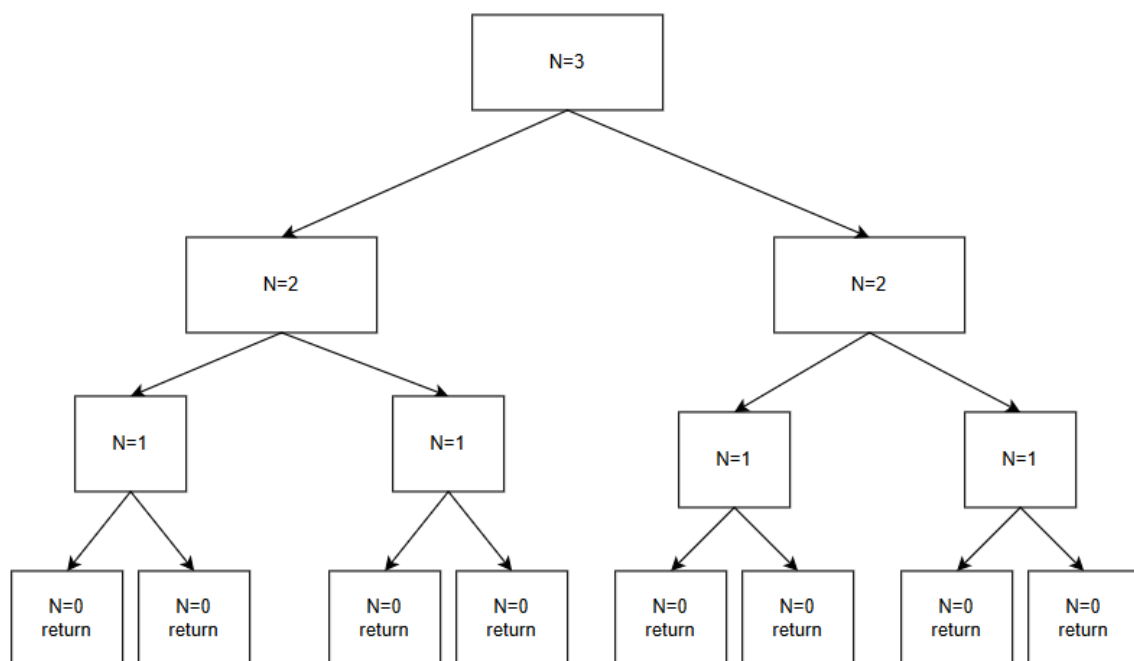
Time Complexity: $O(\log_2 n)$

4. Dry run for $n = 3$. How many total calls are made? What's the time complexity?

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

Ans: -

Dry Run:



Number of recursion calls made: 8

Time Complexity: This recursive function makes two recursive calls at each step, resulting in exponential growth. The time complexity is $O(2^n)$ because at each level, the number of calls doubles.

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);  
}
```

Ans: -

Dry Run:

```
i=0 j=0 k=0  
      k=1  
      k=2  
i=0 j=1 k=0  
      k=1  
      k=2  
i=0 j=2 k=0  
      k=1  
      k=2  
i=1 j=0 k=0  
      k=1  
      k=2  
i=1 j=1 k=0  
      k=1  
      k=2
```

```
i=1 j=2 k=0  
      k=1  
      k=2  
i=2 j=0 k=0  
      k=1  
      k=2  
i=2 j=1 k=0  
      k=1  
      k=2  
i=2 j=2 k=0  
      k=1  
      k=2
```

Number of Total iterations: 27

Time Complexity: Outer loop executes n times, 1st inner loop runs n*n i.e. n^2 times and 2nd inner loop executes n*n*n i.e. **$O(n^3)$** .