Introduction to Complexity Analysis of Non-Recursive Algorithms

Session No.: 6
Course Name: Design and analysis of algorithm
Course Code: R1UC407B
Instructor Name: Mili Dhar
Duration: 50 Min.
Date of Conduction of Class:

GALGOTIAS UNIVERSITY

Review of the key concepts

1. Complexity Analysis
2. Types of Complexity
3. Analyzing Non-Recursive Algorithms

Q: Time complexity to find the second largest element in an array?

Learning Outcome

Analyze time and space complexity for non-recursive algorithms.

Identify best, worst, and average case scenarios.

Session Outline

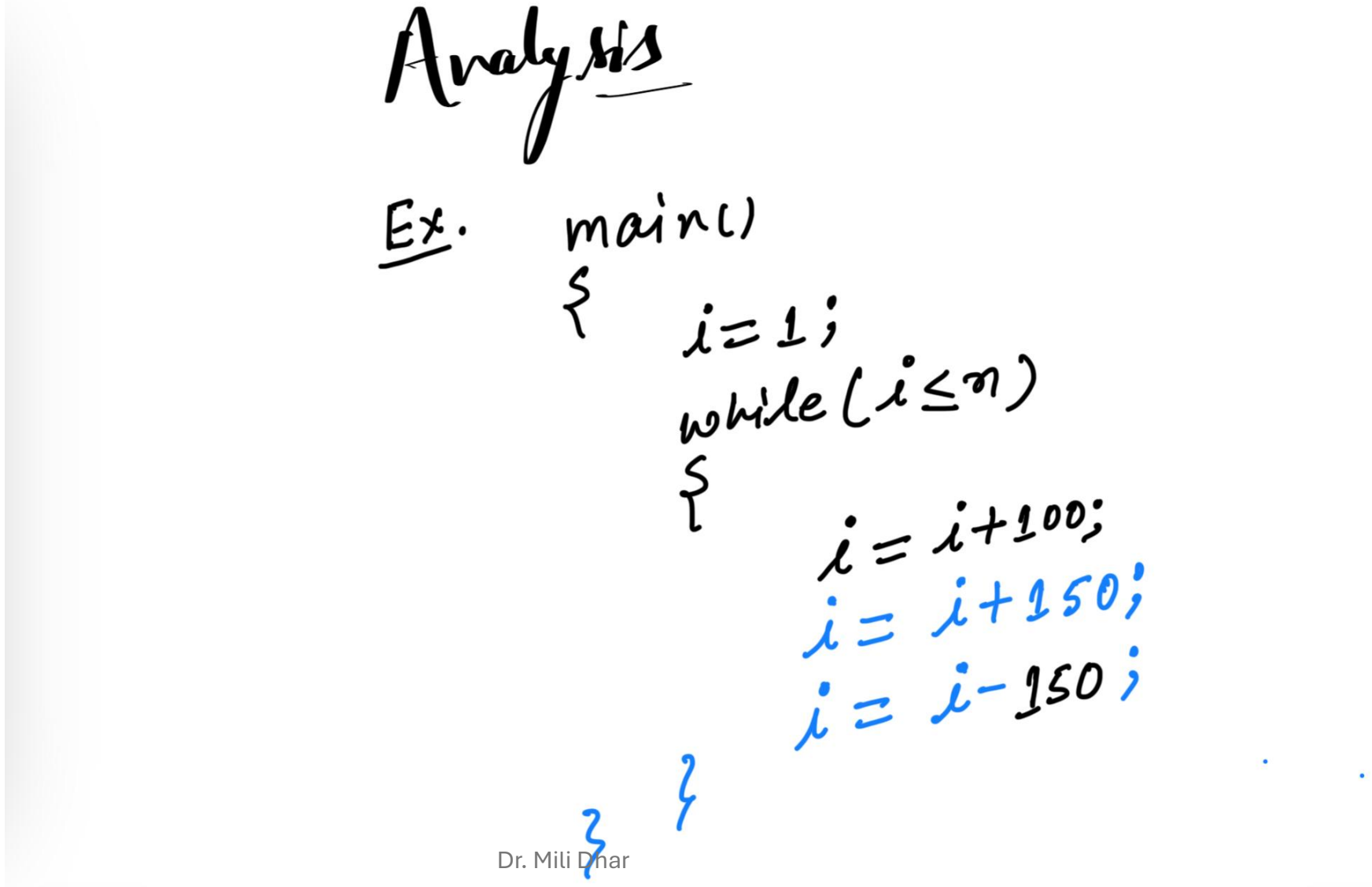
1 Complexity Analysis of Non-Recursive Algorithms

2 Reflection learning activity

3 Conclusion and post-session activity

Complexity Analysis of Non-Recursive Algorithms

Activity1



Analysis

Ex.

```
main()
{
    while (n>1)
    {
        n=n/2;
    }
}
```

Analysis

Ex.

```
main()
{
    while (n>20)
    {
        n = n/20;
    }
}
```

Activity 1 (Wooclap)

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code
LPUQSB

Activity 1 (Wooclap)

Analysis Ex.

```
main()
{
    while (n>1)
    {
        n = n/20;
        n = n/3;
    }
}
```

Activity 2

Analysis Ex.

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

Activity 2

Analysis

Ex.

```
main()
{
    i=1;
    while(i<n)
    {
        i=20*i;
    }
}
```

Activitv 2

Analysis

Ex.

```
main()
{
    i=1;
    while(i<n)
    {
        i=2*i;
        i=5*i;
    }
}
```

Activity 2 (Wooclap)

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code
LPUQSB

Activity 2 (Wooclap)

Analysis

Ex.

```
main()
{
    i=1;
    while(i<n)
    {
        i=2*i;
        i=5*i;
        i=10*i;
        i=i/25;
    }
}
```

Activity 2

Analysis Ex.

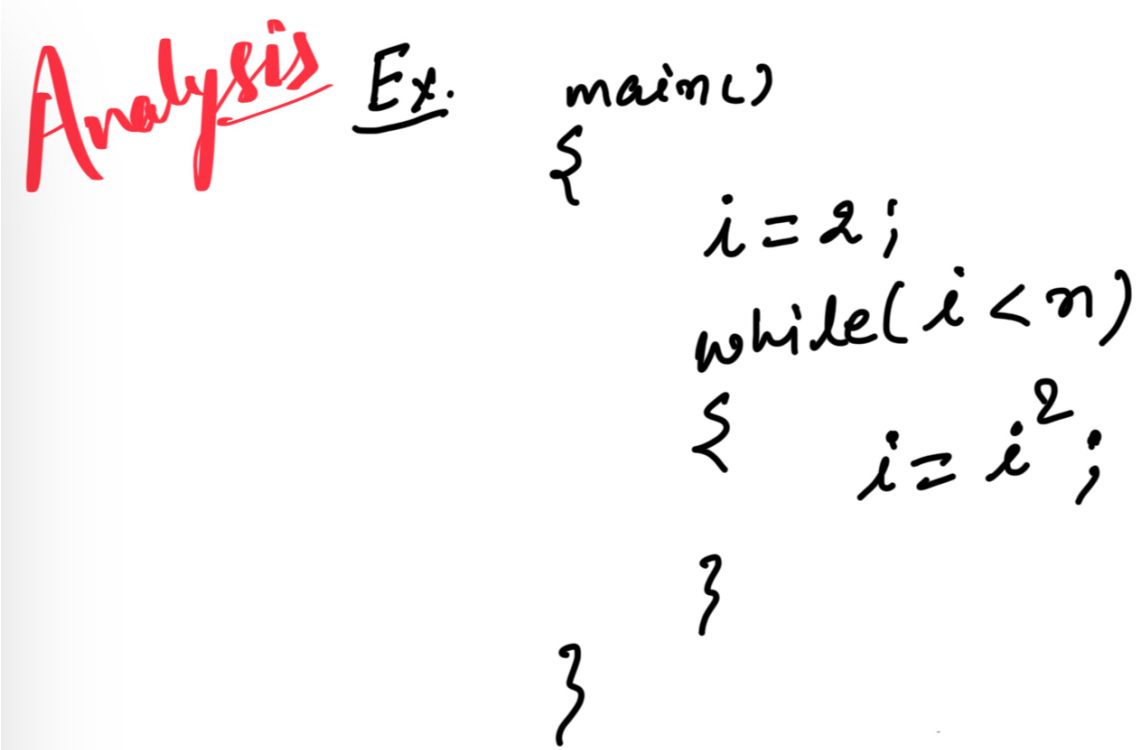
```
main()
{
    while(n>2)
    {
        n = n^1/2;
    }
}
```

Activity 2

Analysis Ex.

```
main()
{
    while(n>29)
    {
        n = n^(1/25);
    }
}
```

Activity 2



Activity 2

Analysis

Ex.

```
int main()
{
    p=1;
    for(i=1; i<n; i=2*i)
    {
        p++;
    }
    q=1;
    for(j=1; j<p; j++)
    {
        q++;
    }
    return q;
}
```

q - value ?

Activity 2

Analysis

Ex.

```
int main()
{
    p=1;
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j=5*j)
        {
            p=p+n;
        }
    }
}
```

p - value ?

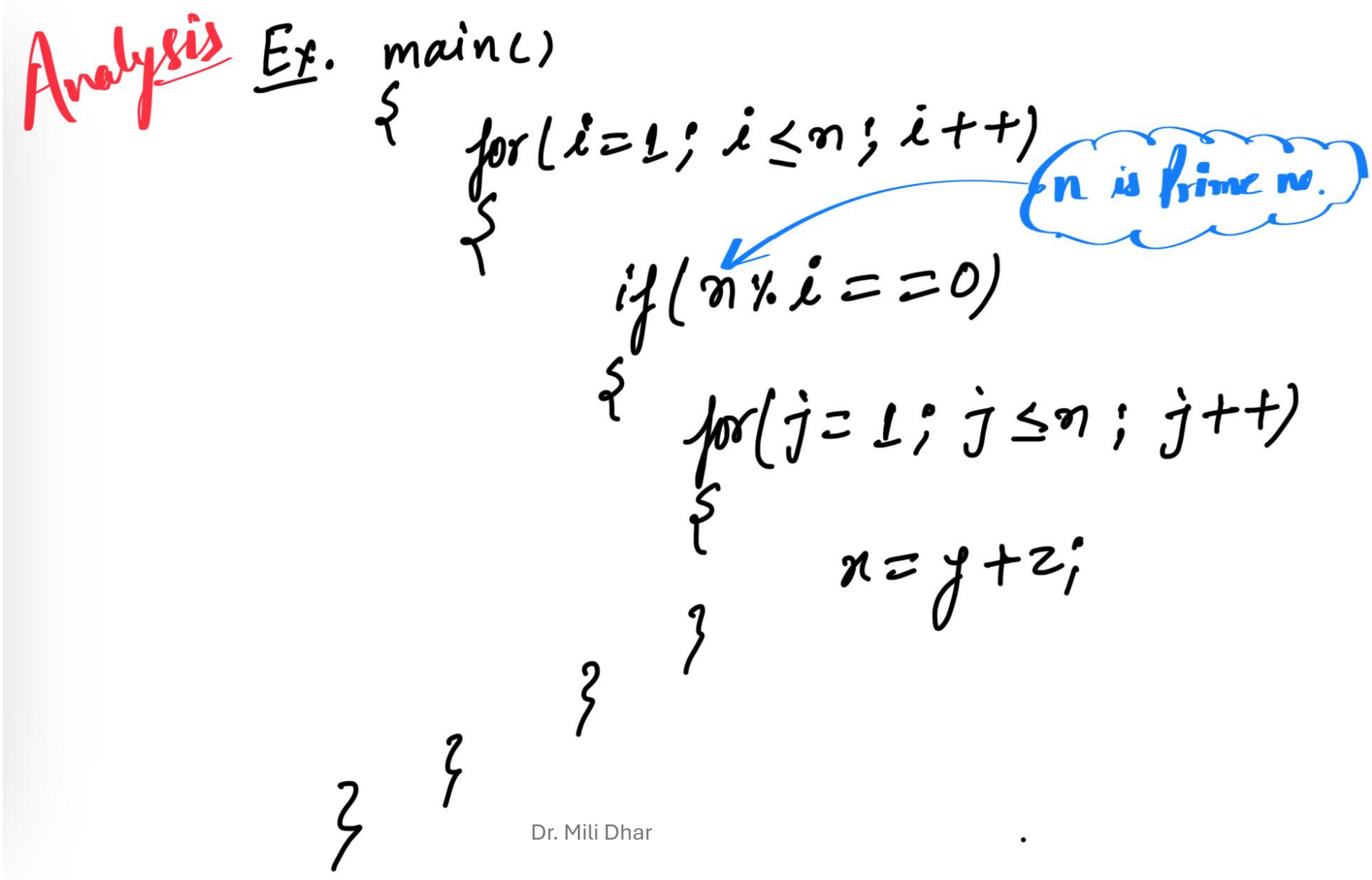
Activity 2

Analysis Ex.

```
main()
{
    for(i=10; i<n; i=i^5)
    {
        for(j=n; j>1; j=j/100)
        {
            for(k=1; k<n^7; k=k+9)
            {
                x=y+z;
            }
        }
    }
}
```

T.C = ?

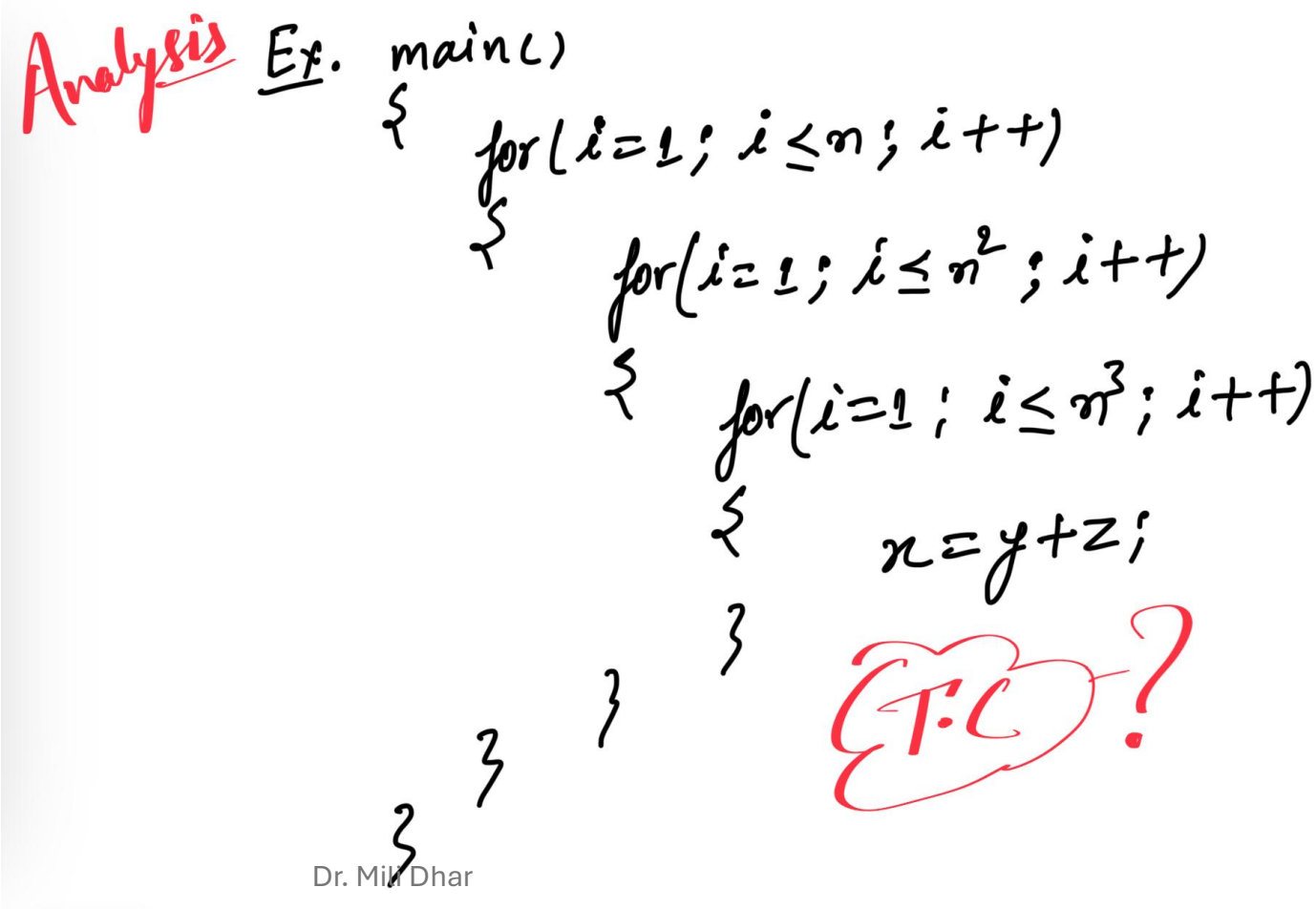
Activity 2



Summery

1. Dividing Loops (e.g., i = i/2) ➞ O(log(n))
2. Multiplicative Loops (e.g., i = i * k) ➞ $O(\log_k(n))$
3. Convert loop execution counts into summation formulas and simplify.
4. Nested Loops: Multiply complexities of inner and outer loops. **Example: $O(n^2)$**
5. Simple Loops: Run a fixed number of times based on n. **Example: O(n)**

Post session activities



In the next session, Analysis of Recursive Algorithms will be discussed in detail.