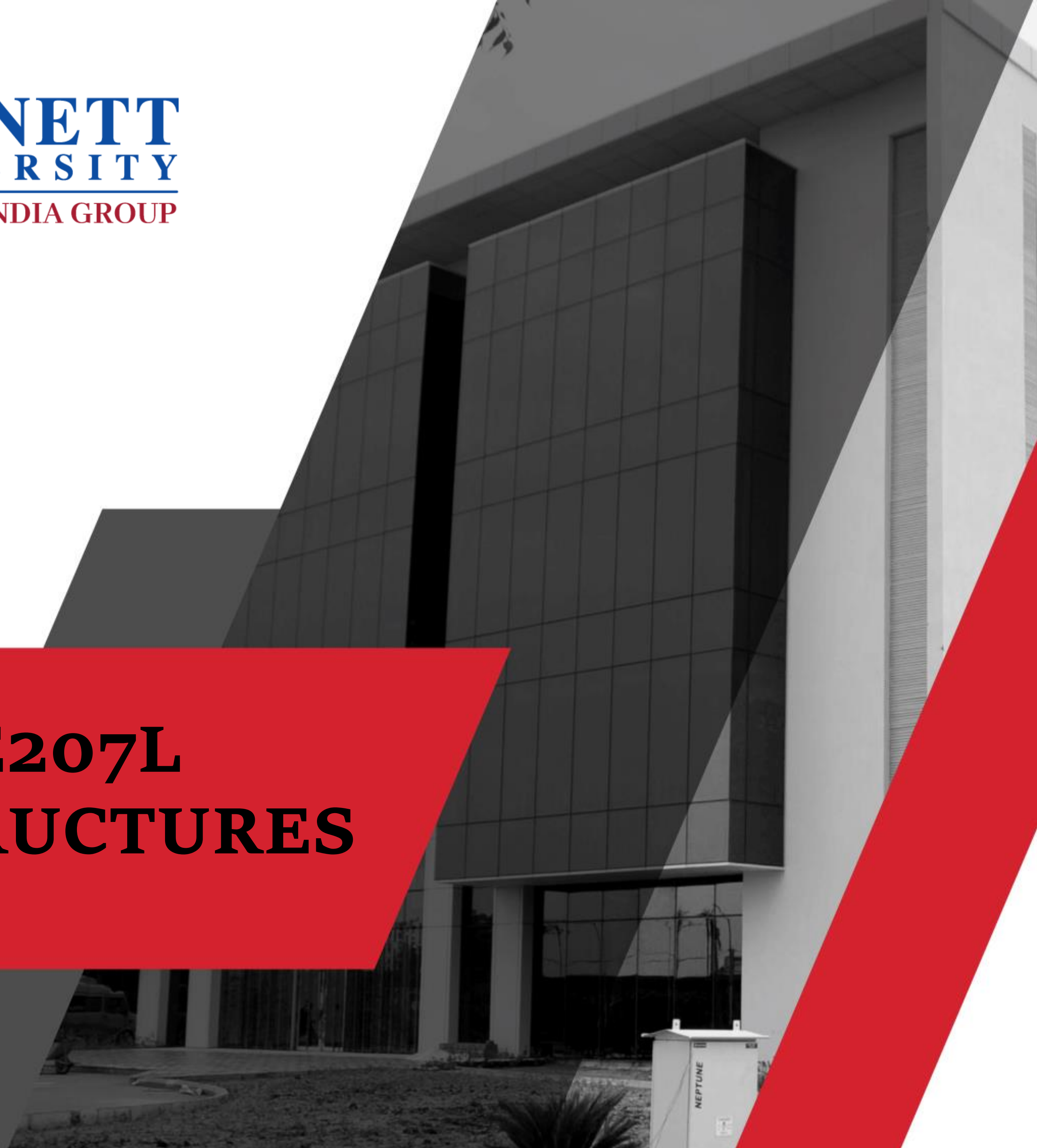
BENNETT UNIVERSITY
TIMES OF INDIA GROUP



ECSE207L DATA STRUCTURES

Dr. Tapas Badal
Dept. of CSE
Bennett University

Tutorial 2

1. **What are differences between performance and complexity?**

O(1)

O(1) represents an algorithm that takes the same amount of time to execute regardless of the number of inputs. So, 1 item takes 1 second, 10 items take 1 second, 100 items take 1 second and so on. Therefore, performance is not affected by the size of the input data.

O(N)

O(N) represents an algorithm where the size of the input data impacts the execution time. The performance of the algorithm is directly proportional to the number of inputs. So, 1 item takes 1 second, 10 items take 10 seconds, 100 items take 100 seconds and so on.

O(log N)

O(log N) represents an algorithm where the number of computations grows linearly as input data grows exponentially. So 1 item takes 1 second, 10 items take 2 seconds, 100 items take 3 seconds and so on.

O(2^N)

O(2^N) represents an algorithm where execution time is doubled for each additional input. So, 1 item takes 2 seconds, 2 items take 4 seconds, 3 items take 8 seconds and so on.

O(N!)

O(N!) represents a factorial algorithm that must perform N! calculations. So, 1 item takes 1 second, 2 items take 2 seconds, 3 items take 6 seconds and so on. An example of this algorithm is one that recursively calculates Fibonacci numbers.

O(N2)

O(N2) represents an algorithm that is directly proportional to the square of the sizes of the inputs and must performs N2calculations (by definition). Bubblesort is a good example of this algorithm.

O(N log N)

(N log N) represents an algorithm that will in increase in execution time proportionate to the number of the input times the logarithm of the number of the input. Mergesort and quicksort are good examples of this algorithm.

1. **Find the time complexity of given expression?**
   a. $3n^2 + 5n + 6$
   b. $10n + 2\,n\log n + 4\log n$
   c. $6\log n + n$
   d. $3n\log n + 2n$
   e. $8\log n + 4\log\log n$
   f. $3n + 2(\log n)^2 + 4\log n$

```
void printFirstElementOfArray(int arr[])
{
printf("First element of array = %d",arr[0]);
}
```

```
void printAllElementOfArray(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d\n", arr[i]);
    }
}
```

```
void printAllPossibleOrderedPairs(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            printf("%d = %d\n", arr[i], arr[j]);
        }
     }
}
```

```
int fibonacci(int num)
{
    if (num <= 1) return num;
    return fibonacci(num - 2) + fibonacci(num - 1);
}
```

```
void printAllItemsTwice(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d\n", arr[i]);
    }

    for (int i = 0; i < size; i++)
    {
        printf("%d\n", arr[i]);
    }
}
```

```
void printFirstItemThenFirstHalfThenSayHi100Times(int arr[], int size)
{
    printf("First element of array = %d\n",arr[0]);

    for (int i = 0; i < size/2; i++)
    {
        printf("%d\n", arr[i]);
    }

    for (int i = 0; i < 100; i++)
    {
        printf("Hi\n");
    }
}
```

```
void printAllNumbersThenAllPairSums(int arr[], int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d\n", arr[i]);
    }

    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            printf("%d\n", arr[i] + arr[j]);
        }
    }
}
```

$O(n^3 + 50n^2 + 10000)$

$O((n + 30) * (n + 5))$

```
bool arrayContainsElement(int arr[], int size, int element)
{
    for (int i = 0; i < size; i++)
    {
        if (arr[i] == element) return true;
    }
    return false;
}
```

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

```
int sum =0;
for(int i = 0; i < n; i+=2)
        {
                for(int j = 0; j <  n; j+=10)
                {
                        sum++;
                }
        }
```

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

```
int sum =0;
        for(int i = 0; i < n; i++)
        {
                for(int j = 0; j < m; j++)
                {
                        sum++;
                }
        }
```

```
int sum =0;
        for(int i = 1; i < n; i+=2)
        {
                for(int j = 1; j <  n; j*=2)
                {
                        sum++;
                }
        }
        for(int k = n; k >= 1; k--)
        {
                sum++;
        }
```

```
int sum =0;
        for(int i = 1; i <= n; i++)
        {
                for(int j = 1; j <=  i; j++)
                {
                        for(int k = 1; k <= 133; k++)
                        {
                        Sum++;
                        }
                }
        }
```

(n + 1)^3 is O(n^3)

f(n) = n Σ i=1 i is O(n^2).

Write formula for computing memory address in a 3D tensor.

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

THANKYOU

@csebennett cse_bennett