Newton School of Technology

3

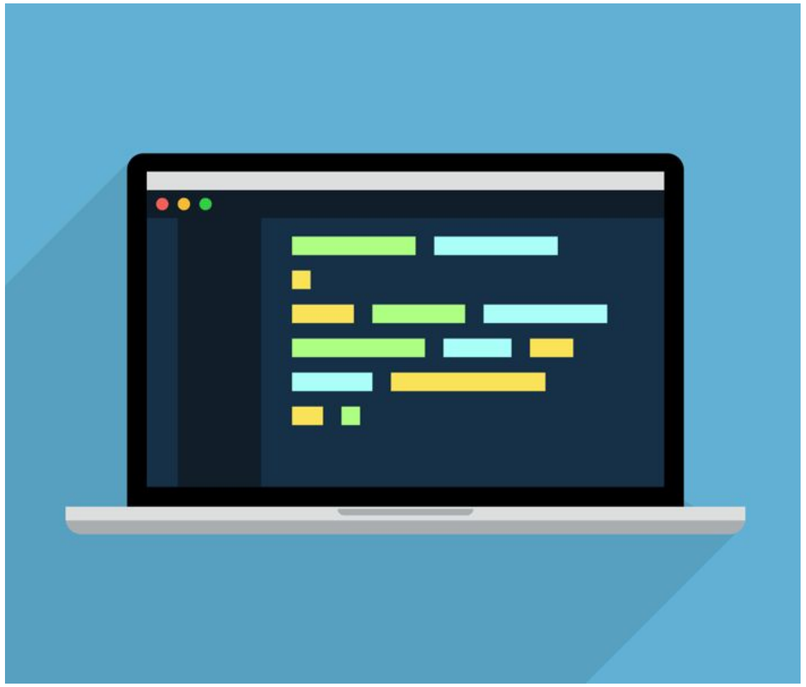
2D List + Problem Solving

Newton School of Technology

Join the lecture online on your dashboard

Quick Recap :

- 1D List Problem Solving
- Prefix sum Technique
- Two Pointers Approach
- Calculating number of operations
- Optimizing code

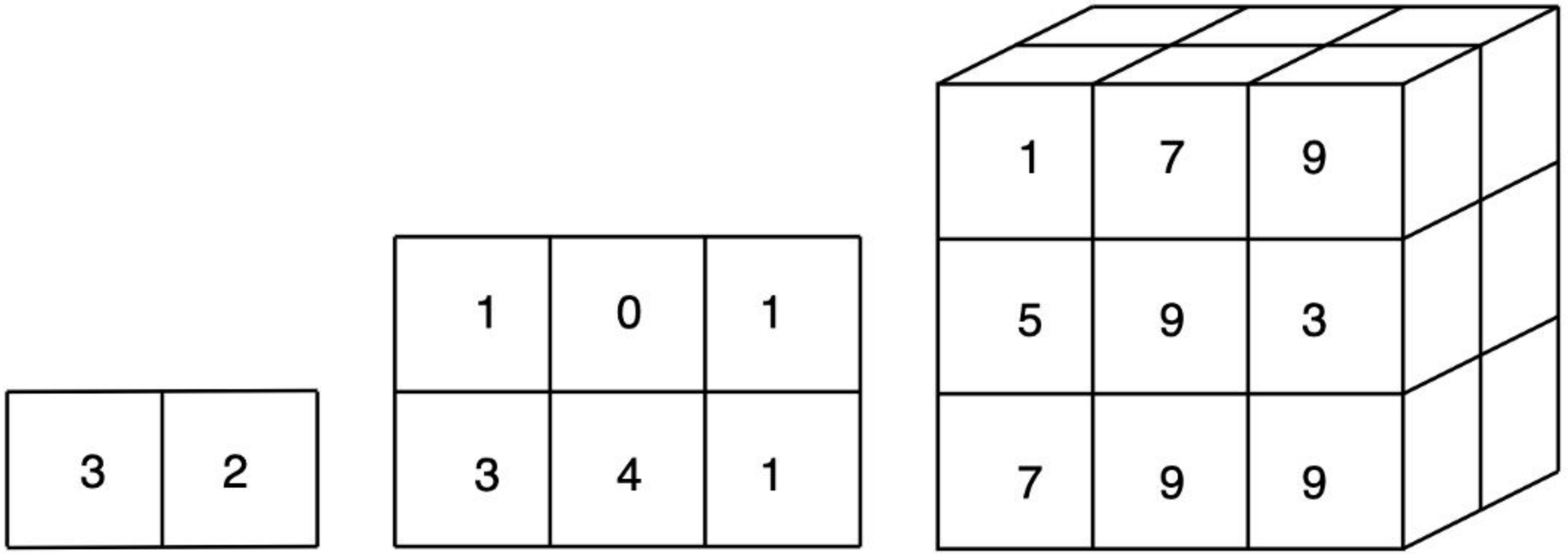


Newton School of Technology

Pre Read Quiz

1D, 2D and 3D Lists

Newton School of Technology



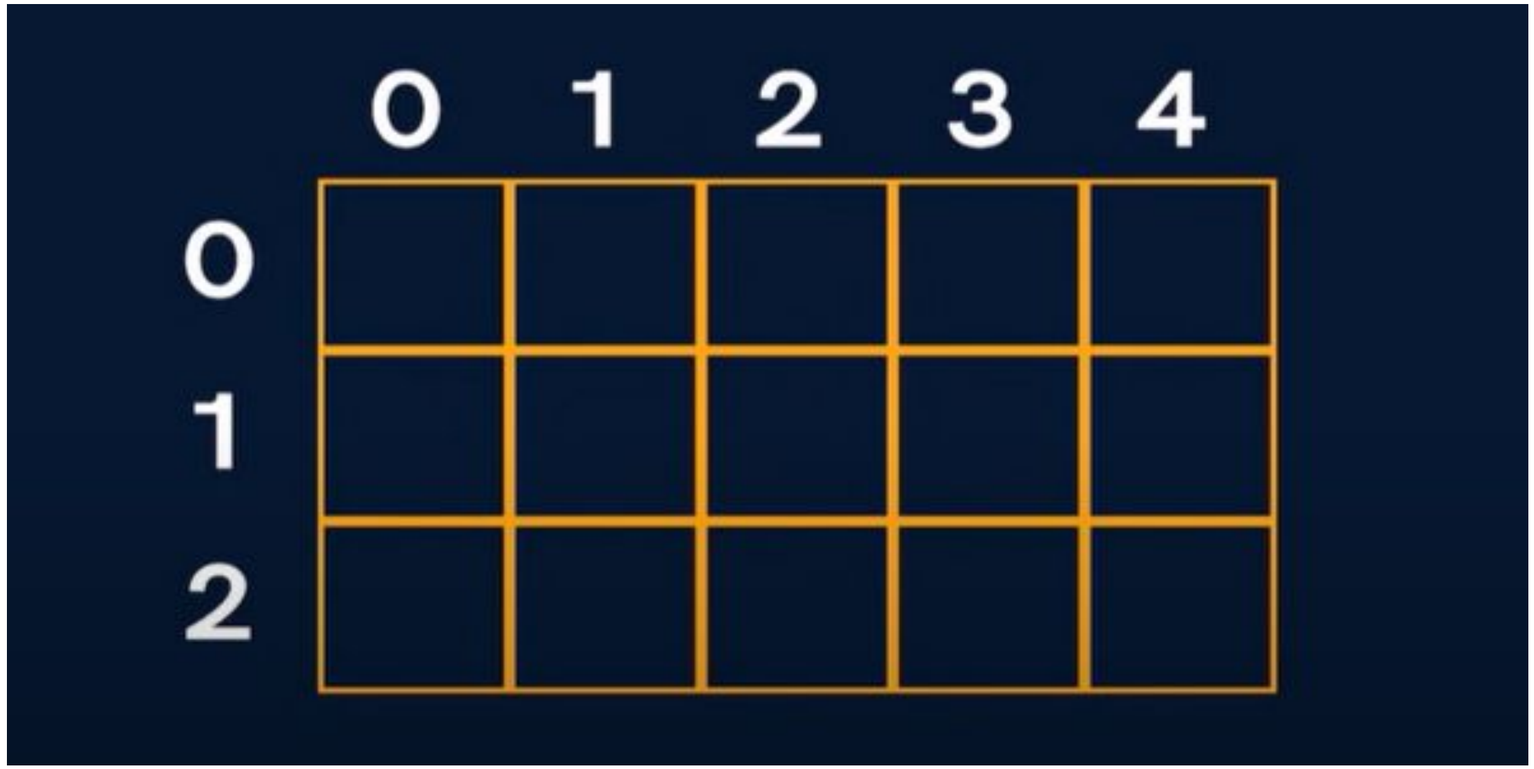
2D List –> List of List

Newton School of Technology

2	7	8
5	11	9
19	13	21
21	42	4

Indexing in 2D Array :

Newton School of Technology



Indexing starts from 0

Accessing Elements of 2D arrays:

Newton School of Technology

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Declaration of 2D Matrix :

Newton School of Technology

matrix

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

matrix [0][2] = ?

matrix [1][1] = ?

matrix[2][1] = ?

Declaration of 2D Matrix :

Newton School of Technology

matrix

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

matrix [0][2] = 3

matrix [1][1] = 5

matrix[2][1] = 8

Declaration of 2D Matrix :

Newton School of Technology

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
matrix = [
    [1, 2, 3] ,
    [4, 5, 6] ,
    [7, 8, 9]
    ]
```

Declaration of 2D Matrix with some initial value :

Newton School of Technology

```
rows = 3
cols = 4
```

Declaration of 2D Matrix with same initial value :

Newton School of Technology

```
rows = 3
cols = 4
array = [[0 for _ in range(cols)] for _ in range(rows)]
```

Declaration of 2D Matrix with same initial value :

Newton School of Technology

```
rows = 3
cols = 4
array = [[0 for _ in range(cols)] for _ in range(rows)]
```

```
[[0, 0, 0, 0],
 [0, 0, 0, 0],
 [0, 0, 0, 0]]
```

Taking input of 2D Matrix:

Newton School of Technology

Step 1 : Input number of Rows and Columns

```
rows = int(input())
cols = int(input())
```

Taking input of 2D Matrix:

Newton School of Technology

Step 2 : Input each row in Loop and append in array

```
for i in range(rows):
    row = list(map(int, input().split()))
    array.append(row)
```

Taking input of 2D Matrix:

Newton School of Technology

```
rows = int(input())
cols = int(input())

array = []

for i in range(rows):
    row = list(map(int, input().split()))
    array.append(row)
```

Problem 1 :

Newton School of Technology

Pixel Colors

Given a matrix A with n rows and m columns and a number k, calculate sum of all elements A[i][j] where i + j == k

SOLUTION:

Newton School of Technology

1. For each query, initialize the *sum* to *0*.
2. Iterate through all rows *i* and columns *j* of the matrix.
3. Check *if i+j=k*:
    a. *If true*, add the pixel value at index *(i,j)* to the *sum*.
4. After processing the entire matrix, *print* the result for the current query.
5. Repeat for all queries.

CODE:

Newton School of Technology

```python
M, N = map(int,input().split())
A = []
for _ in range(M):
    A.append(list(map(int,input().split())))

Q = int(input())

for _ in range(Q):
    query = int(input())

    sum1 = 0
    #check if i+j == k
    for i in range(M):
        for j in range(N):
            if i+j == query:
                sum1 += A[i][j]
    print(sum1)
```

SOLUTION: O(N)

Newton School of Technology

1. For each query, initialize the *sum* to *0*.
2. Iterate through all rows *i*:
   a. *A*dd the pixel value at index *(i, k-i)* to the *sum*.
3. After processing the entire matrix, *print* the result for the current query.
4. Repeat for all queries.

SOLUTION: O(N)

Newton School of Technology

```python
def precompute_diagonal_sums(matrix, M, N):
    diagonal_sum = {}

    # Compute the sum for each diagonal (i + j = k)
    for i in range(M):
        for j in range(N):
            k = i + j
            if k not in diagonal_sum:
                diagonal_sum[k] = 0
            diagonal_sum[k] += matrix[i][j]

    return diagonal_sum


# Input
M, N = map(int, input().split())  # Dimensions of the matrix
matrix = [list(map(int, input().split())) for _ in range(M)]  # Matrix elements
Q = int(input())  # Number of queries
queries = [int(input()) for _ in range(Q)]  # Queries

# Precompute diagonal sums
diagonal_sum = precompute_diagonal_sums(matrix, M, N)

# Answer each query in O(1)
for k in queries:
    print(diagonal_sum.get(k, 0))
```

CODE:

Newton School of Technology

Problem 2 :

Newton School of Technology

Matrix multiplication - I

3*3 matrix multiplication

Solution:

Newton School of Technology

Let's dry run the matrix multiplication:

1	2	3
4	5	6
7	8	9

10	20	30
40	50	60
70	80	90

0	0	0
0	0	0
0	0	0

Solution:

Newton School of Technology

`To calculate R[0][0]:`

1	2	3
4	5	6
7	8	9

10	20	30
40	50	60
70	80	90

0	0	0
0	0	0
0	0	0

Solution:

Newton School of Technology

`To calculate R[0][0]:`

1	2	3
4	5	6
7	8	9

10	20	30
40	50	60
70	80	90

10	0	0
0	0	0
0	0	0

Solution:

Newton School of Technology

`To calculate R[0][0]:`

1	2	3
4	5	6
7	8	9

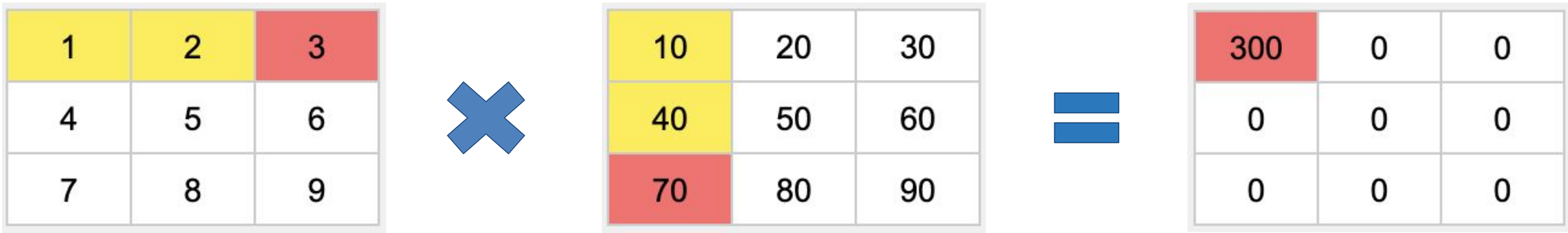
10	20	30
40	50	60
70	80	90

90	0	0
0	0	0
0	0	0

Solution:

Newton School of Technology

`To calculate R[0][0]:`



Calculating R[0][0]: Multiplying **Row with index 0** of **A** by **Column with index 0** of **B**

A[0][0] * B[0][0] = 1 * 10 = 10

A[0][1] * B[1][0] = 2 * 40 = 80

A[0][2] * B[2][0] = 3 * 70 = 210

R[0][0] = 300

Solution:

Newton School of Technology

`To calculate R[0][1]:`

1	2	3
4	5	6
7	8	9

10	20	30
40	50	60
70	80	90

300	20	0
0	0	0
0	0	0

Solution:

Newton School of Technology

`To calculate R[0][1]:`

1	2	3
4	5	6
7	8	9

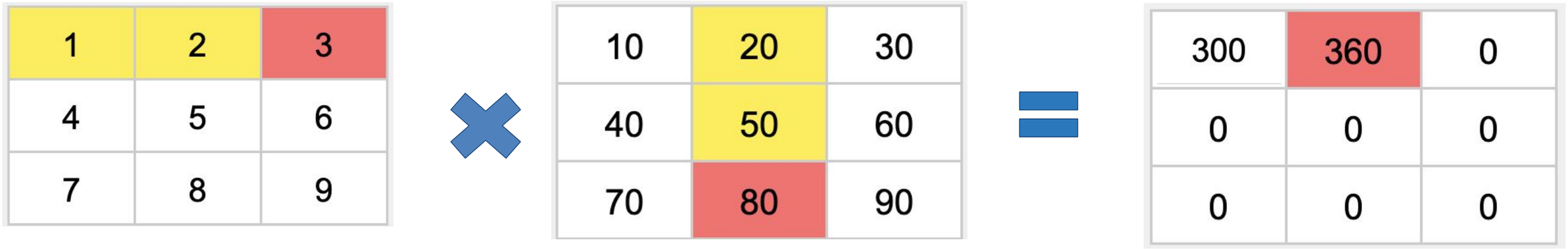
10	20	30
40	50	60
70	80	90

300	120	0
0	0	0
0	0	0

Solution:

Newton School of Technology

`To calculate R[0][1]:`



Calculating R[0][1]: Multiplying **Row with index 0** of **A** by **Column with index 1** of **B**

A[0][0] * B[0][1] = 1 * 20 = 20

A[0][1] * B[1][1] = 2 * 50 = 100

A[0][2] * B[2][1] = 3 * 80 = 240

R[1][2] = 360

Formalize the pattern

Newton School of Technology

$$R[i][j] = \sum_{k=0}^{2} A[i][k] \times B[k][j]$$

Where:

- $R[i][j]$ is the element in the resulting matrix R,
- $A[i][k]$ is the element from the i-th row of A,
- $B[k][j]$ is the element from the j-th column of B.

For calculating **R[i][j]**:

1. Pick **row** with **index i** of **A** and
2. Pick **column** with **index j** of **B**
3. Add the product of corresponding elements from ***A*** and ***B*** to result[i][j].

Can you code it now?

Newton School of Technology

Code:

Newton School of Technology

Problem 3 :

Newton School of Technology

Matrix multiplication - II

n*n matrix multiplication

Constraints:

- 1 <=N <= 100

Solution:

Newton School of Technology

The advantage of formalizing the pattern is that the same solution can be extended for variable matrix size.

3*3 Matrix multiplication

$$R[i][j] = \sum_{k=0}^{2} A[i][k] \times B[k][j]$$

n*n Matrix multiplication

$$R[i][j] = \sum_{k=0}^{n-1} A[i][k] \times B[k][j]$$

Code:

Newton School of Technology

```python
def multiply(A,B):
    #create one empty 2d 2d list of 3*3
    C = [[0 for i in range(N)] for i in range(N)]

    for i in range(N):
        for j in range(N):
            for k in range(N):
                C[i][j] += A[i][k] * B[k][j]

    return C

A = []
B = []
N = int(input())
for i in range(N):
    row = list(map(int,input().split()))
    A.append(row)

for i in range(N):
    row = list(map(int,input().split()))
    B.append(row)


C = multiply(A,B)

for row in C:
    for item in row:
        print(item, end=" ")
    print()
```

Further Read on Matrix Multiplication:

https://en.wikipedia.org/wiki/Strassen_algorithm

Homework:

Newton School of Technology

What if the matrix problem we did at the start (i+j==k) problem had 10^6 queries with other constraints remaining same?

The current solution will give tle. How can you solve it then?

Homework – 2:

Newton School of Technology

In the matrix problem we did at the start we had:

f(i, j) = i+j = k

What if f(i, j) was:

f(i, j) = i - j = k

f(i, j) = gcd(i, j) = k

f(i, j) = i ^ j = k

f(i, j) = i % j = k

DSA For Internships

Newton School of Technology

Newton School of Technology

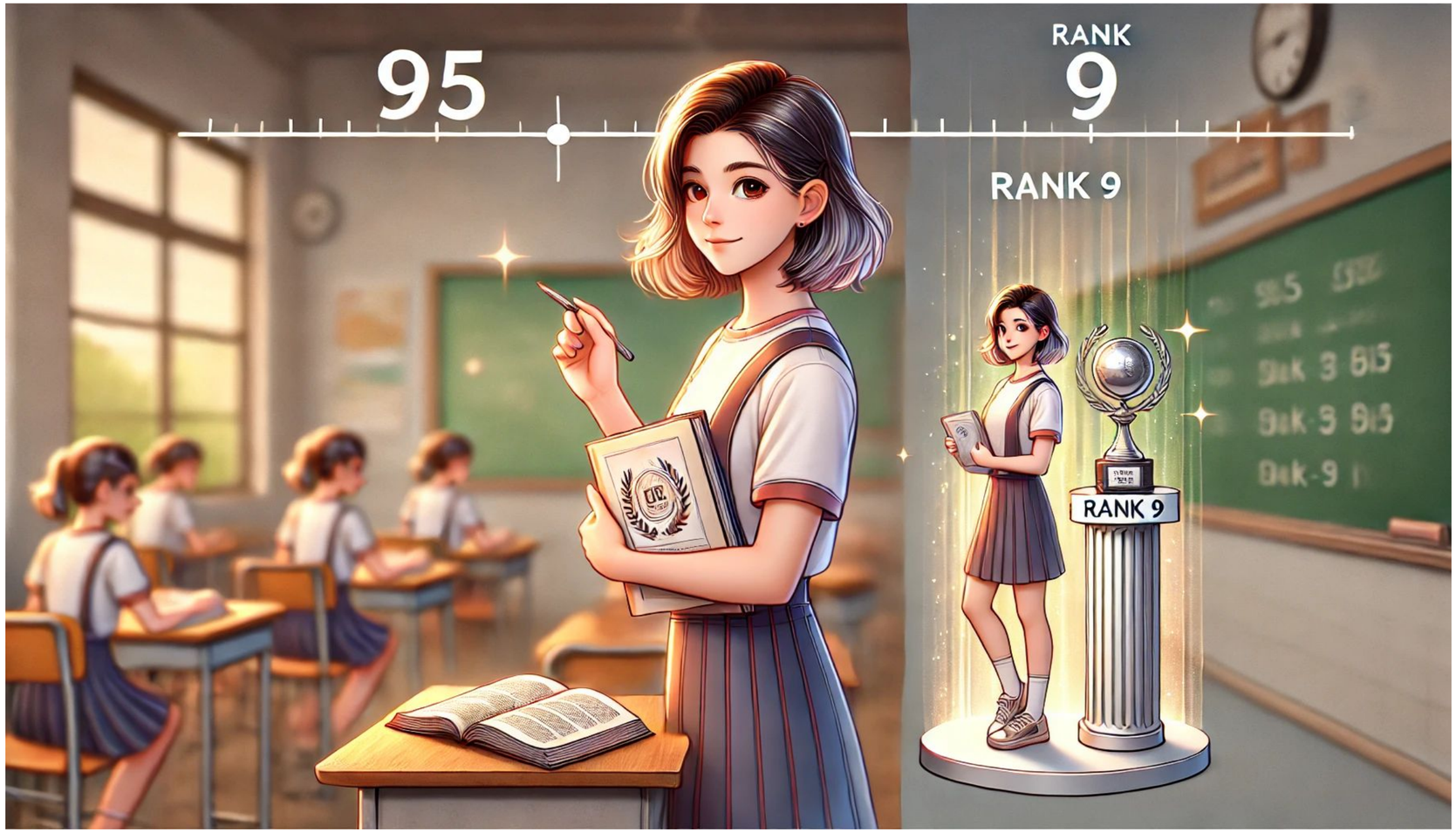
Know your Instructor

Your Instructor for Lectures :

Newton School of Technology

- **B.Tech in CSE** from LPU
- Technical Instructor **- UpGrad, Function Up, Newton School, Optus Edtech, Coding Spoon, Shikuyaa, Lara Technologies, takeUForward**
- **Designation - ** Technical Instructor & Program Manager
- Mentored **4000+** students across India
- Expertise : Data Structures & Algorithms

Can anyone learn DSA ?



Summary:

Newton School of Technology

- **2D List:**
  - **Taking Input**
  - **Indexing: Representing as rows and columns**
- **Matrix Multiplication**
  - **Understanding the pattern**
  - **Formalizing the pattern**
  - **Coding the solution**

Newton School of Technology

Summary Quiz

Thank You!