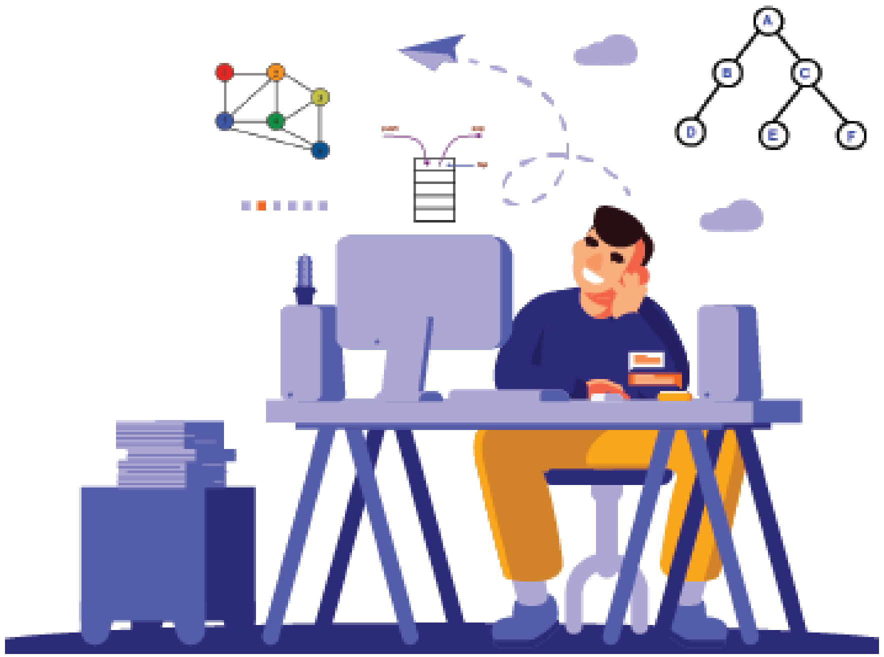
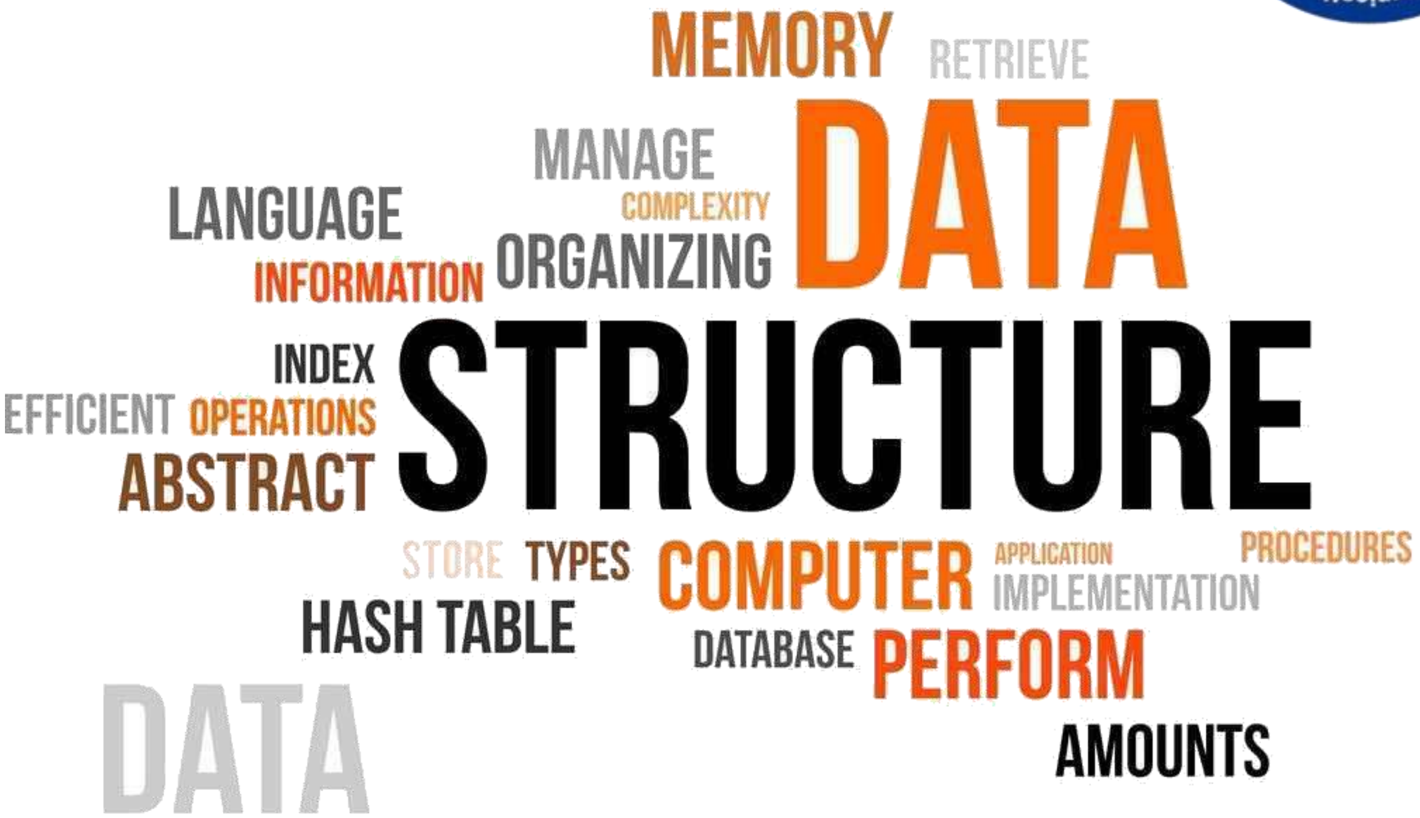
Data Structures

Course code: IT623





Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Matrices
Linked List

MATRICES

"Vectors" and "Matrices" are mathematical terms which refer to collection of numbers which are analogues, respectively to linear and two-dimensional arrays.

(a) An n- element vector V is a list of n number usually given in the form

$$\boxed{V = (V_1, V_2, \ldots, V_n)}$$

(b) An $m \times n$ matrix A is an array of $m \cdot n$ numbers arranged in $\boxed{m \text{ rows}}$ and $\boxed{n \text{ columns}}$ as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & & A_{mn} \end{bmatrix}$$

* In the context of vectors and matrices, the term scalar is used for individual numbers.

* A matrix with one row (column) may be viewed as a vector and, similarly, a vector may be viewed as a matrix with only one row (column).

* A matrix with the same number n of rows and columns is called a square matrix or an n-square matrix.

Algebra of Matrices →

* Suppose $\boxed{A}$ and $\boxed{B}$ are $\boxed{m \times n}$ matrices. The sum of A and B, written $\boxed{A+B}$ is the $\boxed{m \times n}$ matrix obtained by adding corresponding elements from A and B.

* The product of a scalar K and the matrix A, written $K \cdot A$, is the $m \times n$ matrix obtained by multiplying each element of A by K.

Some Simple Example :-

(a) Suppose

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 0 & 4 & 5 \end{bmatrix} \text{ and } B = \begin{bmatrix} 3 & 0 & -6 \\ 2 & -3 & 1 \end{bmatrix}$$

Then

$$A+B = \begin{bmatrix} 4 & -2 & -3 \\ 2 & 1 & 6 \end{bmatrix} \qquad 3A = \begin{bmatrix} 3 & -6 & 9 \\ 0 & 12 & 15 \end{bmatrix}$$

(b) Suppose $U = (1, -3, 4, 5)$, $V = (2, -3, -6, 0)$ and $w = (3, -5, 2, -1)$. Then:

Scalar product of two vector U and V is defined as

$$\boxed{U \cdot V = U_1 \cdot V_1 + U_2 \cdot V_2 + \cdots + U_n V_n = \sum_{K=1}^{n} U_K V_K}$$

$U \cdot V = ?$

$U \cdot W = ?$

Algorithm (Matrix Multiplication) MATMUL (A, B, C, M, P, N)

Let A be an M×P matrix array, and let B be P×N matrix array. This algorithm stores the product of A and B in an M×N matrix array C.

1. Repeat Steps 2 to 4 for I=1 to M:
2. Repeat steps 3 and 4 for J=1 to N:
3. Set C[I, J] = 0 [Initializes C[I, J]]
4. Repeat for K=1 to P:

   C[I, J] = C[I, J] + A[I, K] * B[K, J]

   [End of inner loop.]

   [End of step 2 middle loop.]

   [End of step 1 outer loop]

5. Exit.

* The complexity of a matrix multiplication algorithm is measured by counting the number C of multiplications.

* The reason that additions are not counted in such algorithms is that computer multiplication takes much more time than computer addition.

* The complexity of Algorithm for Matrix Multiplication is

$$\boxed{C = m \cdot n \cdot p}$$

↳ comes from step: 4 only.

Suppose A and B are 2×2 matrices. We have:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \qquad B = \begin{pmatrix} e & f \\ g & h \end{pmatrix} \qquad \text{and} \qquad AB = \begin{pmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{pmatrix}$$

In Algorithm 4.7, the product matrix AB is obtained using $C = 2 \cdot 2 \cdot 2 = 8$ multiplications. On the other hand, AB can also be obtained from the following, which uses only 7 multiplications:

$$AB = \begin{pmatrix} (1 + 4 - 5 + 7) & (3 + 5) \\ (2 + 4) & (1 + 3 - 2 + 6) \end{pmatrix}$$

1. $(a + d)(e + h)$
2. $(c + d)e$
3. $a(f - h)$
4. $d(g - e)$
5. $(a + b)h$
6. $(c - a)(e + f)$
7. $(b - d)(g + h)$

A **sparse matrix** is a matrix that contains a large number of zero elements compared to the total number of elements. In other words, it is a matrix that is mostly empty.

Sparse matrices are typically represented in a compressed format that only stores the non-zero elements and their positions. There are several different formats for storing sparse matrices, each with its own advantages and disadvantages.

Here are some of the most common formats:

1. **Coordinate Format (COO):** This format stores each non-zero element of the matrix along with its row and column indices. This format is simple and flexible, but it can be inefficient for large matrices with many non-zero elements.

2. Compressed Sparse Row (CSR) Format: This format stores the non-zero elements of each row in a separate array, along with the column indices of those elements.

It also stores an index array that indicates the start of each row in the data array.

This format is efficient for matrix-vector multiplication, which is a common operation in many numerical algorithms.

3. **Compressed Sparse Column (CSC) Format:** This format is similar to CSR, but it stores the non-zero elements of each column in a separate array.

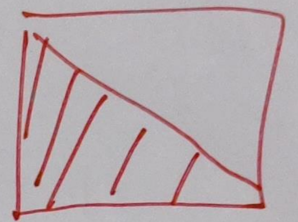
It is also efficient for matrix-vector multiplication, but it may be more difficult to construct than CSR.

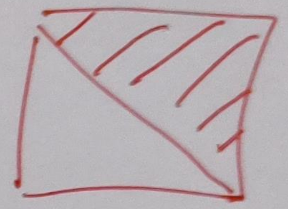
Sparse matrices can offer significant advantages in terms of memory usage and computational efficiency, especially for large-scale problems.

However, they can also require specialized algorithms and data structures to take advantage of their sparsity.

Two general types of n- square sparse matrix, which occur in various application are:

1) <u>Triangular matrix</u>
   - lower
   - upper





2) <u>Tridiagonal matrix</u>

The non-zero entries can only occur on the diagonal or on elements immediately above or below the diagonal is called a <u>tridiagonal matrix</u>.

$$
\begin{pmatrix}
4 & & & & \\
\underset{(21)}{3} & \underset{(22)}{-5} & & & \\
1 & 0 & 6 & & \\
-7 & 8 & -1 & 3 & \\
5 & -2 & 0 & 2 & \underset{(55)}{-8}
\end{pmatrix}
$$

(a) Triangular matrix

$$
\begin{pmatrix}
\overset{(1,1)}{5} & \overset{(1,2)}{-3} & & & & & \\
\underset{(2,1)}{1} & \overset{(2,2)}{4} & \overset{(2,3)}{3} & & & & \\
 & 9 & -3 & 6 & & & \\
 & & 2 & 4 & -7 & & \\
 & & & 3 & -1 & 0 & \\
 & & & & 6 & -5 & 8 \\
 & & & & & 3 & -1
\end{pmatrix}
$$

(b) Tridiagonal matrix

LINKED LISTS

→ "List" refers to a linear collection of data items.

→ Data processing frequently involves storing and processing data organized into lists.

→ One way to store such data is by means of arrays.
↳ linear relationship between the data element.

→ Another way of storing a list in memory is to have each element in the list contain a field, called ~~a~~ link or pointer, which contains the address of next element.

→ Thus, successive elements in the list need not occupy adjacent space in memory.

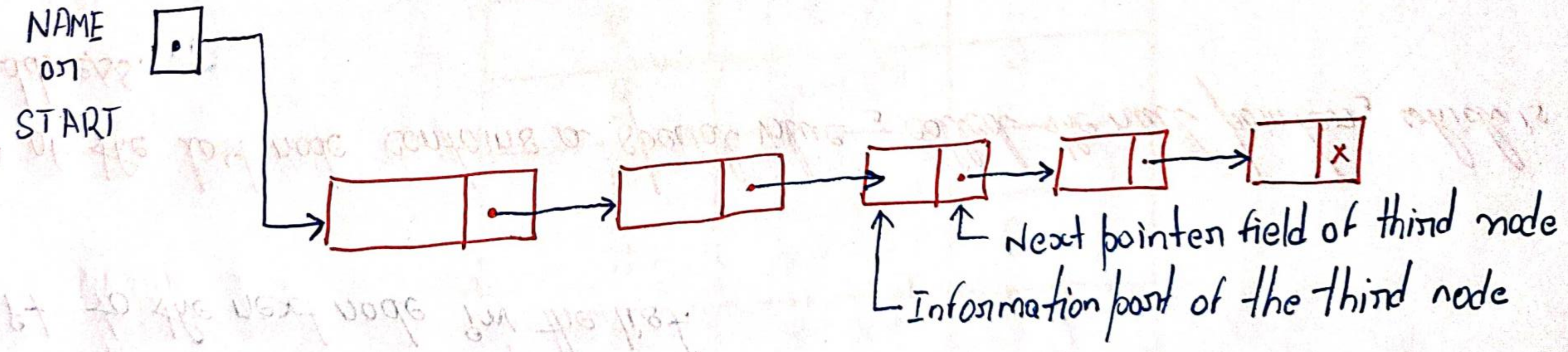
→ Making easier to insert and delete elements in the list.

⇒ A linked-list, or one-way list, is a linear collection of data elements, called nodes, where the linear order is given by means of pointers.

⇒ Each node is divided into two parts:

a> the first part contains the information of the element

b> the second part called the link field or next pointer field contains the address of the next node.



* The left part represents the information part of the node, which may contain an entire record of data item.

* The right part represents the next pointer field of the node, and there is an arrow drawn from it to the next node in the list.

* The pointer of the last node contains a special value, called the null pointer, which is any invalid address.

* $\boxed{\text{"0" or a negative number is used for a null pointer.}}$

* The linked list also contains a list pointer variable called START or NAME which contains the address of first node in the list.

* It implies we only need the address of START or NAME node to access entire list

* Hospital example on the next slide.

Example

START: 6

Bed Number	Patient	Next
1	Kirk	7
2		
3	Maxwell	11
4	Adams	12
5		
6	Lane	3
7	Green	4
8	Grece	1
9	Samuels	0
10		
11	Fields	8
12	Nelson	9
13	Rohan	2

Information (Bed Number, Patient) — Next pointer (Next)