Data Structure and Algorithms

Lecture 3 : Array

Things to cover

- Multidimensional Array:
  - Initialization
  - Pass array to a function
  - Memory storage
  - Operations : Transpose
  - Sparse Matrix

Initialization

- int arr[2][3];
- int arr[2][3] = {0,1,2,3,4,5};
- int arr[2][3] = {{0,1,2},{3,4,5}};

```
#include <stdio.h>

int main() {

    // Create and initialize an array with 3 rows
       // and 2 columns
    int arr[3][2] = { { 0, 1 }, { 2, 3 }, { 4, 5 } };

    // Print each array element's value
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
            printf("arr[%d][%d]: %d     ", i, j, arr[i][j]);
        }
       printf("\n");
    }
    return 0;
}
```

Passing Array to a function

```c
#include <stdio.h>

// Funtion that takes 2d array as parameter
void print(int arr[3][3], int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++)
            printf("%d ", arr[i][j]);
        printf("\n");
    }
}

int main() {
    int arr[3][3] = {{1, 2, 3},
                     {4, 5, 6},
                     {7, 8, 9}};

    // Passing the array along with the
    // size of rows and columns
    print(arr, 3, 3);

    return 0;
}
```

int arr[][3]

As Pointer to an Array : int (*arr)[3]

Passing Array to a function

```c
#include <stdio.h>

// Funtion that takes 2d array as parameter
void print(int arr[3][3], int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++)
            printf("%d ", arr[i][j]);
        printf("\n");
    }
}

int main() {
    int arr[3][3] = {{1, 2, 3},
                     {4, 5, 6},
                     {7, 8, 9}};

    // Passing the array along with the
    // size of rows and columns
    print(arr, 3, 3);

    return 0;
}
```

int arr[][3]

As Pointer to an Array : int (*arr)[3]

As Single Level Pointer : int *arr

print((int *)arr, 3, 3);

How Stored in memory

- Row Major Order

Address of A[I][J] = B + W * ((I – LR) * N + (J – LC))

I = Row Subset of an element whose address to be found,

J = Column Subset of an element whose address to be found,

B = Base address,

W = Storage size of one element store in an array(in byte),

LR = Lower Limit of row/start row index of the matrix(If not given assume it as zero),

LC = Lower Limit of column/start column index of the matrix(If not given assume it as zero),

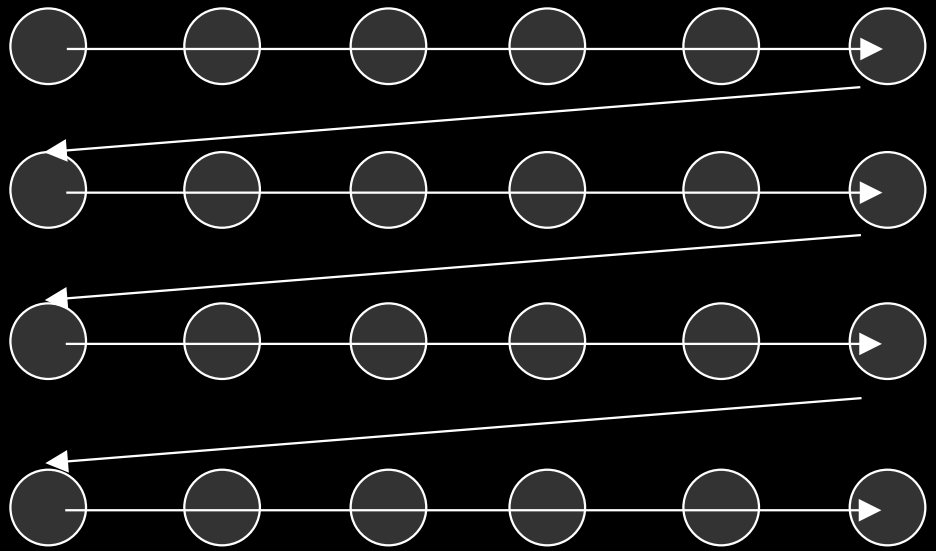
N = Number of column given in the matrix.

- Column Major Order

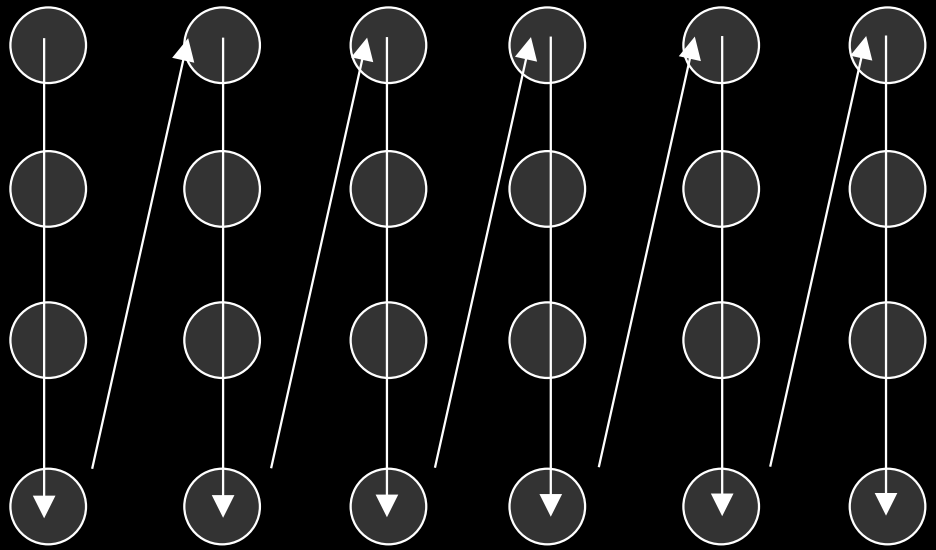
Address of A[I][J] = B + W * ((J – LC) * M + (I – LR))

M = Number of rows given in the matrix.

Row-major



Column-major



Sparse Matrix

$$f_n = \begin{vmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & c_{n-1} & a_n \end{vmatrix}.$$

Upper Triangular Matrix

$$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}_{4\times4}$$

Lower Triangular Matrix

$$L = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}_{4\times4}$$

Sparse Matrix