1.11 - 2D Arrays

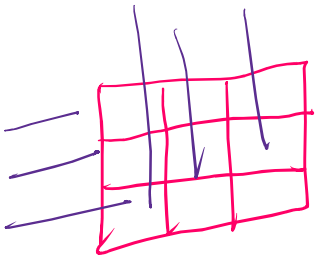
Saturday, September 13, 2025 2:46 PM

```
for (i=0; i<n-3; i++) {
    cout << "Hello";
}
```

$O(n-3) \rightarrow O(n)$

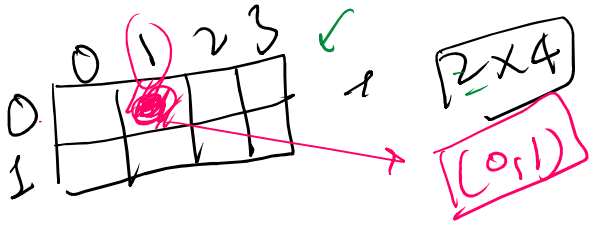
$O(n-k) = O(n+k) = O(n)$

2D Arrays



3 row
3 columns

3 × 3
rows columns

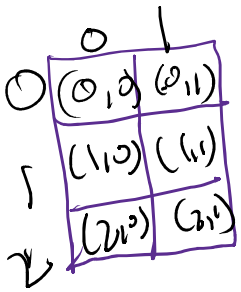


⇒ A 2D array is nothing but a matrix

⇒ A $n \times m$ 2D array / matrix

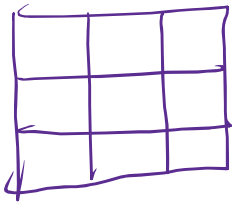
rows columns

⇒ Position / index of every cell in a 2D matrix is represented by (i, j)



3 x 2

⇒ Total no. of elements is n*m.



3 x 3 = n x n

n == m

square matrix

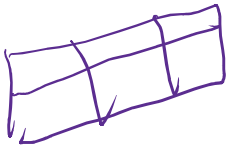
How to use 2D arrays in Code

int arr[] = new int [10];

int arr[][] = new int [2][3];

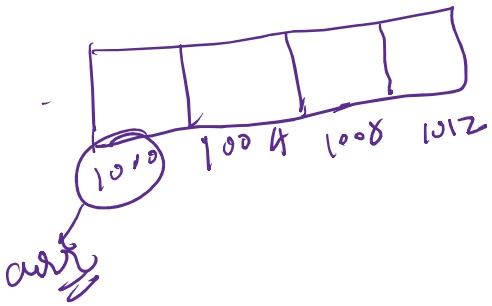
rows cols

arr → Base address

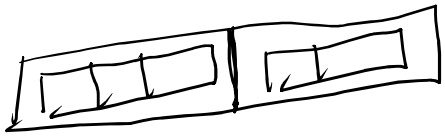


Memory representation of 2D array

int arr[] = new int [4];



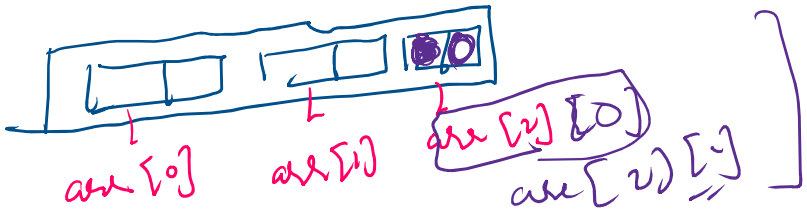
int arr[][] = new int [2][3]



⇒ int arr[][] = new int[n][m]

This means creating 'n' 1D arrays each of size 'm'

int arr[][] = new int[3][2]



Traversal

```
for (i=0; i<n; i++)
    sout (arr[i]);
```

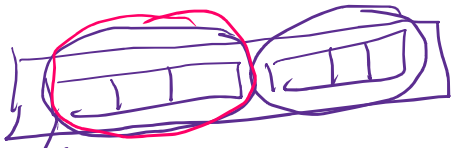
arr[n][m] ⇒ for (i=0; i<n; i++)
for (j=0; j<m; j++)
arr[i][j];

⇒ arr.length

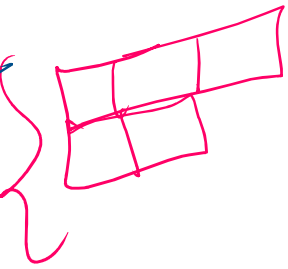
[arr.length] ⇒ rows

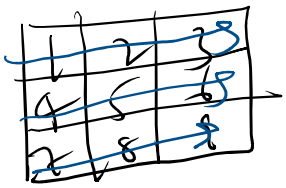
arr[0].length ⇒ columns

int arr[][]
= new int[2][3]

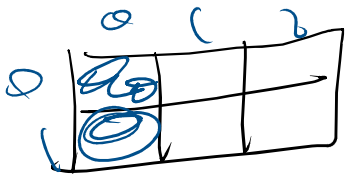


jagged matrix





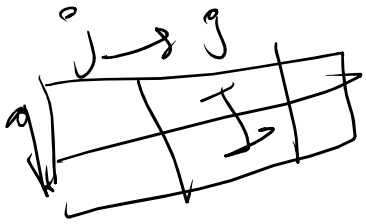
1 2 3 4 5 6 7 8 9



2×3

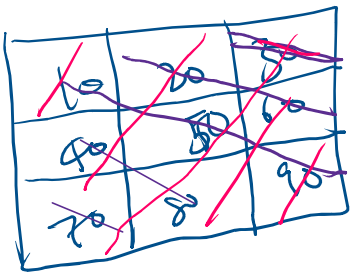
for (i=0; i<2; i++)
for (j=0; j<3; j++)
arr[i][j]

i=0, j=0
j=1
j=2
i=1
j=0



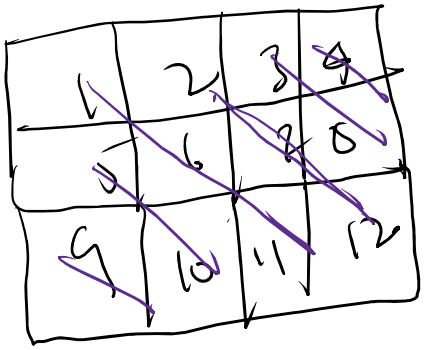
Q) Take a n x m matrix input and calculate the sum of the matrix.

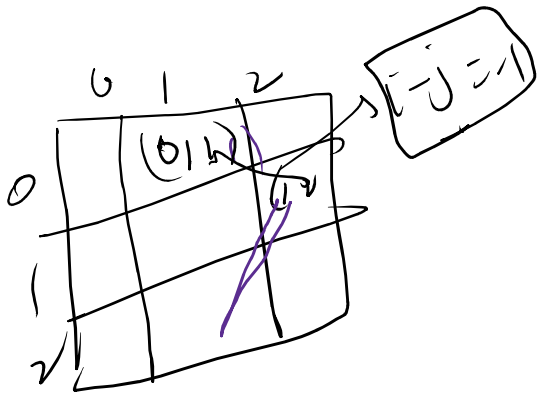
Concept of diagonals



10, 50, 90 → major diagonal

30 50 70 → major Anti diagonal



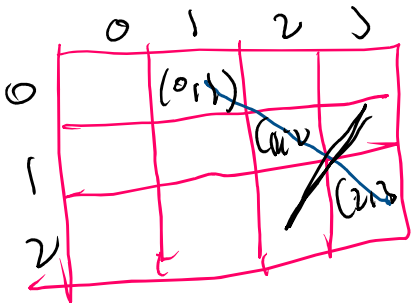


i==j → major diagonal

i+j==2 → Anti major

i-j=-1

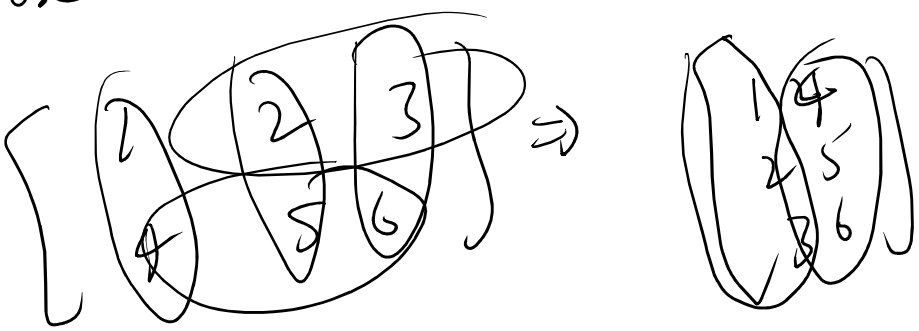
(#) Every diagnol of a matrix can be represented by some mathematical formula b/w i and j.



3 x 4

i-j==-1

Transpose



$[n \times m]^T = m \times n$

$[n \times m]^T = m \times n$