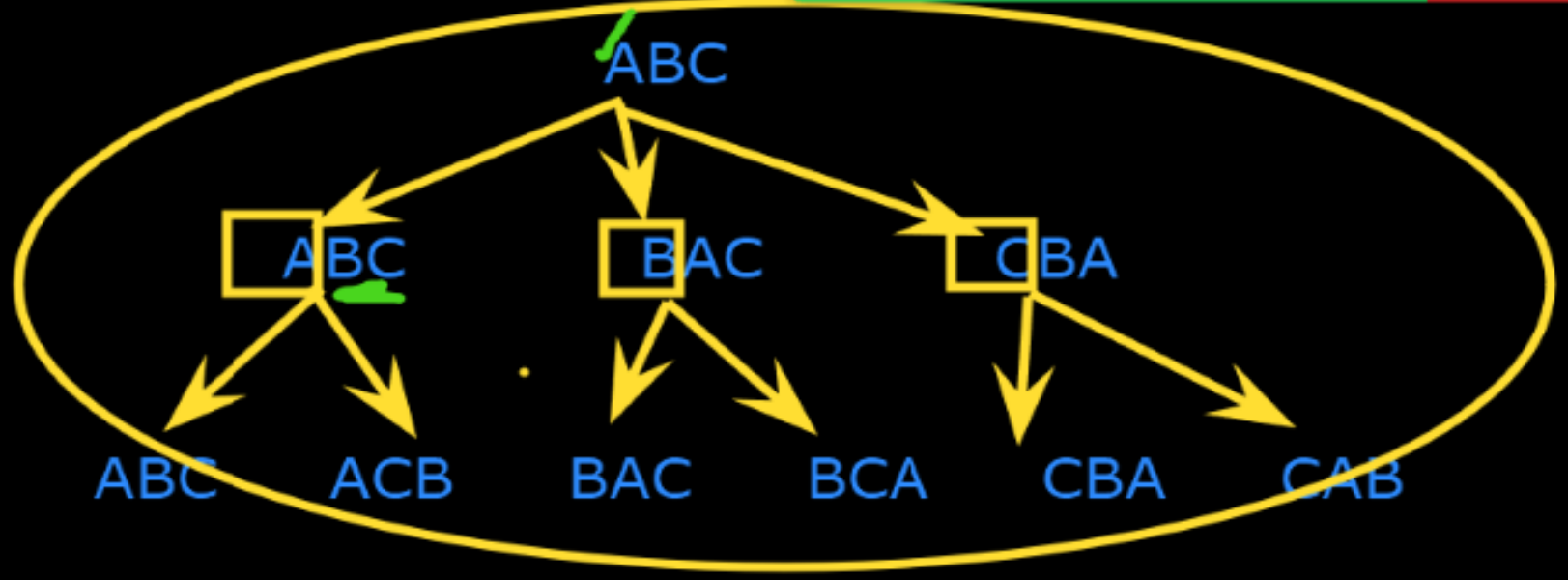
Sep22 : Day 2

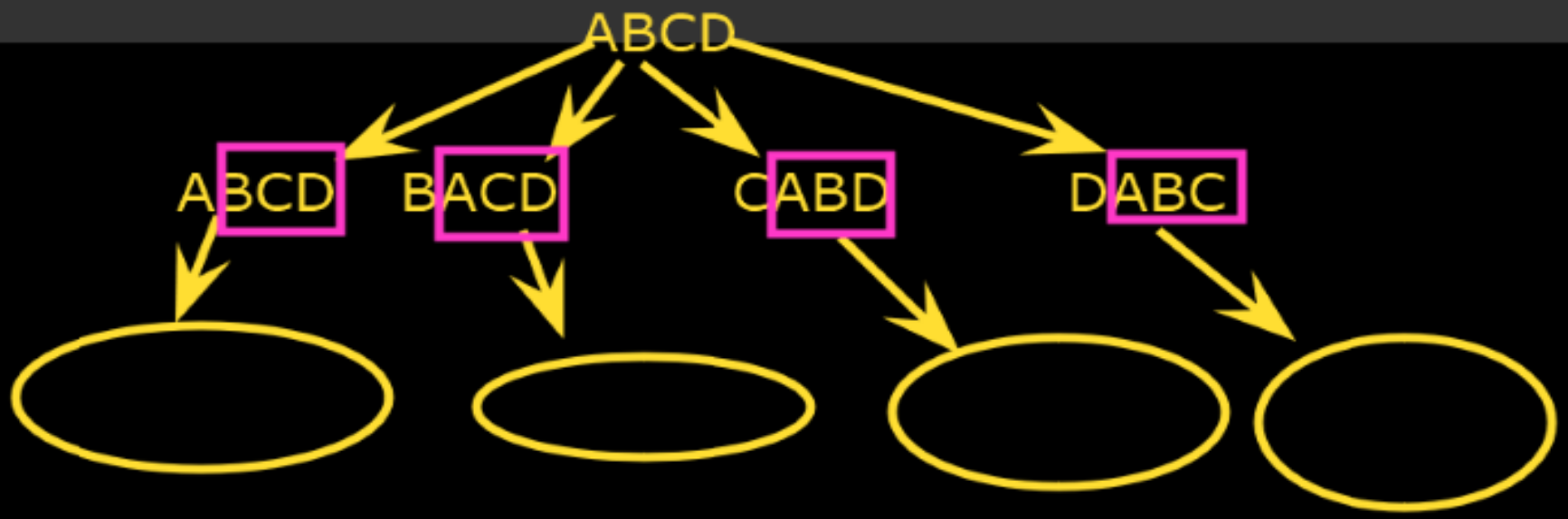
Kiran Waghmare

CDAC Mumbai

AB
BA
-----ABC
A(BC)
ACB
BAC
BCA
CAB
CBA



ABCD
A(B(C(D)))
B(A(C(D)))
C(A(B(D)))
D(A(B(C)))



Homework:

1.write a recursive program for reverse a string.(2/3 types)

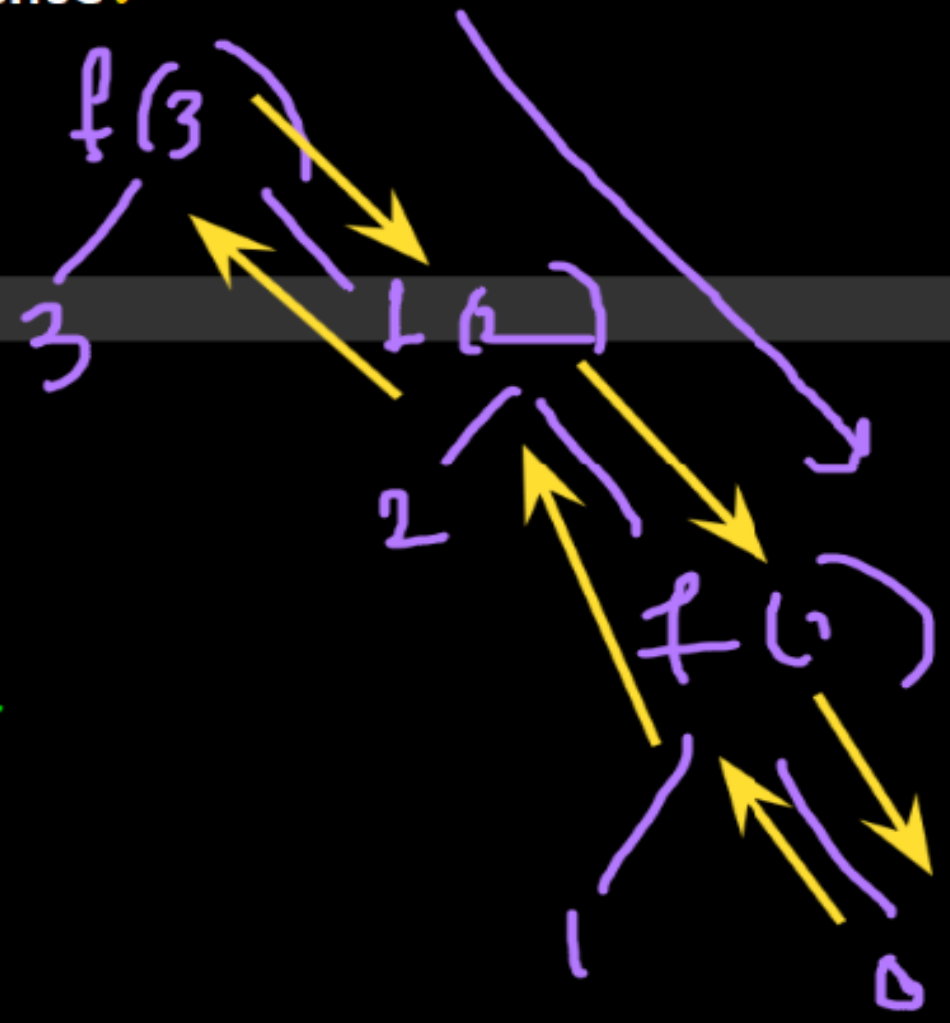
2.write a recursive program to reverse a sentence.

Types of recursive:

---

1. Tail recursion:

```
void fun(int n)
{
    if(n>0)
        SOP(n);
    else
        fun(n-1);//recursive function call
}
```



2. Head recursion

Homework:

1.write a recursive program for reverse a string.(2/3 types)

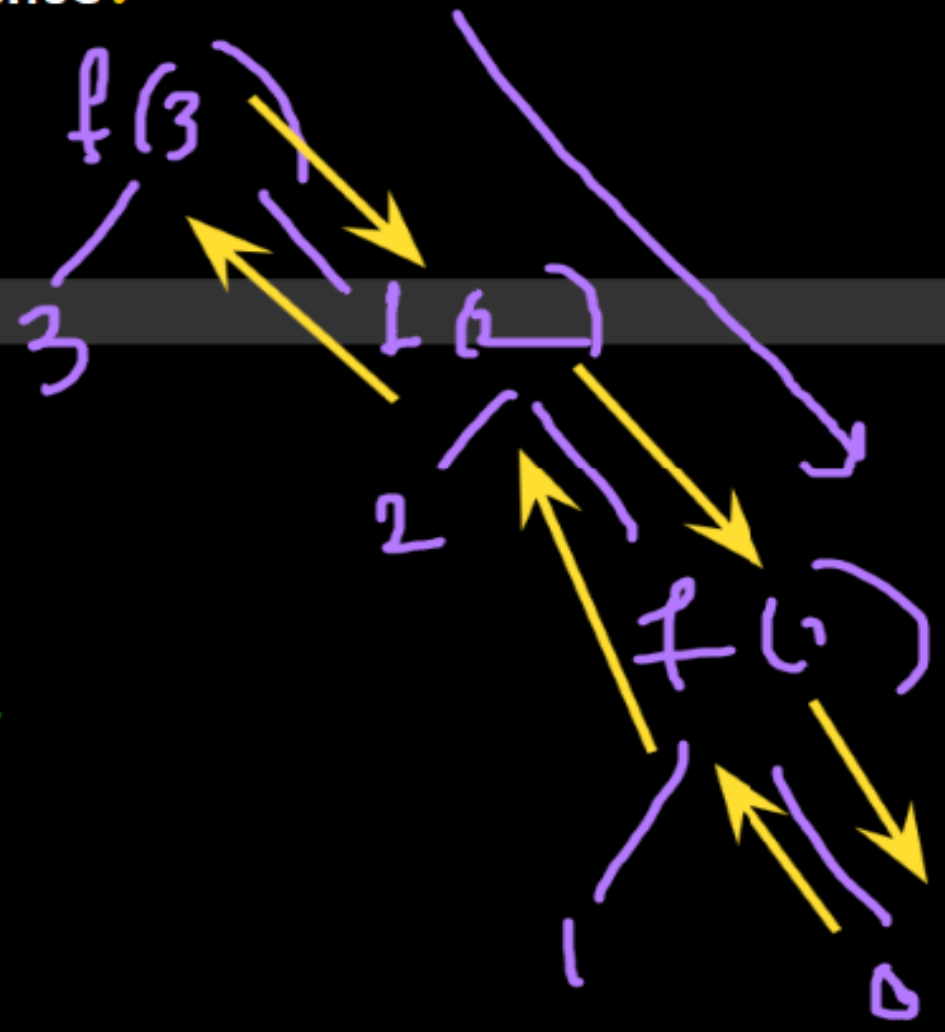
2.write a recursive program to reverse a sentence.

Types of recursive:

---

1. Tail recursion:

```
void fun(int n)
{
    if(n>0)
        SOP(n);
    else
        fun(n-1);//recursive function call
}
```



2. Head recursion

Head vs. Tail recursion

Note: base case is ALWAYS 1st

f(3)

f(2)

f(1)

f(0)

head(3) is: 2 3

```
void head(int n)
{
    if(n == 1)
        return;
    else
        head(n-1); // ←
    printf("head - n=%i\n",n););
}
```

tail(3) is: 3 2 1

```
void tail(int n)
{
    if(n == 0)
        return;
    else
        printf("tail - n=%i\n",n);
    tail(n-1);  // ←
}
```

Recursive program to find the Sum of the series 1 – 1/2 + 1/3 – 1/4 … 1/N

Given a positive integer N, the task is to find the sum of the series 1 – (1/2) + (1/3) – (1/4) +…. (1/N) using recursion.

Examples:

Input: N = 3
Output: 0.8333333333333333
Explanation:
1 – (1/2) + (1/3) = 0.8333333333333333

Input: N = 4
Output: 0.5833333333333333
Explanation:
1- (1/2) + (1/3) – (1/4) = 0.5833333333333333

Recursive Program to print multiplication table of a number

Given a number N, the task is to print its multiplication table using recursion.

Examples

Input: N = 5
Output:
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

Input: N = 8
Output:
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80

Recursive program to print formula for GCD of n integers

Given a function gcd(a, b) to find GCD (Greatest Common Divisor) of two number. It is also known that GCD of three elements can be found by gcd(a, gcd(b, c)), similarly for four element it can find the GCD by gcd(a, gcd(b, gcd(c, d))). Given a positive integer n. The task is to print the formula to find the GCD of n integer using given gcd() function.

Examples:

Input : n = 3
Output : gcd(int, gcd(int, int))

Input : n = 5
Output : gcd(int, gcd(int, gcd(int, gcd(int, int))))

GCD (4,6)=2
4=2*2
6=2*3

```
if(n==1)
return "int"
```

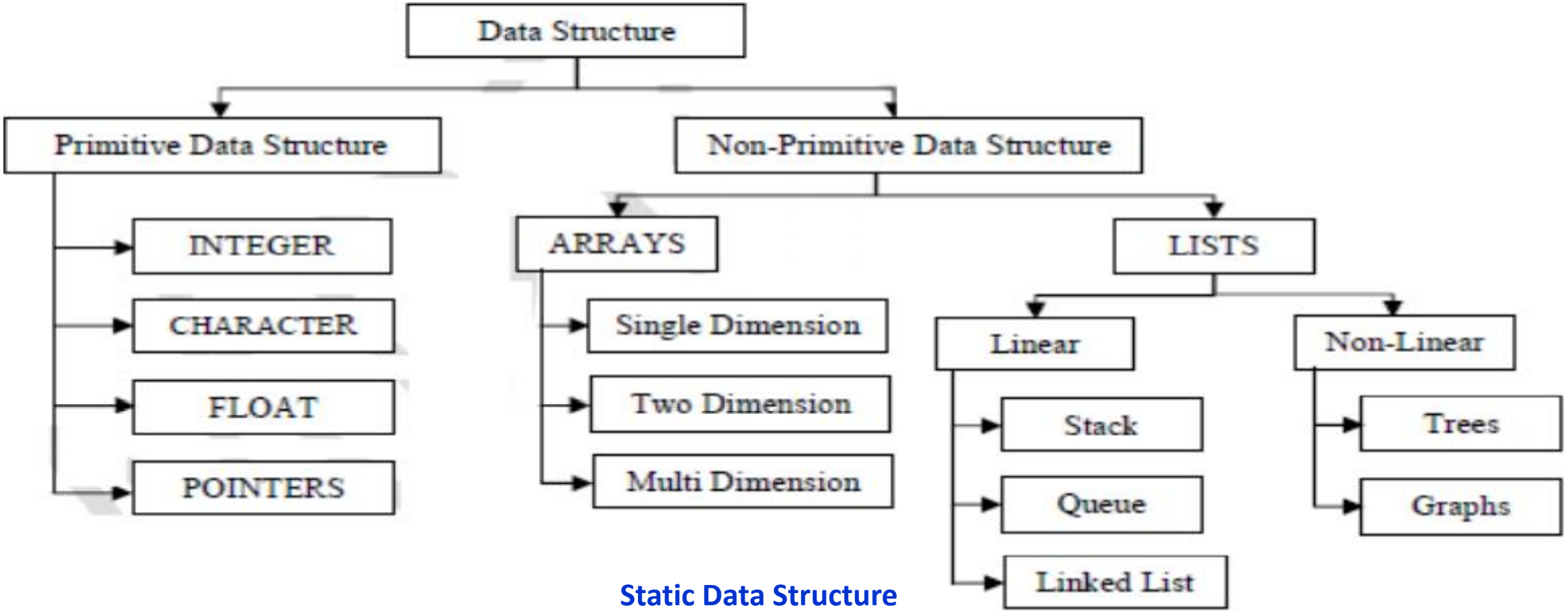
GCD(105,91)=

```
GCD(a,b)
if(a>b)
    GCD(a%b,b)
else
    GCD(a,b%a)
```

Algorithms & Data Structure
Arrays

Kiran Waghmare

Classification of Data Structure



Static Data Structure

Dynamic data structure

```
            SOP("hhjkjkd");
            SOP("hhjkjkd");
        }
```

Linear & Non-linear:

Terminologies:

Linear: data are arranged in sequence.

Non-linear:data elements are not arranged in sequence.

Homogeneous:similar type of elements.

-primitive data type:

    -int, float, byte, double, long

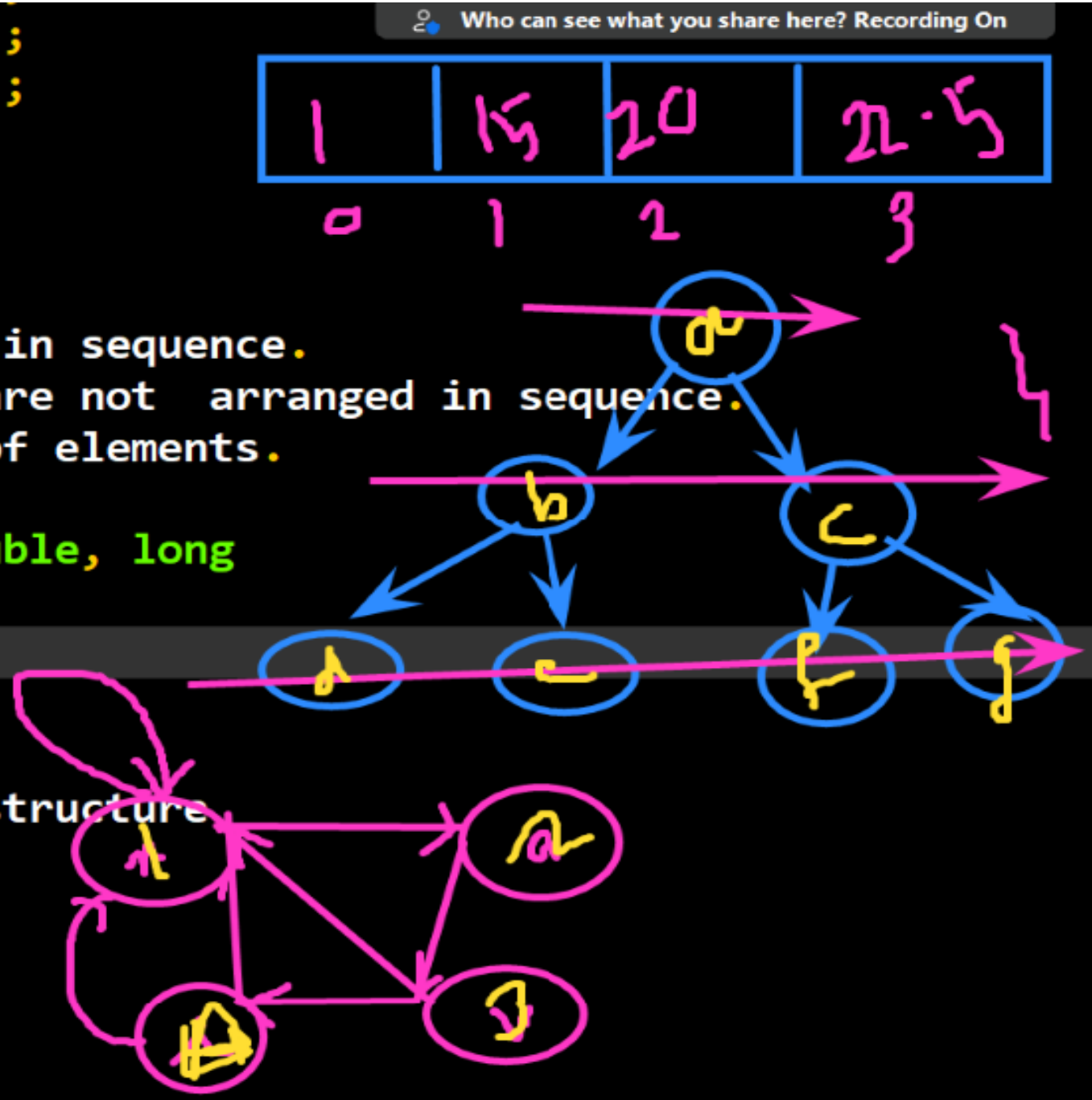
Non-homogeneous:Vector,

Static:

Dynamic:

Note: Data type and Data structure

Array:



Array:

Array is an finite,ordered, and homogenous collection of elements.

Type of indexing:

0:zero-based indexing

1:one-based indexing

n:n-based indexing

a\

A	R	R	A	Y
~~0~~	1	2	3	4

400

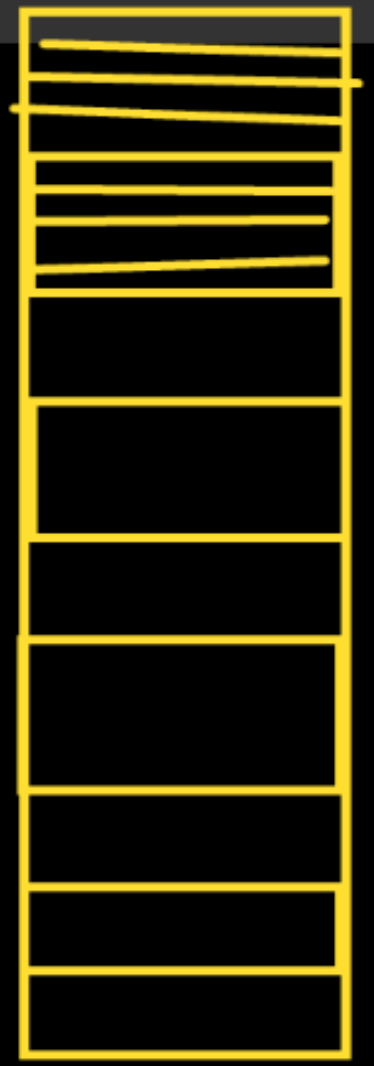
404

408

8

a[3] = 0 X 8 = 8

8 X 8 = 64



Array:

Array is an finite,ordered, and homogenous collection of elements.

Type of indexing:

0:zero-based indexing
1:one-based indexing
n:n-based indexing

1	2	3	4	5

$O(1) + O(1) + O(1) + O(1) + O(1)$

```
int a1[5]={};
int a1[]={1,2,3,4};//HW
```

Operations:

Insert
Delete
Search
Display

$O(\log n)$ $O(n)$ $O(5)$

$O(n^3)$ linear constant $O(n^2)$

Operations:

Insert

- -single element: O(1)
- -n elements: O(n)

Delete

- -single element: O(1)
- -n elements: O(n)

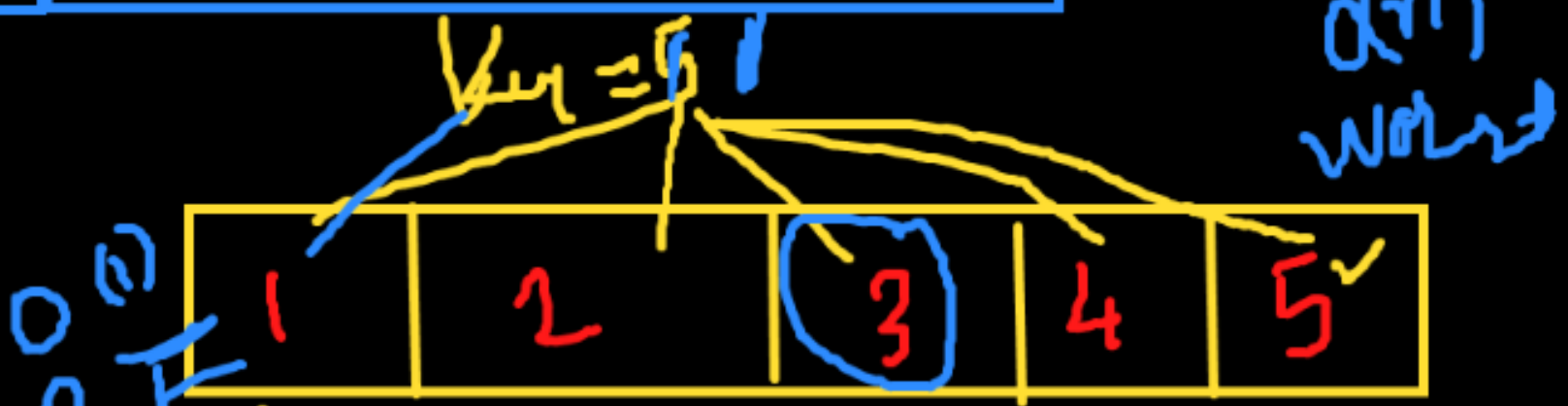
Search

- -n elements: O(n)
- -Best case: O(1)
- -Average case: O(n)
- -Worst case: O(n)

Access:

- -O(1)

Display



O(1) + O(1) + O(1) + O(1) + O(1)

5/2 (n/2)

O(n)

O(5)

O(n)

constant

Concept of array

Size of Array:

Size(A)= U-L+1

=4-0+1

=5

U=365

L=211

Address of Array:

1-D:

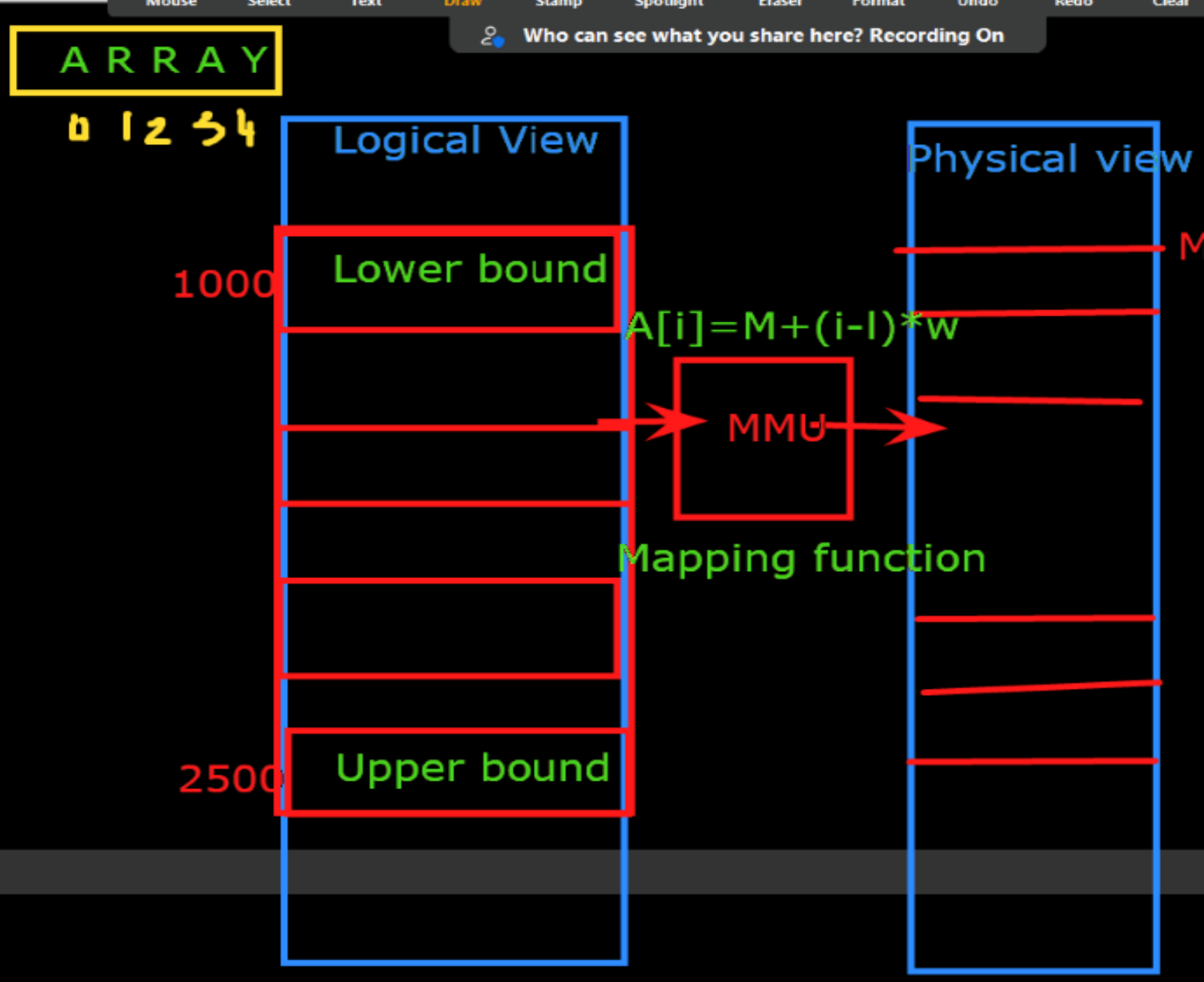
A[i]=M+(i-L)*w

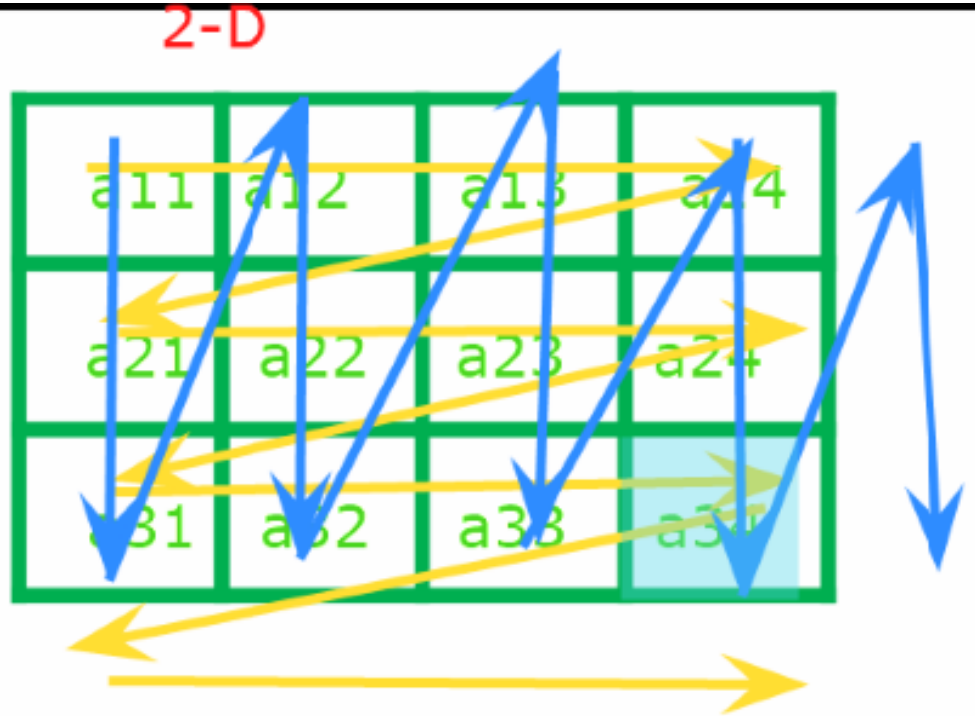
M=100, i=2, int

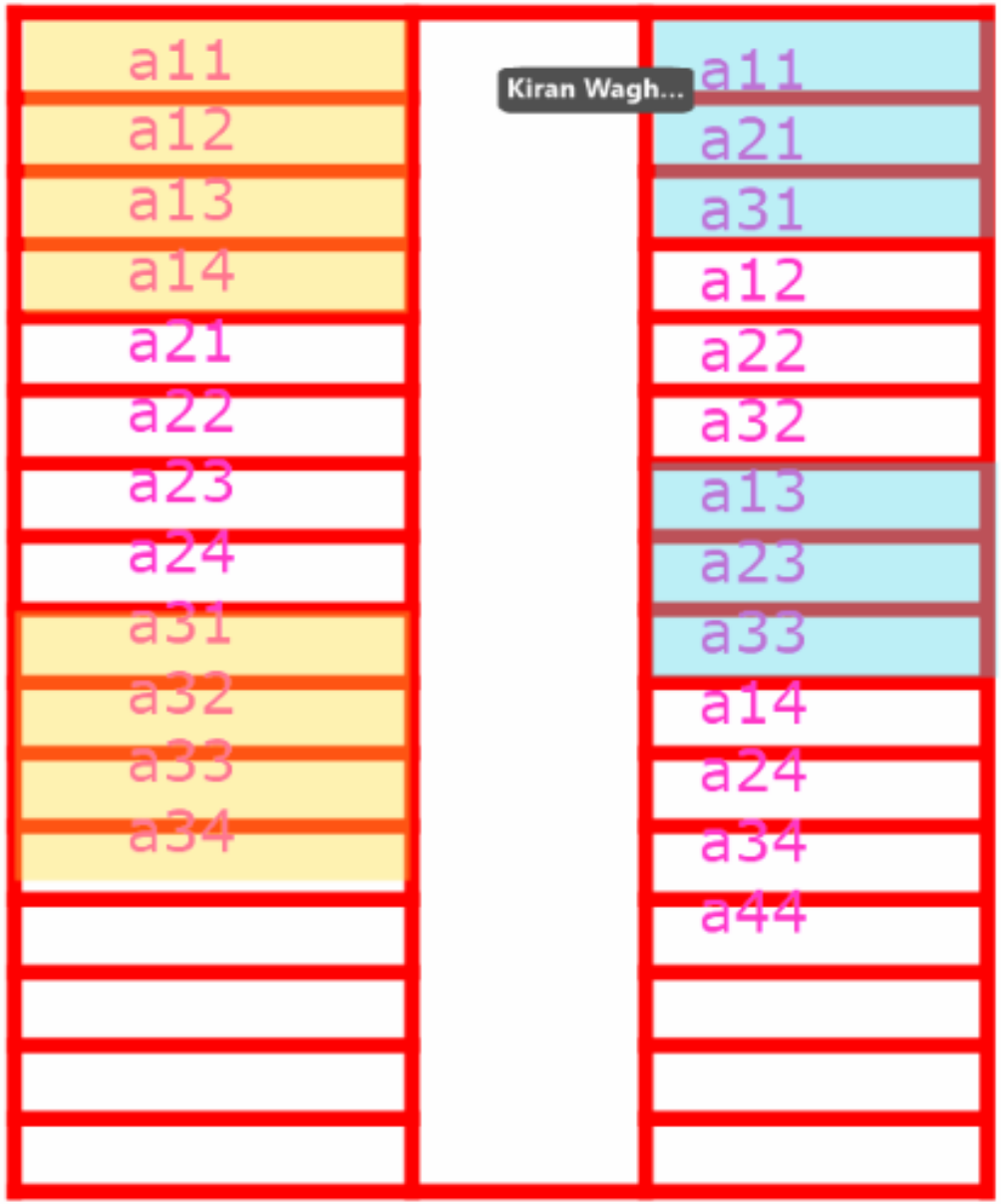
A[2]=100+(2-0)*2

=104

2-D:




2-D
a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34


a11
a12
a13
a14
a21
a22
a23
a24
a31
a32
a33
a34
Kiran Wagh...
a11
a21
a31
a12
a22
a32
a13
a23
a33
a14
a24
a34
a44
Row Major
Column Major

Address of Array:

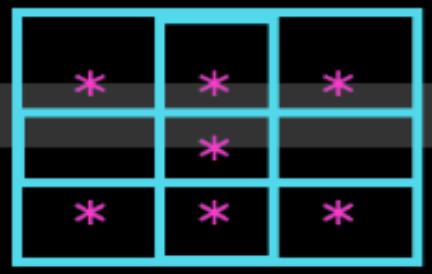
1-D:

A[i]=M+(i-L)*w

M=100, i=2, int

A[2]=100+(2-0)*2

=104



2-D:

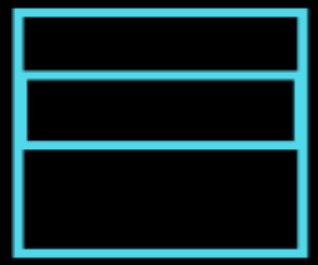
Row major Order:

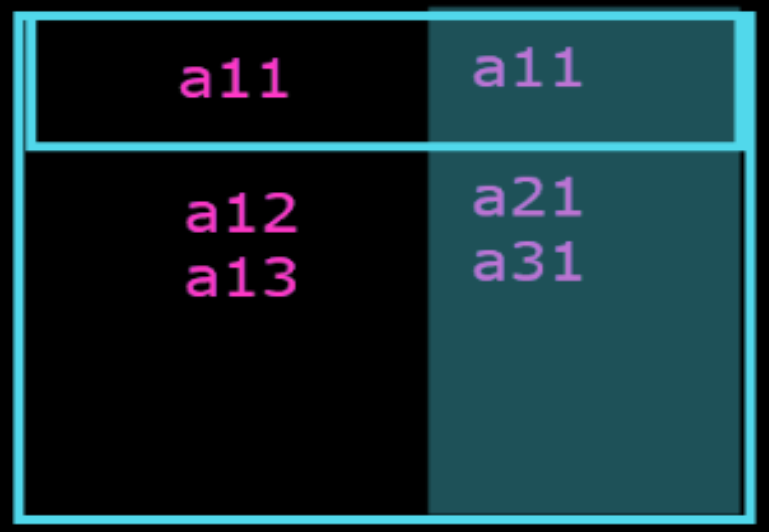
Address[aij]=M+(i-1)*n+j-1

M=100

a13=100+(1-1)*4+3-1

=102





a13

Column major Order:

Address[aij]=M+(j-1)*m+i-1

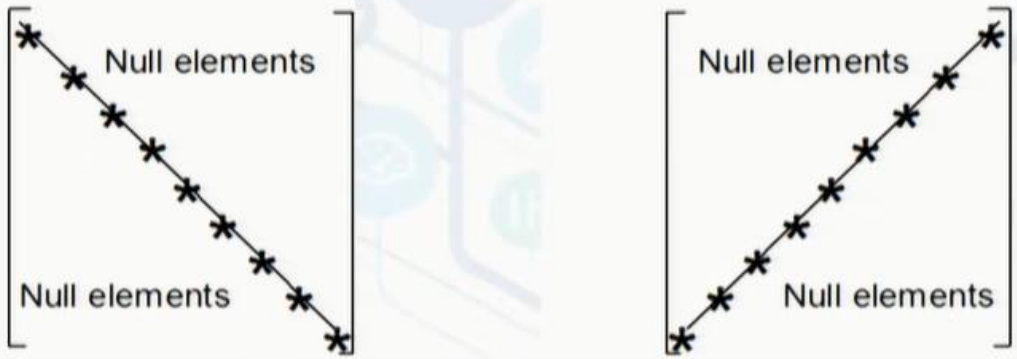
a13=100+(3-1)*3+1-1

Sparse matrix

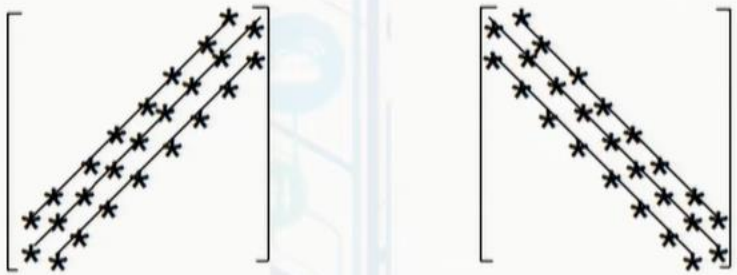
A *sparse* matrix is a two-dimensional array having the value of majority elements as null

```
[ - - - * - - - * - - - - * - - ]
[ - - - - * - - - * - - - - - - ]
[ - - - - - - * * * - - - - - ]
[ * - - - - * - - - * - - - - * ]
[ - - - - * - - - - * - - - - * ]
[ - - * - - - - * - - - - - - * ]
[ - - - - - * - - - - - * - - - ]
[ - - - - * - - - - - * - - - - ]
```

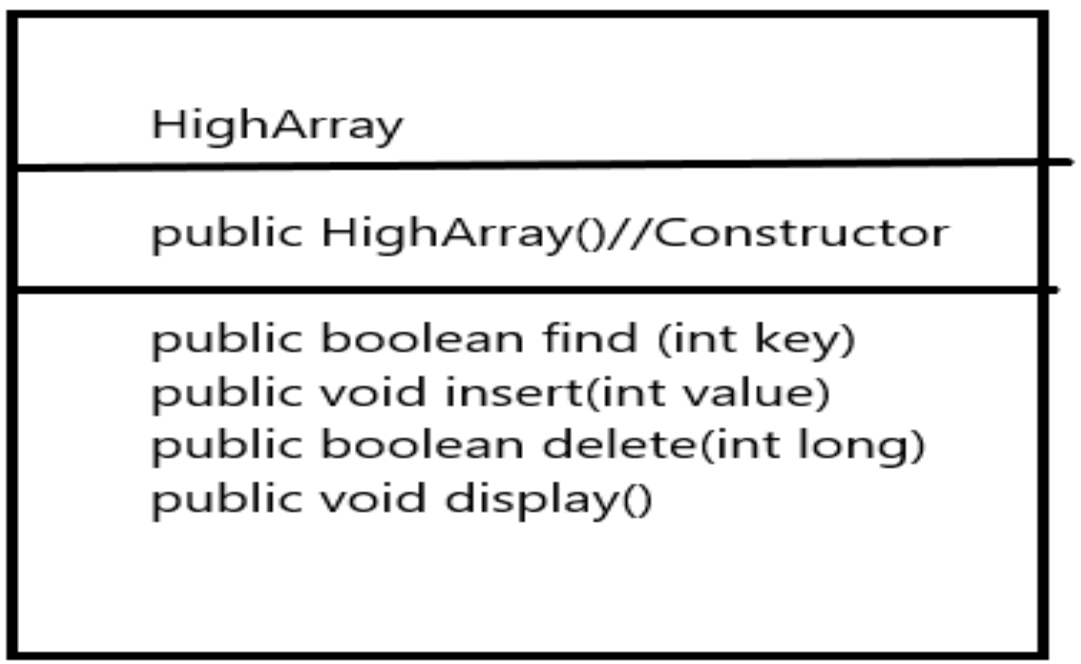
Diagonal sparse matrices

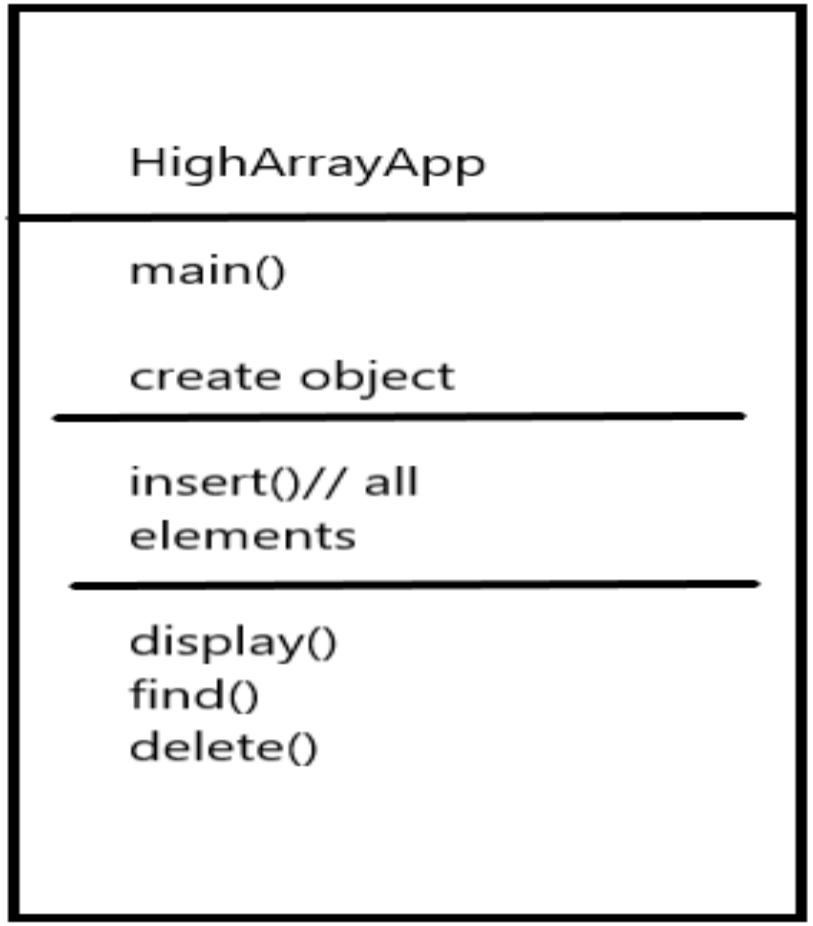


Tri-diagonal sparse matrices



Program


HighArray
public HighArray()//Constructor
public boolean find (int key)
public void insert(int value)
public boolean delete(int long)
public void display()


HighArrayApp
main()
create object
insert()// all elements
display()
find()
delete()