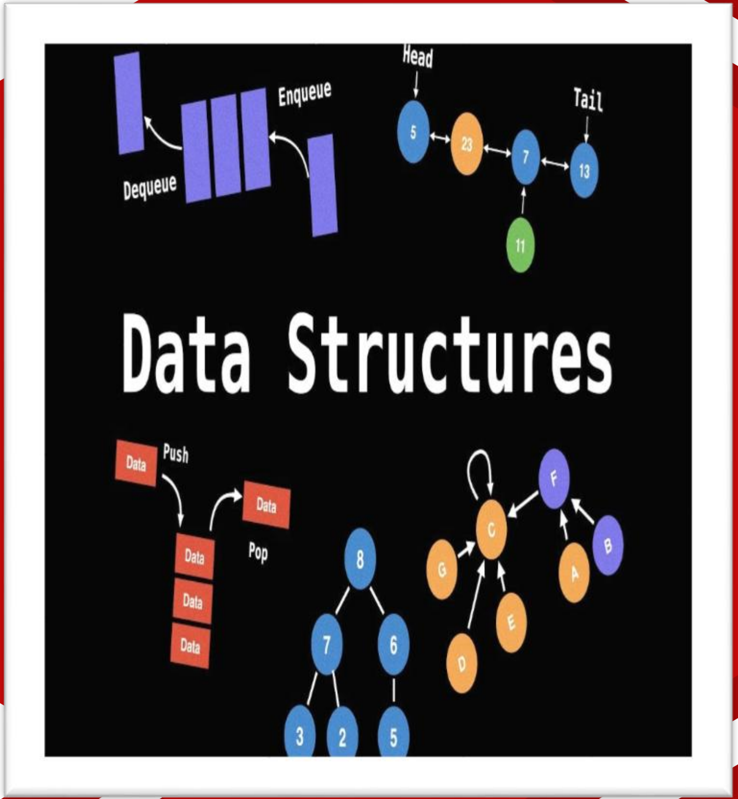
Algorithms and Data Structures

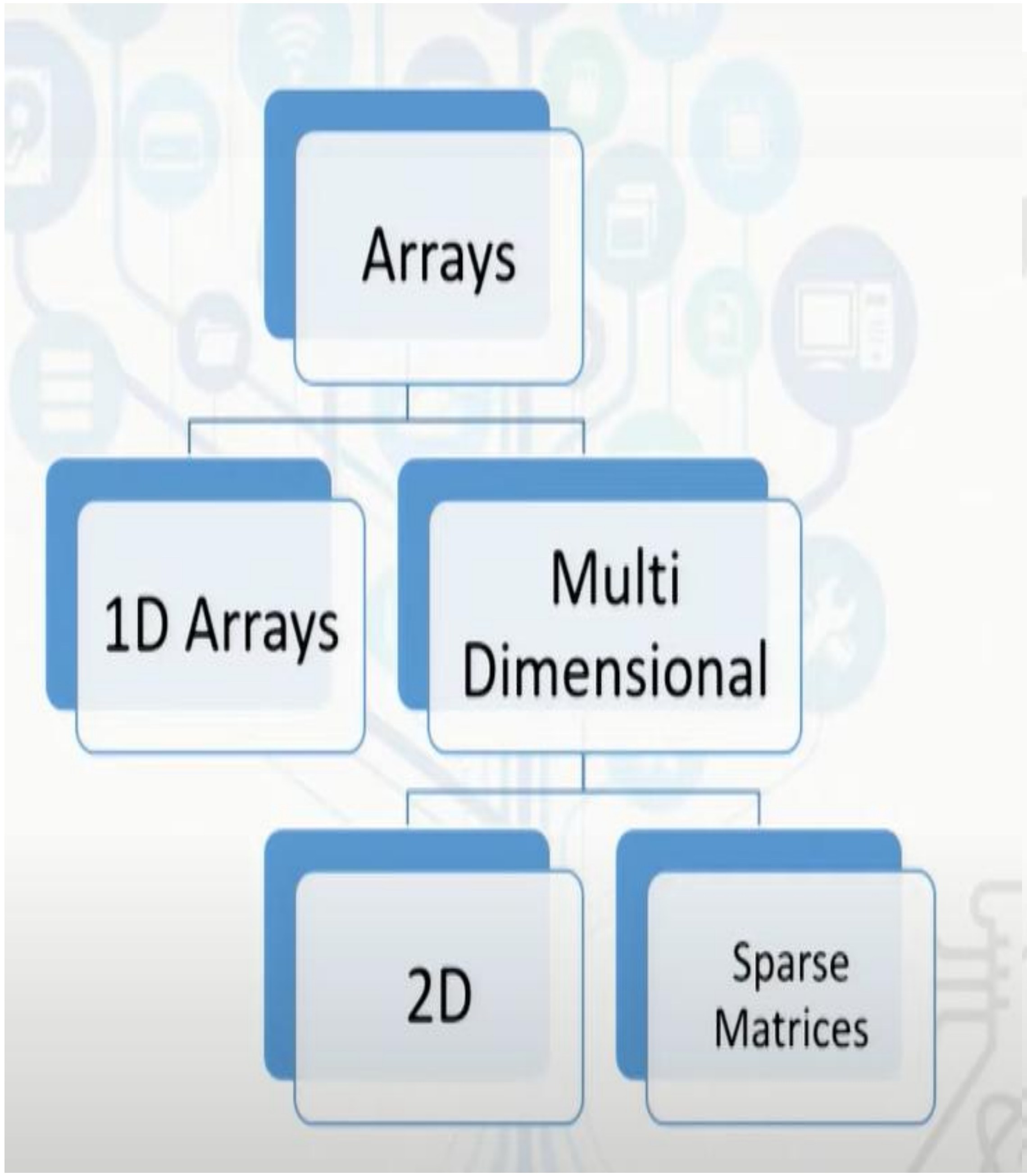
Arrays: Data Structure

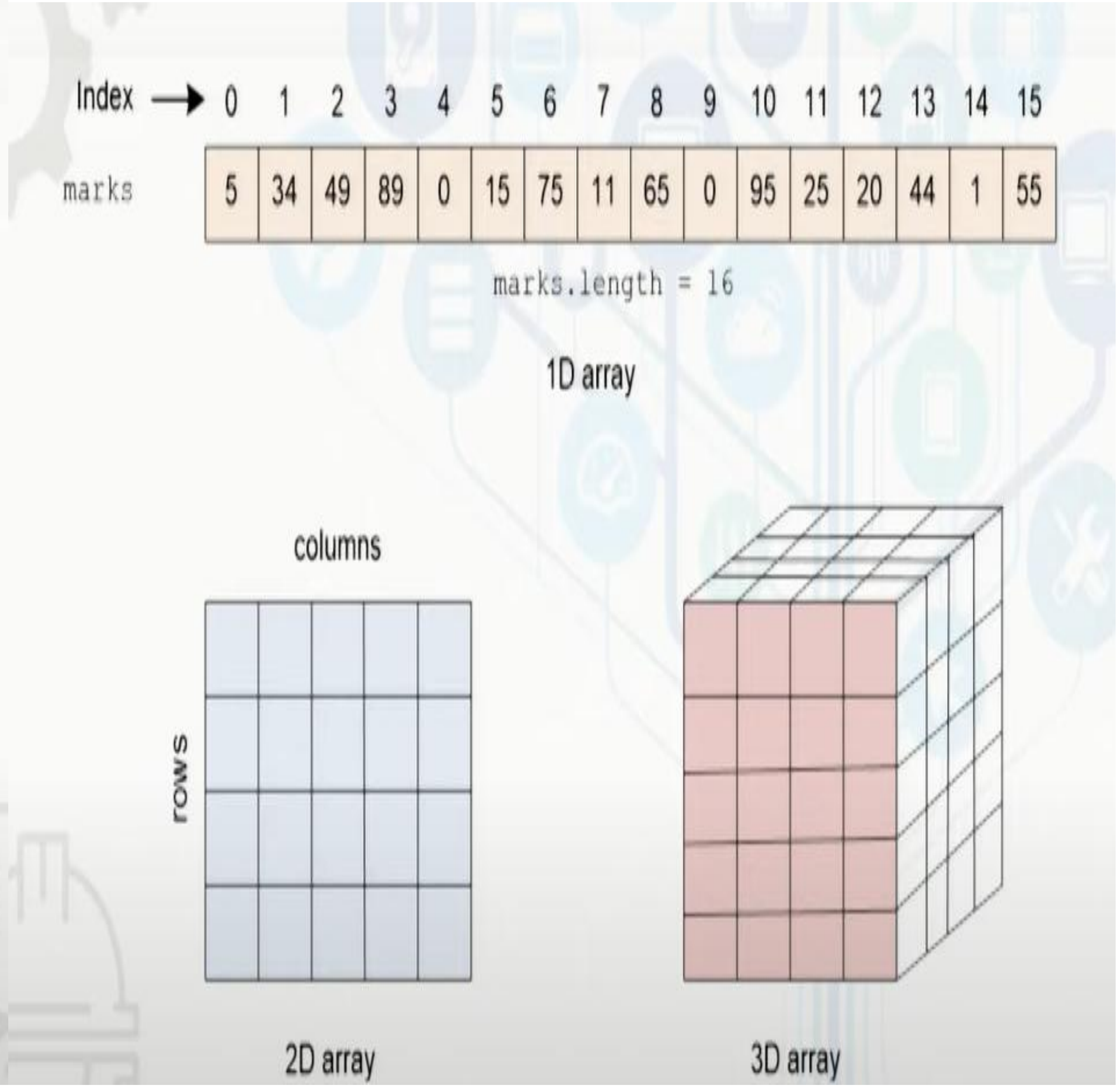
S e s s i o n : D a y 3

Dr Kiran Waghmare
CDAC Mumbai



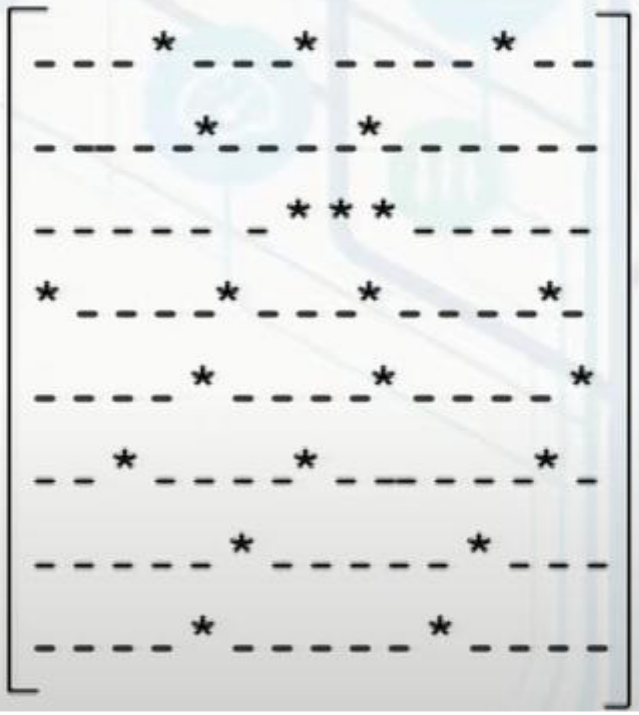
Concept of array


Arrays
1D Arrays
Multi Dimensional
2D
Sparse Matrices

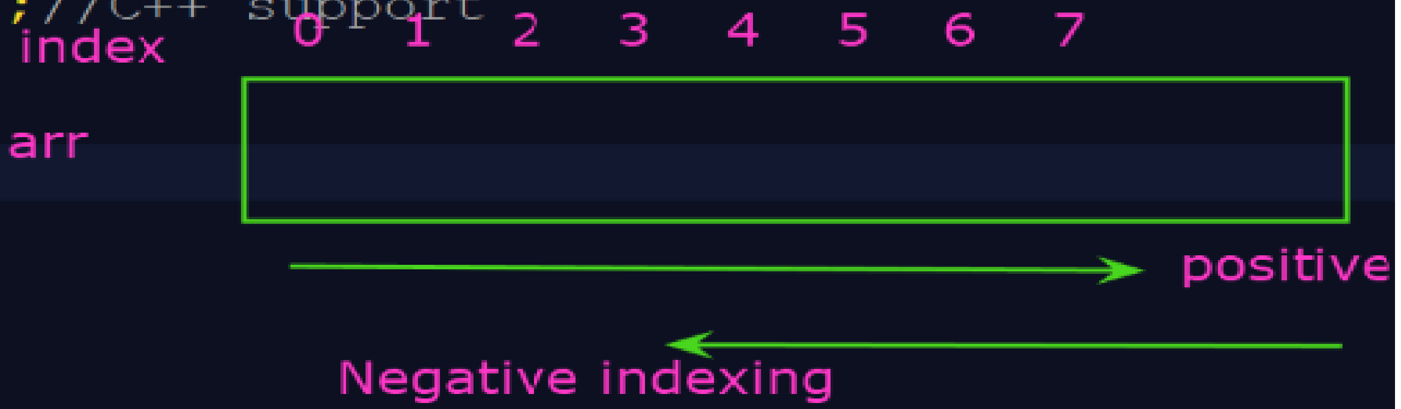

Index → 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
marks 5 34 49 89 0 15 75 11 65 0 95 25 20 44 1 55
marks.length = 16
1D array
columns
rows
2D array
3D array

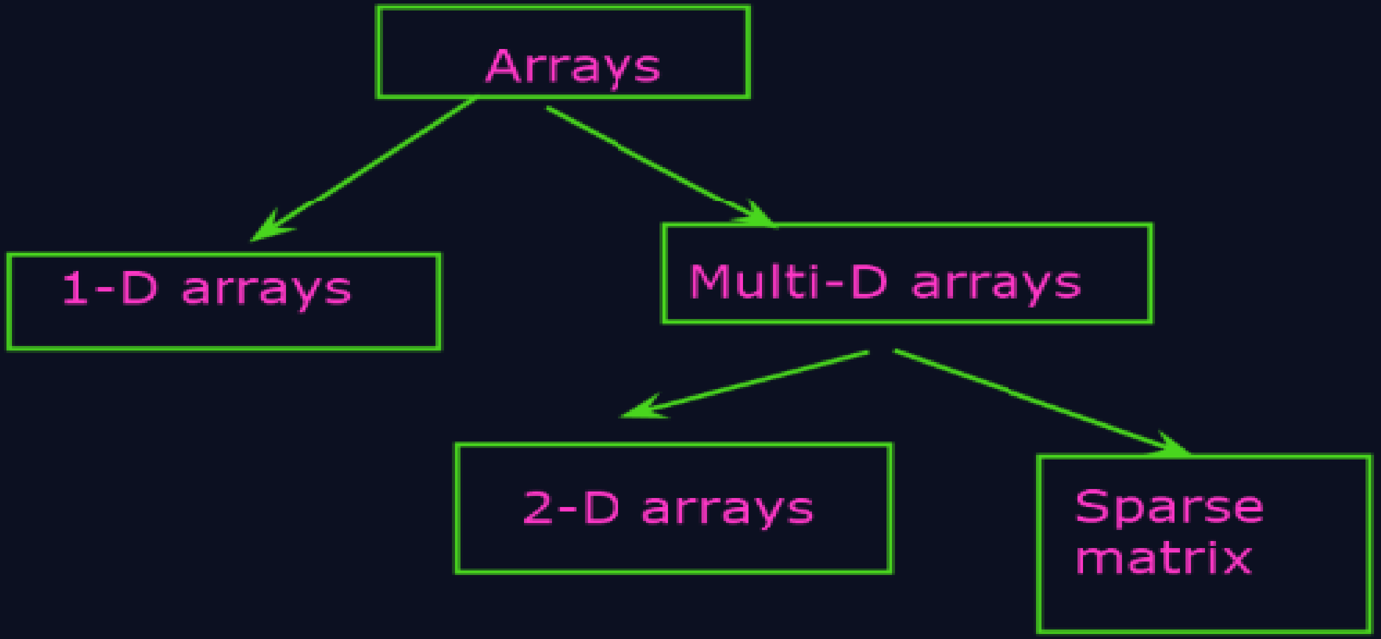
Sparse matrix

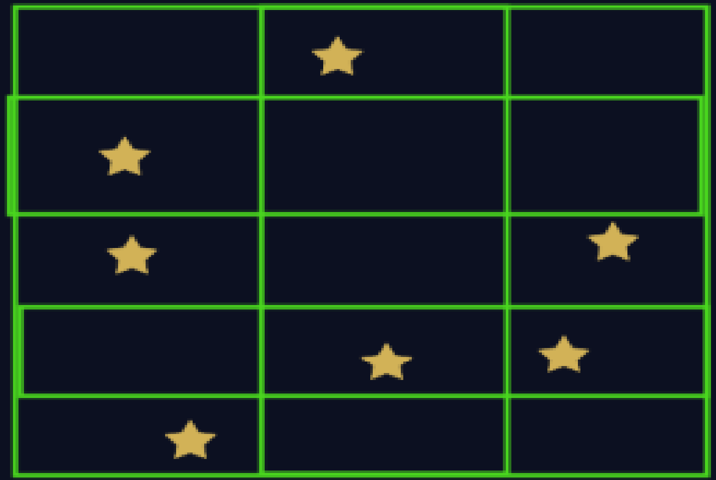
A *sparse* matrix is a two-dimensional array having the value of majority elements as null



```
int arr[ ] {1,2,3,4,5,6,7,8};//C++ support
```



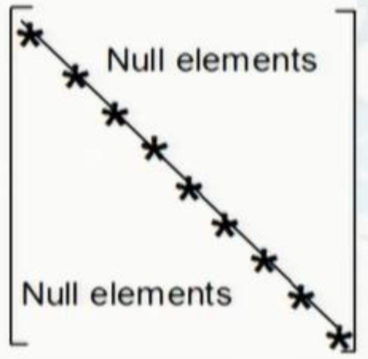


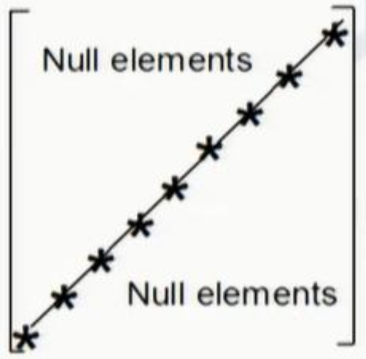


Sparse matrix: It is a 2-D matrix array having the value od elements as null

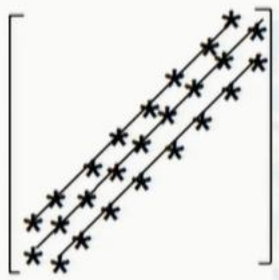
Sparse matrix

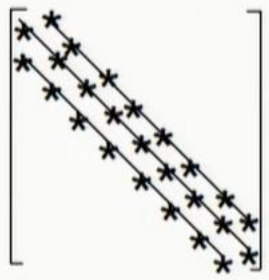
Diagonal sparse matrices

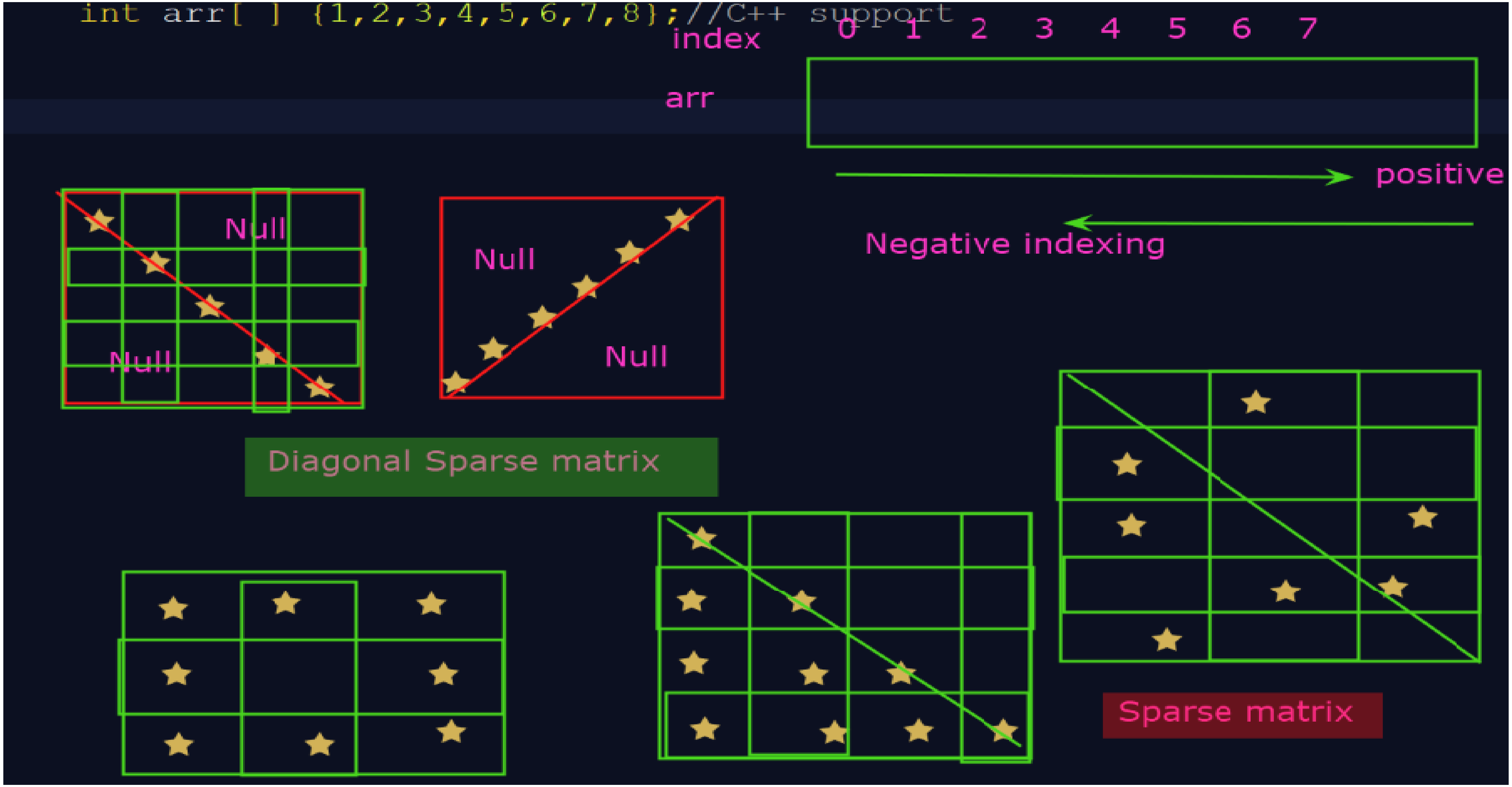




Tri-diagonal sparse matrices







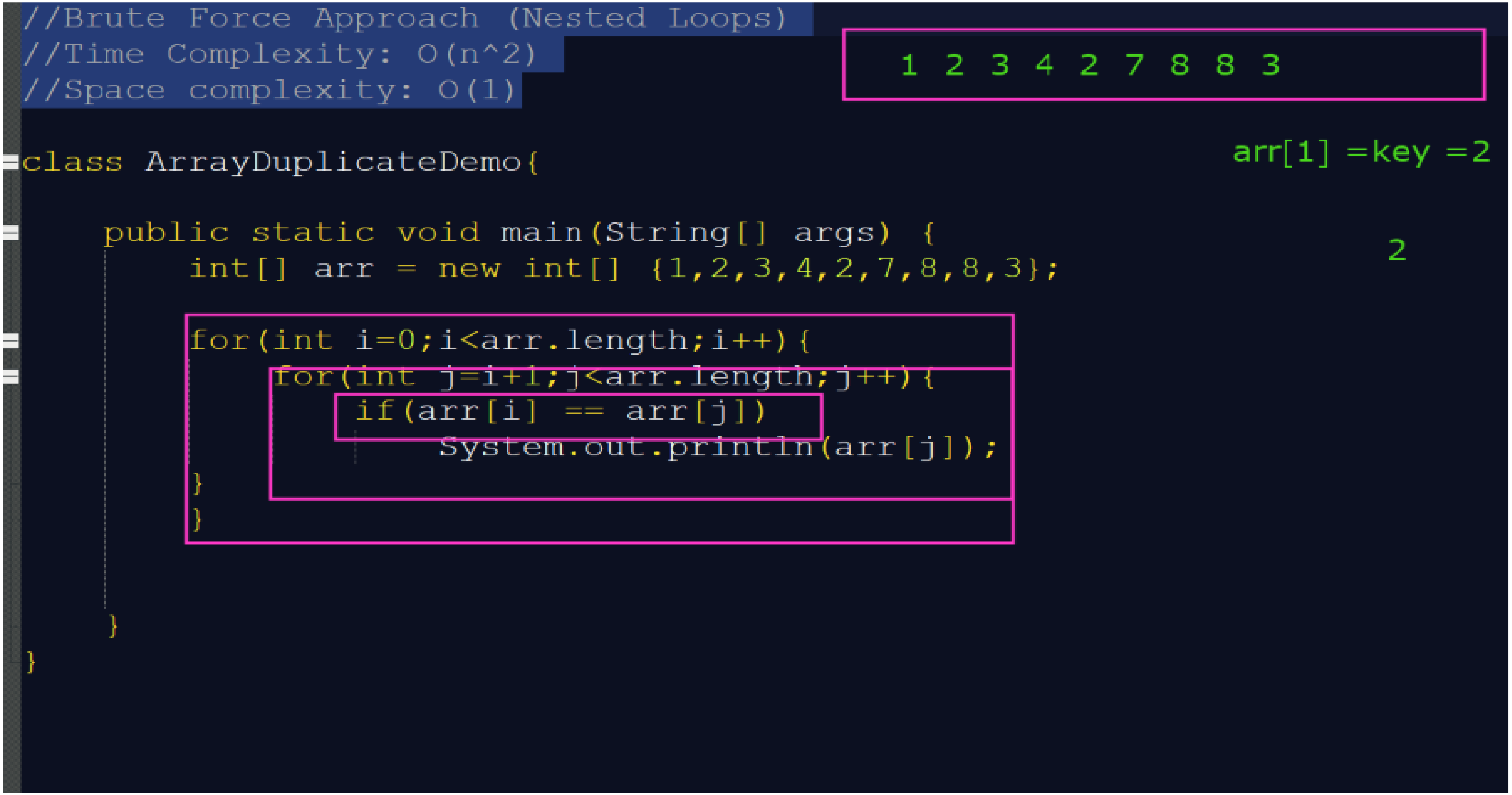
```
int arr[ ] {1,2,3,4,5,6,7,8};//C++ support
```

Problem statement: Find duplicates in an array

- Given an array a1[] of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array.

- **Example 1:**
  - Input:
    - N = 4
    - a[] = {0,3,1,2}
  - Output: -1
  - Explanation: N=4 and all elements from 0 to (N-1 = 3) are present in the given array. Therefore output is -1.

- **Example 2:**
  - Input:
    - N = 5
    - a[] = {2,3,1,2,3}
  - Output: 2 3
  - Explanation: 2 and 3 occur more than once in the given array.

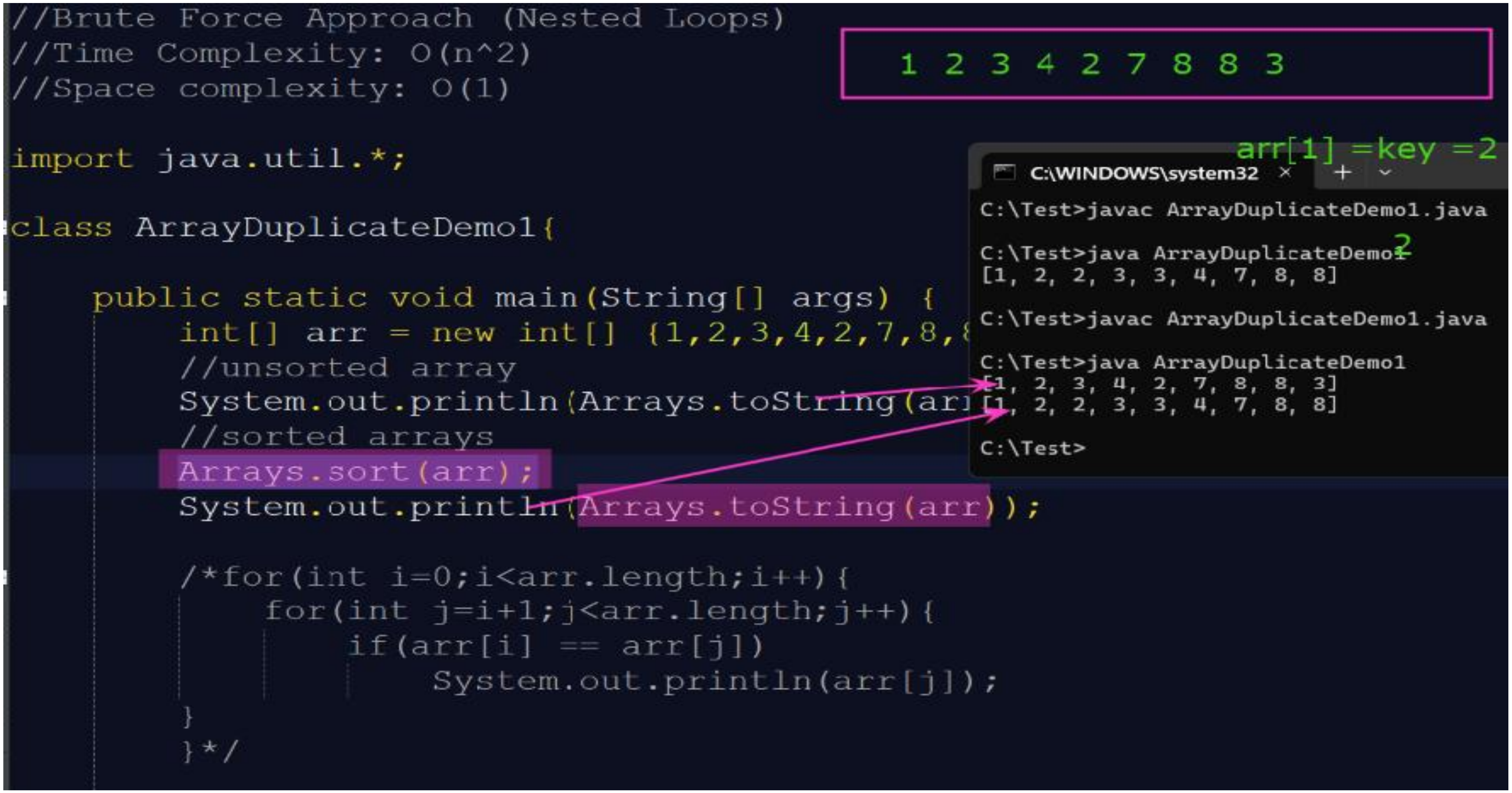


```
//Brute Force Approach (Nested Loops)
//Time Complexity: O(n^2)
//Space complexity: O(1)

class ArrayDuplicateDemo{

    public static void main(String[] args) {
        int[] arr = new int[] {1,2,3,4,2,7,8,8,3};

        for(int i=0;i<arr.length;i++){
            for(int j=i+1;j<arr.length;j++){
                if(arr[i] == arr[j])
                    System.out.println(arr[j]);
        }
        }

    }
}
```

1 2 3 4 2 7 8 8 3

arr[1] =key =2

2



```
//Brute Force Approach (Nested Loops)
//Time Complexity: O(n^2)
//Space complexity: O(1)

import java.util.*;

class ArrayDuplicateDemo1{

    public static void main(String[] args) {
        int[] arr = new int[] {1,2,3,4,2,7,8,
        //unsorted array
        System.out.println(Arrays.toString(ar
        //sorted arrays
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr));

        /*for(int i=0;i<arr.length;i++){
            for(int j=i+1;j<arr.length;j++){
                if(arr[i] == arr[j])
                    System.out.println(arr[j]);
        }
        }*/
```

1 2 3 4 2 7 8 8 3

arr[1] =key =2

C:\WINDOWS\system32

```
C:\Test>javac ArrayDuplicateDemo1.java

C:\Test>java ArrayDuplicateDemo1
[1, 2, 2, 3, 3, 4, 7, 8, 8]

C:\Test>javac ArrayDuplicateDemo1.java

C:\Test>java ArrayDuplicateDemo1
[1, 2, 3, 4, 2, 7, 8, 8, 3]
[1, 2, 2, 3, 3, 4, 7, 8, 8]

C:\Test>
```

2

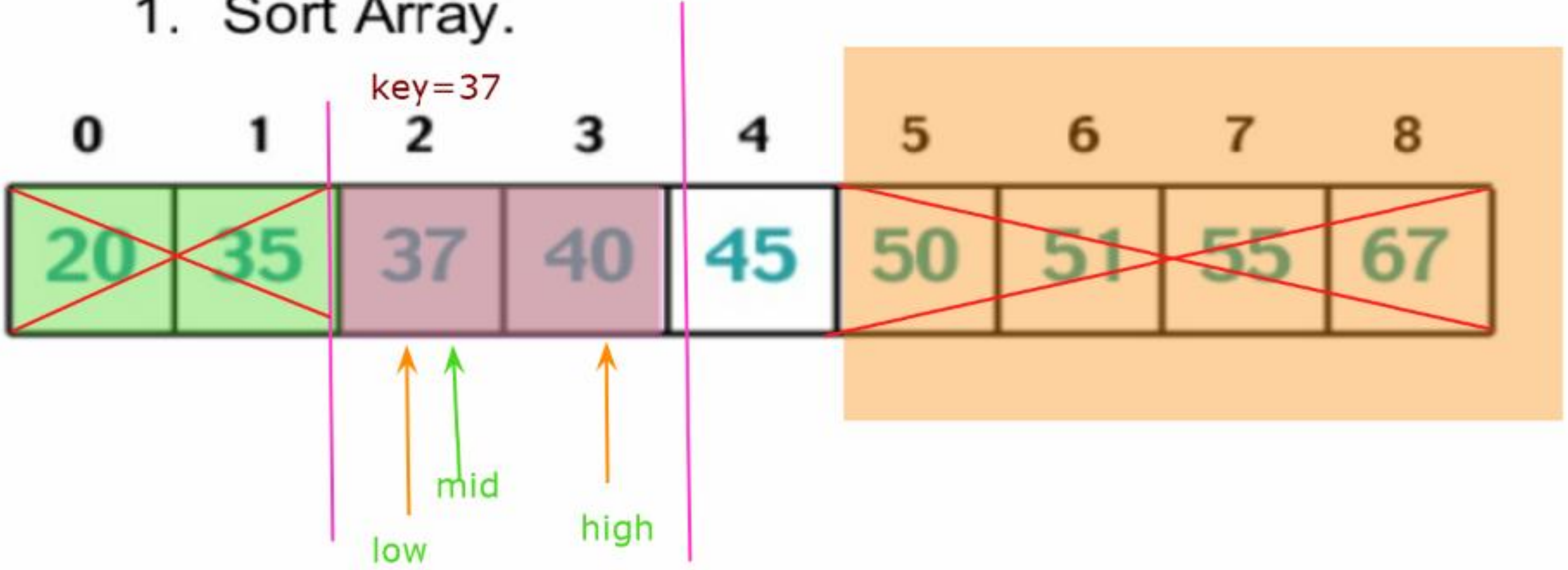
Binary Search

mid=(0+8)/2=4

mid=(0+3)/2=1.5~1

mid=(2+3)/2=2.5~2

- Find 37?
  1. Sort Array.

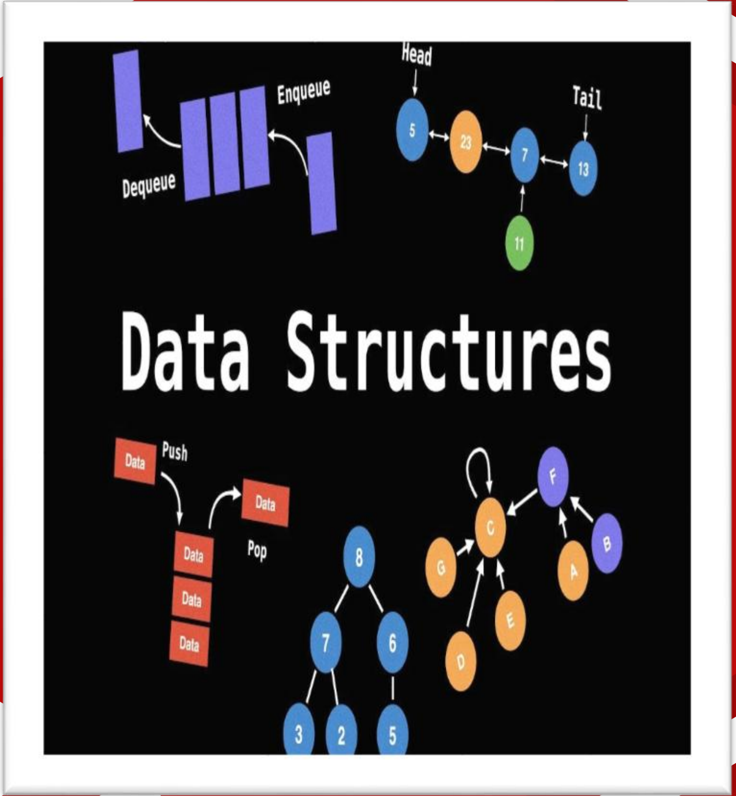


Algorithms and Data Structures

Arrays: Data Structure
Searching Techniques

S e s s i o n : D a y 3

Dr Kiran Waghmare
CDAC Mumbai



Searching in Arrays

- **Searching:** It is used to find out the location of the data item if it exists in the given collection of data items.

  E.g. We have linear array A as below:

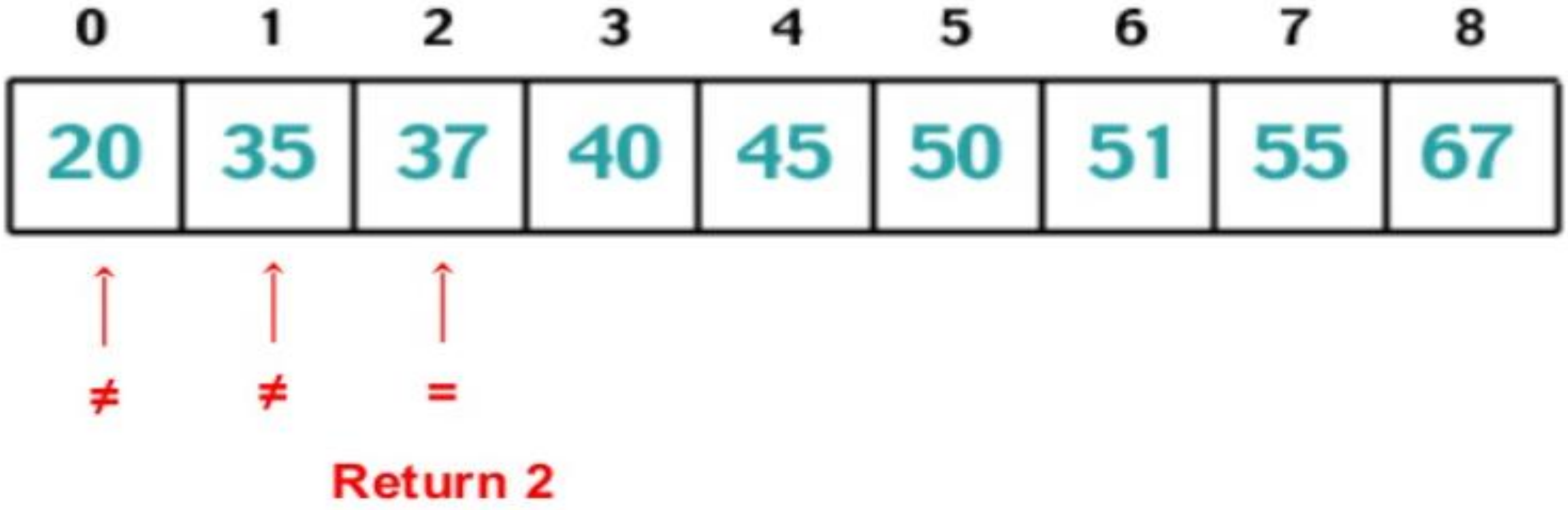
1	2	3	4	5
15	50	35	20	25

Suppose item to be searched is 20. We will start from beginning and will compare 20 with each element. This process will continue until element is found or array is finished. Here:

1) Compare 20 with 15
   20 # 15, go to next element.
2) Compare 20 with 50
   20 # 50, go to next element.
3) Compare 20 with 35
   20 #35, go to next element.
4) Compare 20 with 20
   20 = 20, so 20 is found and its location is 4.

Linear Search

- Find 37?



Program 1

Problem: Given an array arr[] of n elements, write a function to search a given element x in arr[].

Examples :

Input : arr[] = {10, 20, 80, 30, 60, 50, 110, 100, 130, 170}
x = 110;
Output : 6
Element x is present at index 6

Input : arr[] = {10, 20, 80, 30, 60, 50,
110, 100, 130, 170}
x = 175;
Output : -1
Element x is not present in arr[].

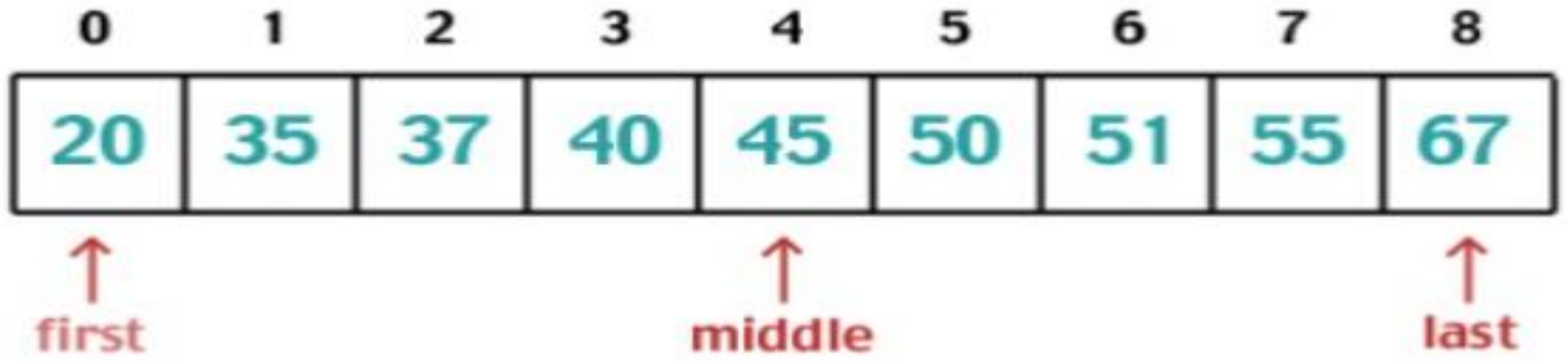
Binary Search

- Find 37?
  1. Sort Array.

0	1	2	3	4	5	6	7	8
20	35	37	40	45	50	51	55	67

Binary Search

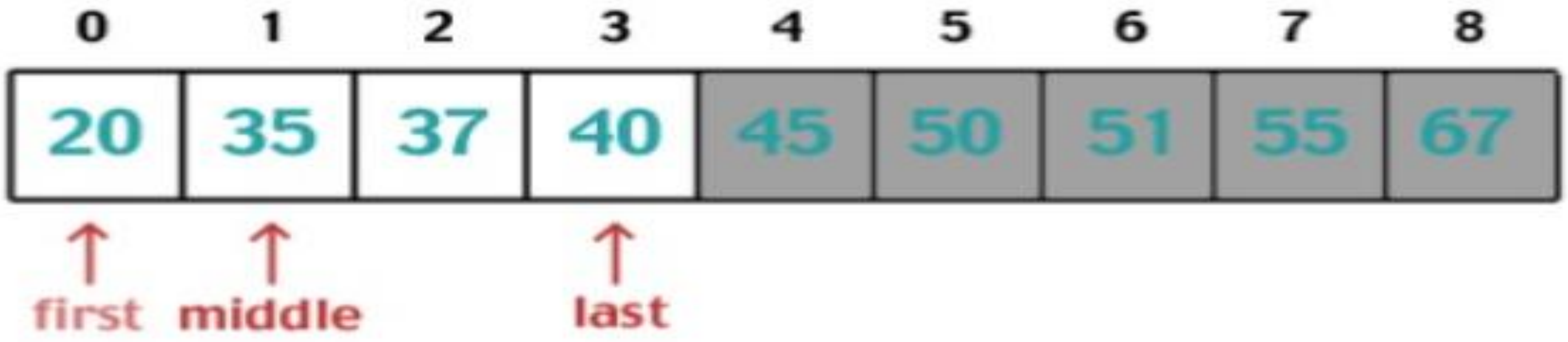
2. Calculate middle = (low + high) / 2.
   = (0 + 8) / 2 = 4.



If 37 == array[middle] ➔ return middle
Else if 37 < array[middle] ➔ high = middle -1
Else if 37 > array[middle] ➔ low = middle +1

Binary Search

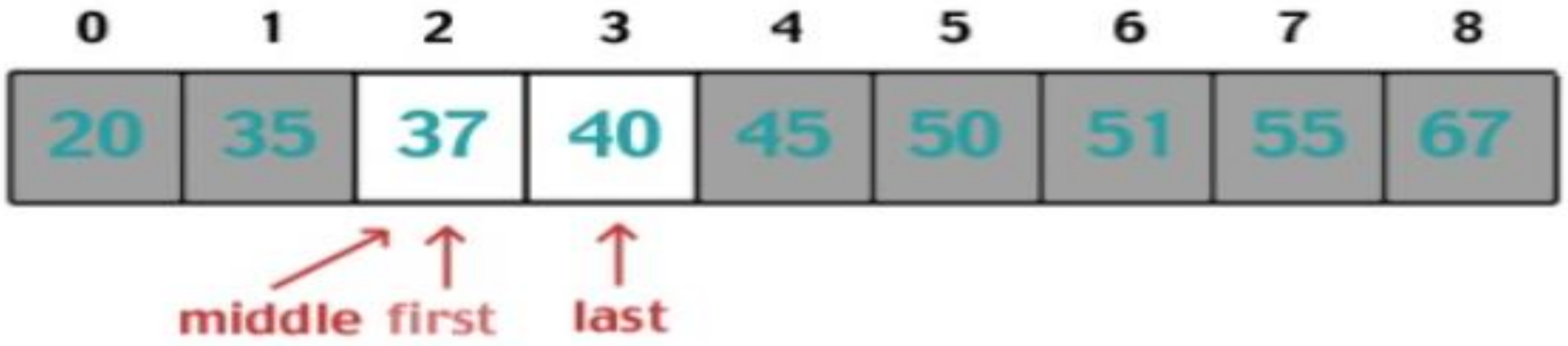
Repeat 2. Calculate middle = (low + high) / 2.
= (0 + 3) / 2 = 1.



If 37 == array[middle] ➔ return middle
Else if 37 < array[middle] ➔ high = middle -1
Else if 37 > array[middle] ➔ low = middle +1

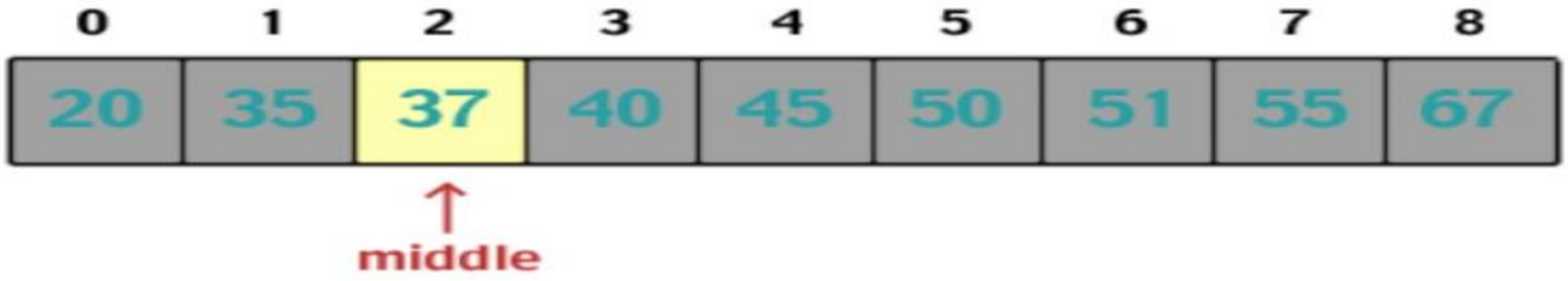
Binary Search

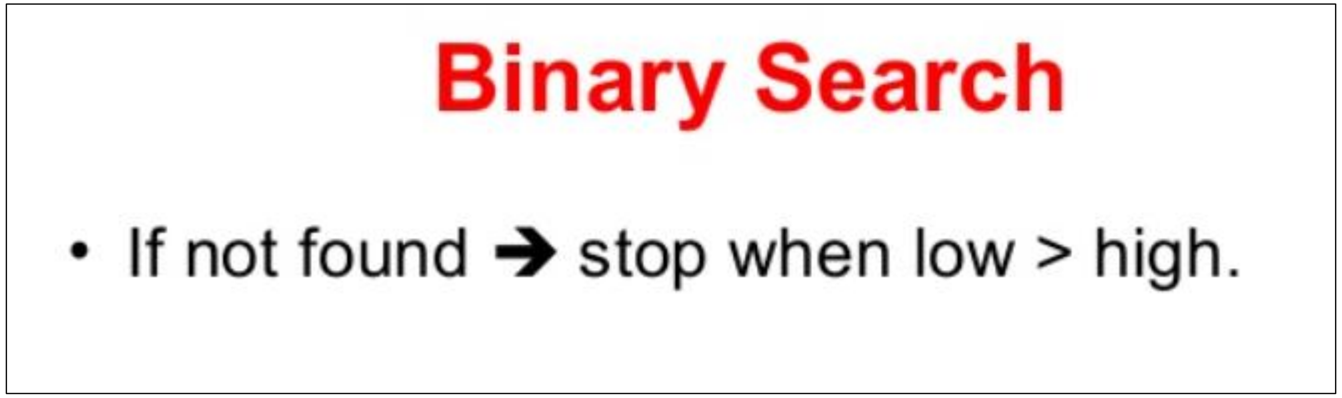
Repeat 2. Calculate middle = (low + high) / 2.
= (2 + 3) / 2 = 2.



If 37 == array[middle] ➔ return middle
Else if 37 < array[middle] ➔ high = middle -1
Else if 37 > array[middle] ➔ low = middle +1

Binary Search





Binary Search

- If not found ➔ stop when low > high.

Home Work

Move all negative numbers to beginning and positive to end with constant extra space

An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers.
Examples :

Input: -12, 11, -13, -5, 6, -7, 5, -3, -6
Output: -12 -13 -5 -7 -3 -6 11 6 5