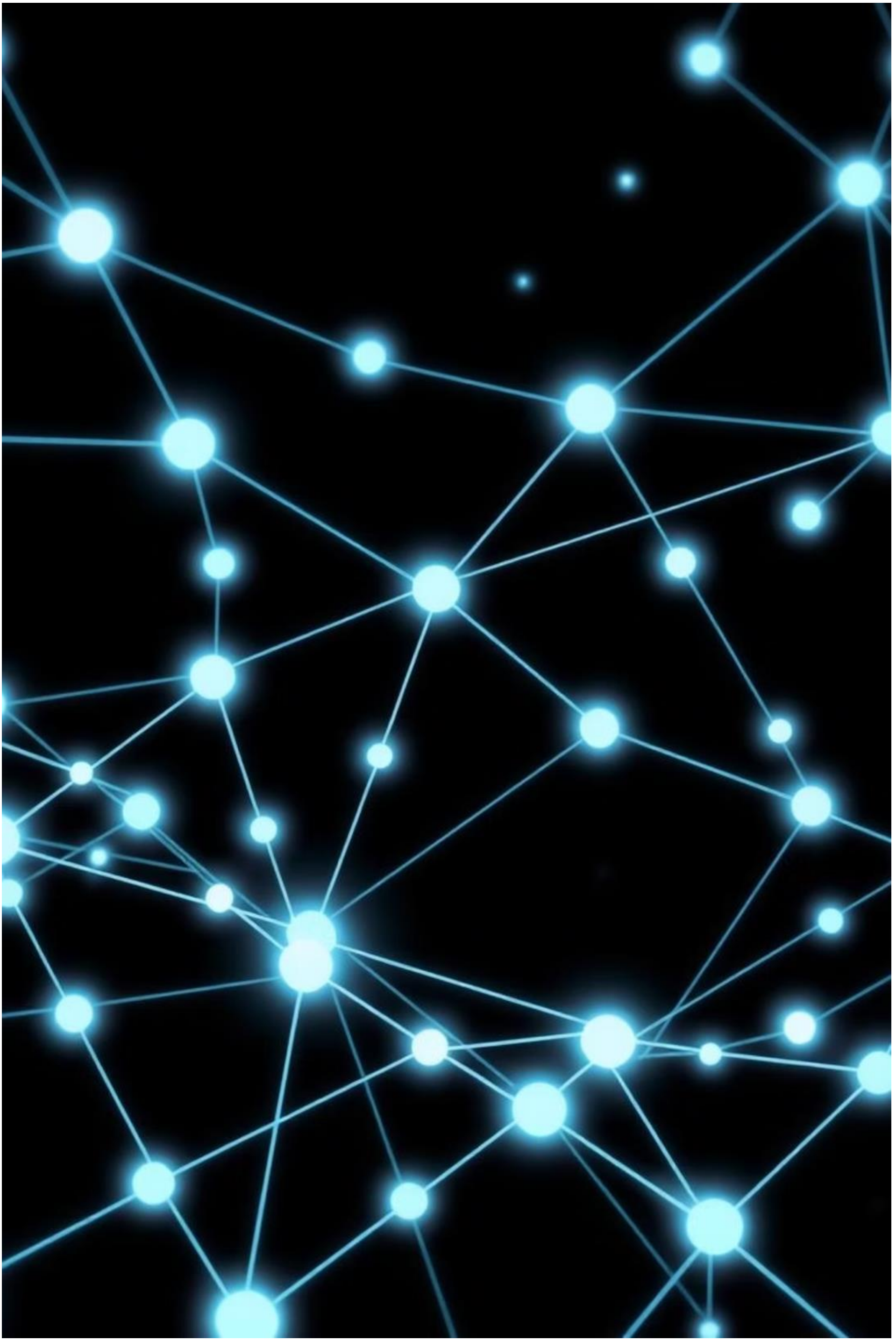


Data Structures

Lecture 8: Searching & Sorting

Instructor:

Md Samsuddoha

Assistant Professor

Dept of CSE, BU

Contents

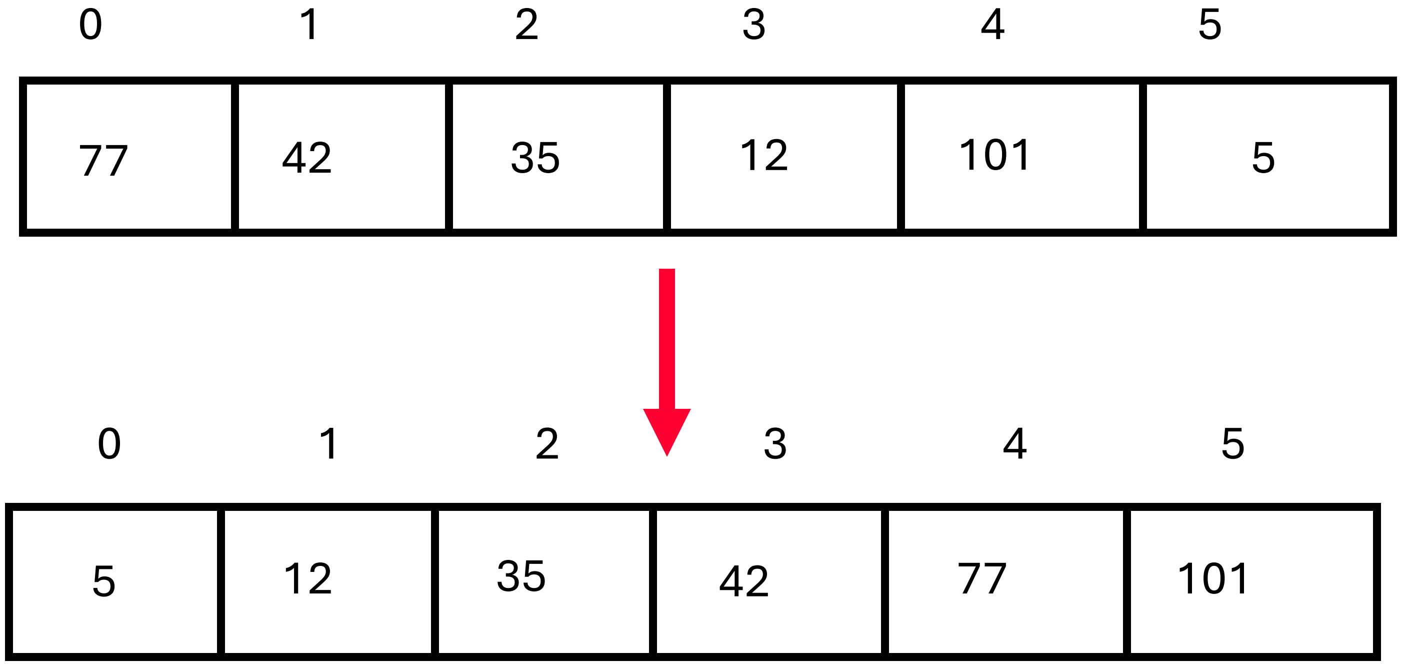
- Sorting
- Bubble sort
- Selection Sort

Sorting

- Sorting in data structures is the process of ***arranging a collection of elements*** in a specific order.
- This order can be numerical (ascending or descending), alphabetical, chronological, or based on any other defined criterion.
- The primary goal of sorting is to organize data in a way that facilitates more efficient ***searching, retrieval, and manipulation***.

Sorting

- **Sorting takes an unordered collection and makes it an ordered one.**

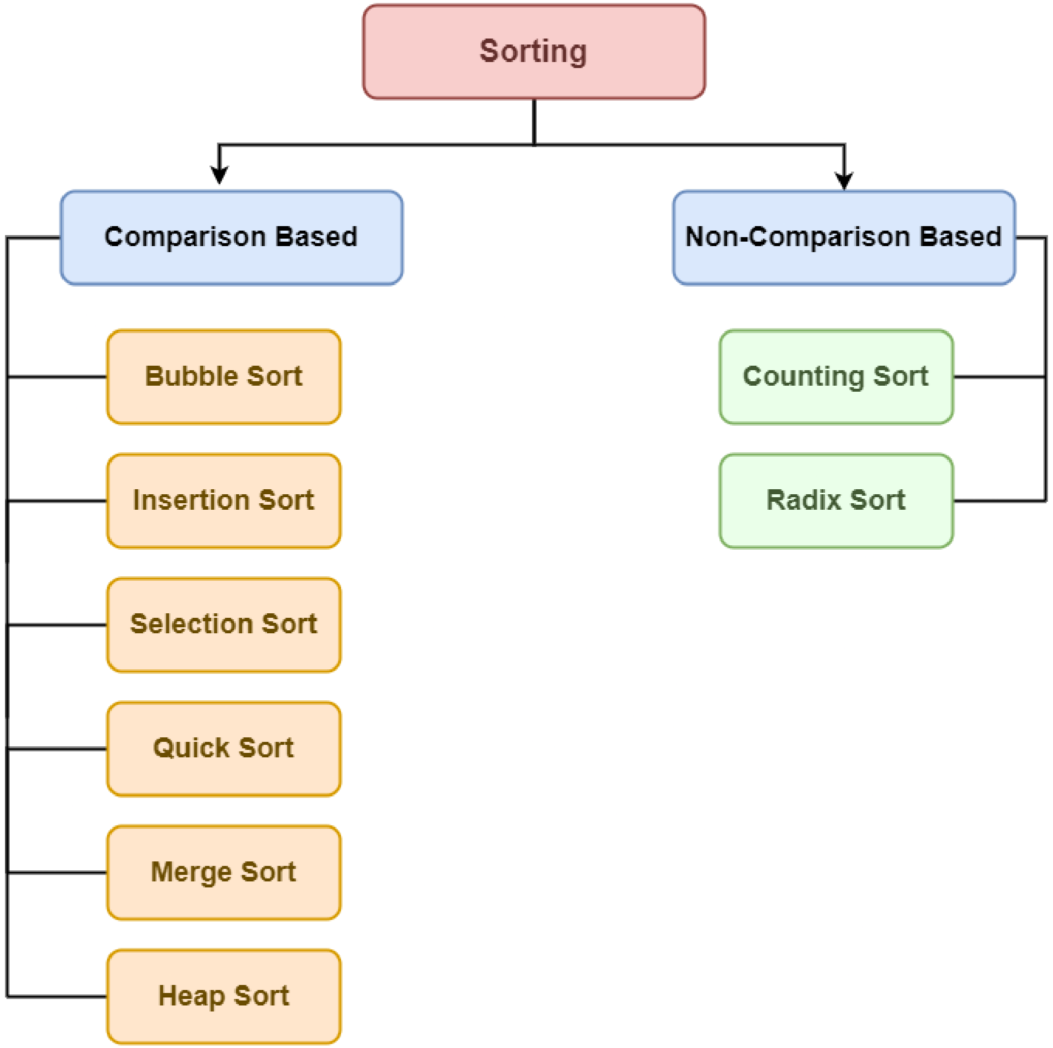


0	1	2	3	4	5
77	42	35	12	101	5

0	1	2	3	4	5
5	12	35	42	77	101

Sorting Algorithms

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- *Counting Sort*
- *Radix Sort*
- *Bucket Sort*



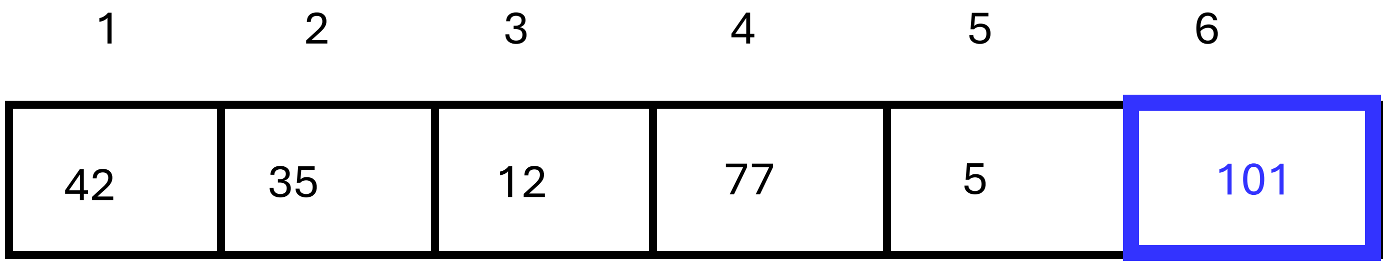
Bubble Sort: "Bubbling Up" the Largest Element

- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**

1	2	3	4	5	6
77	42	35	12	101	5

"Bubbling Up" the Largest Element

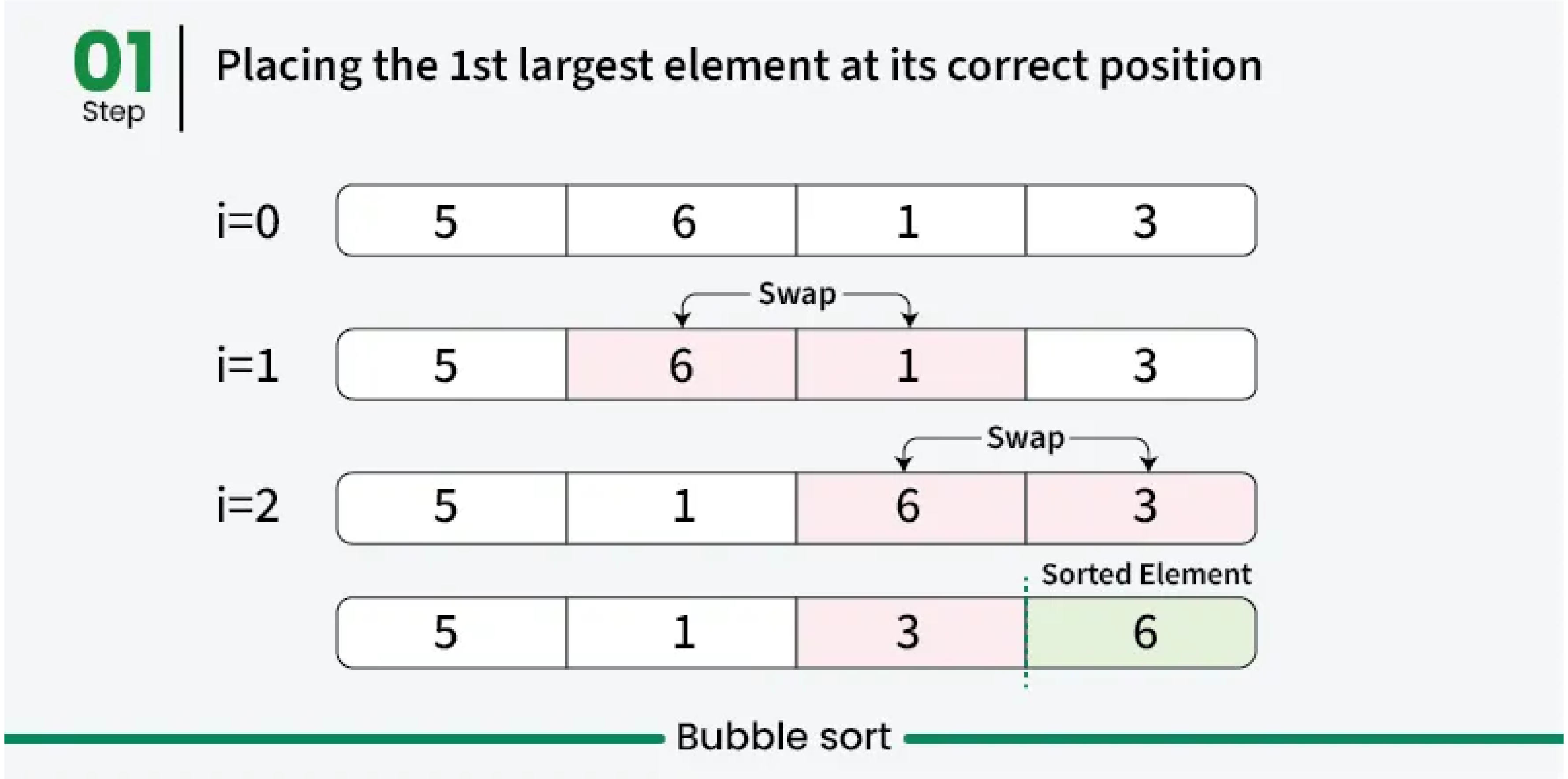
- **Traverse a collection of elements**
  - **Move from the front to the end**
  - **“Bubble” the largest value to the end using pair-wise comparisons and swapping**



1	2	3	4	5	6
42	35	12	77	5	101

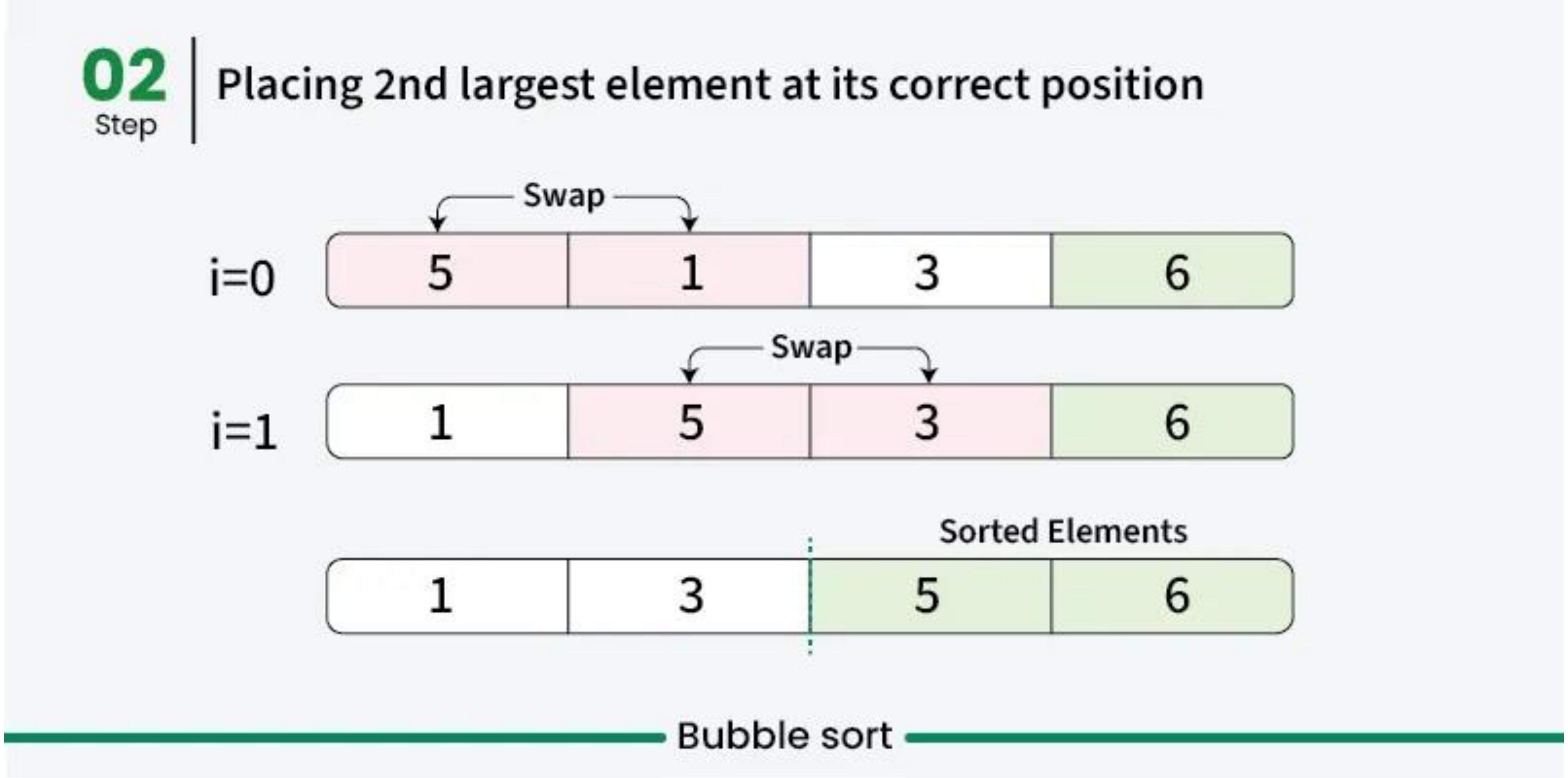
Largest value correctly placed

Bubble Sort (Simulation)



01 Step | Placing the 1st largest element at its correct position

Bubble Sort (Simulation)

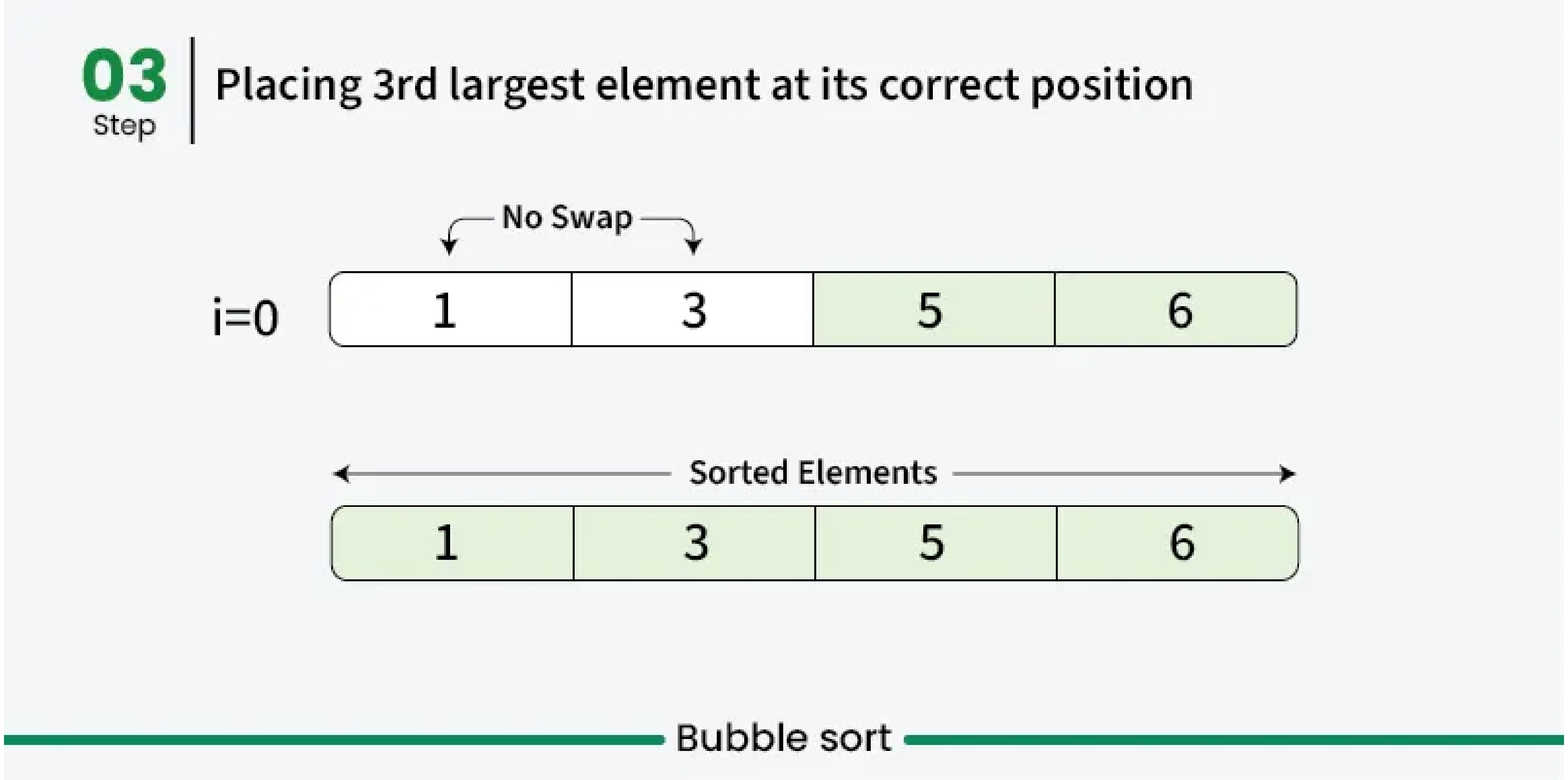


02 Step

Placing 2nd largest element at its correct position

i=0	5	1	3	6
i=1	1	5	3	6
	1	3	5	6

Bubble Sort (Simulation)



03 Step | Placing 3rd largest element at its correct position

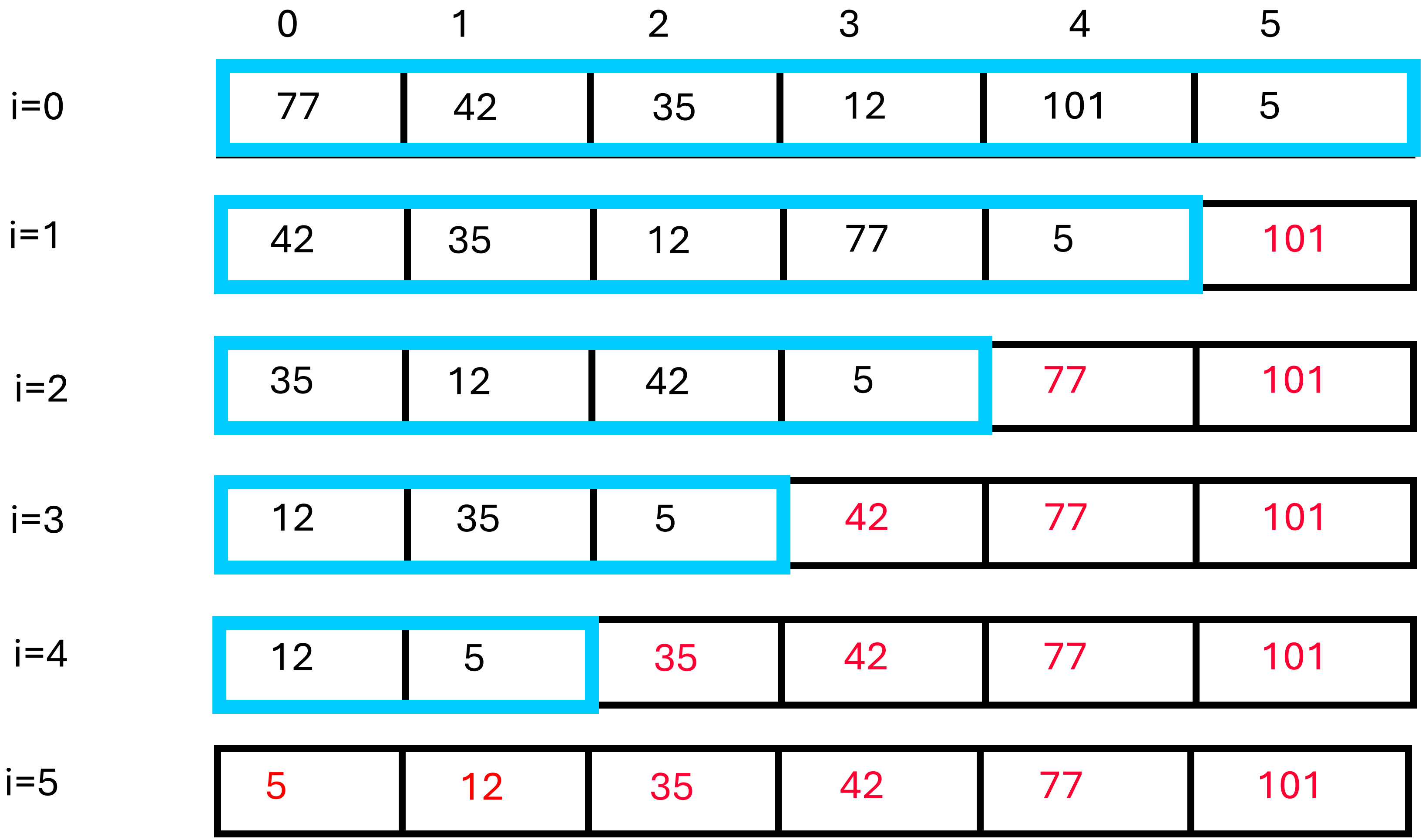
“Bubbling” All the Elements

0	1	2	3	4	5
42	35	12	77	101	5

N - 1

42	35	12	77	5	101
35	12	42	5	77	101
12	35	5	42	77	101
12	5	35	42	77	101
5	12	35	42	77	101

Reducing the Number of Comparisons

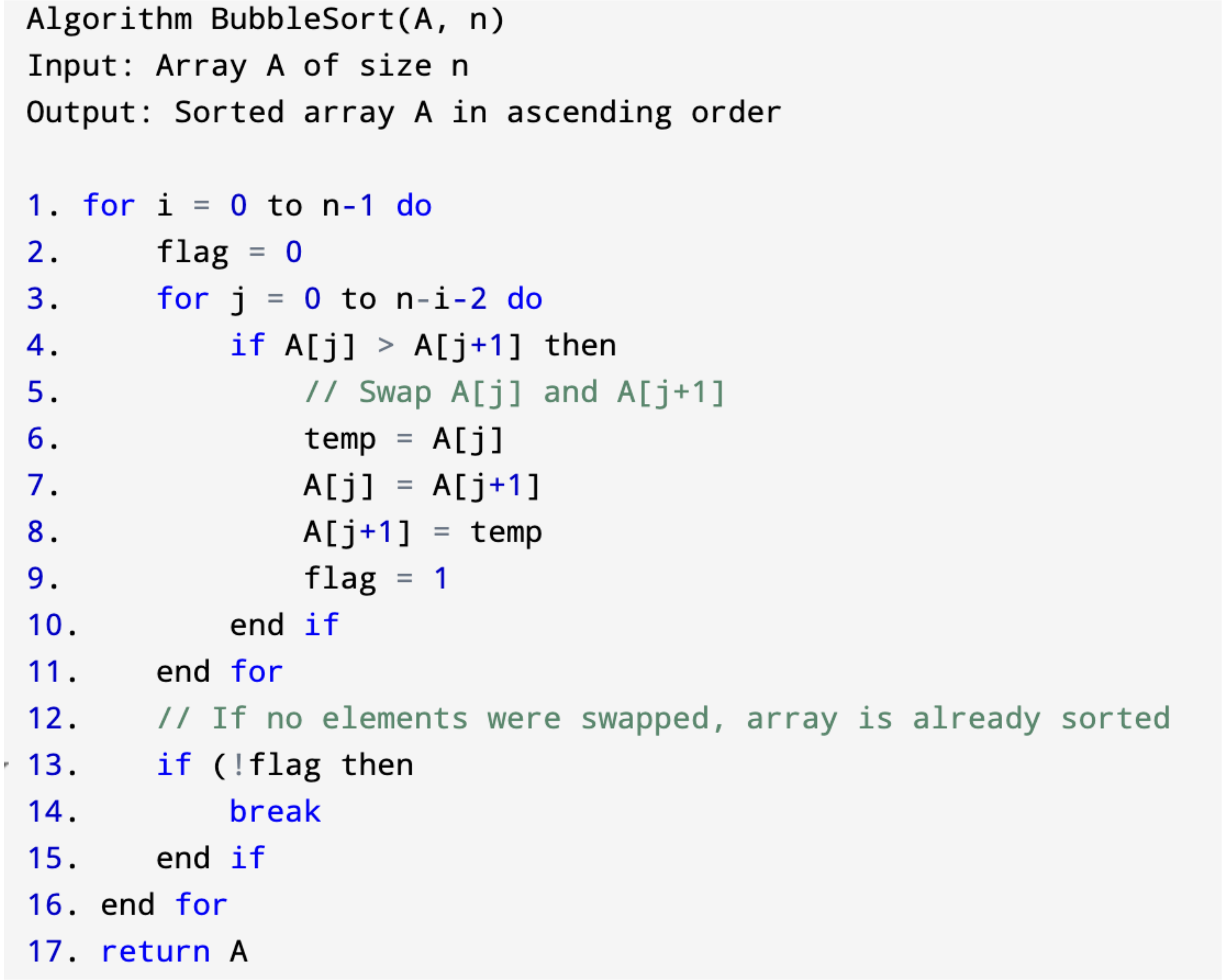


	0	1	2	3	4	5
i=0	77	42	35	12	101	5
i=1	42	35	12	77	5	101
i=2	35	12	42	5	77	101
i=3	12	35	5	42	77	101
i=4	12	5	35	42	77	101
i=5	5	12	35	42	77	101

Complexity

- Worst case: array is in **reverse order**.
- Number of Iterations: N-1
- Number of comparisons in each pass:
    - 1st pass → n-1 comparisons
    - 2nd pass → n-2 comparisons
    - …
    - Last pass → 1 comparison
- Total comparisons = (n-1) + (n-2) + … + 1 = n(n-1)/2
- **Worst-case time complexity:** $O(n^2)$

Pseudocode



```
Algorithm BubbleSort(A, n)
Input: Array A of size n
Output: Sorted array A in ascending order

for i = 0 to n-1 do
    flag = 0
    for j = 0 to n-i-2 do
        if A[j] > A[j+1] then
            // Swap A[j] and A[j+1]
            temp = A[j]
            A[j] = A[j+1]
            A[j+1] = temp
            flag = 1
        end if
    end for
    // If no elements were swapped, array is already sorted
    if (!flag then
        break
    end if
end for
return A
```

Coding

- Write a C program for Bubble sort.

Selection Sort

- Selection Sort is a simple comparison-based sorting algorithm.
- It repeatedly selects the **smallest (or largest)** element from the unsorted part of the array and **swaps** it with the first unsorted element.

Simulation

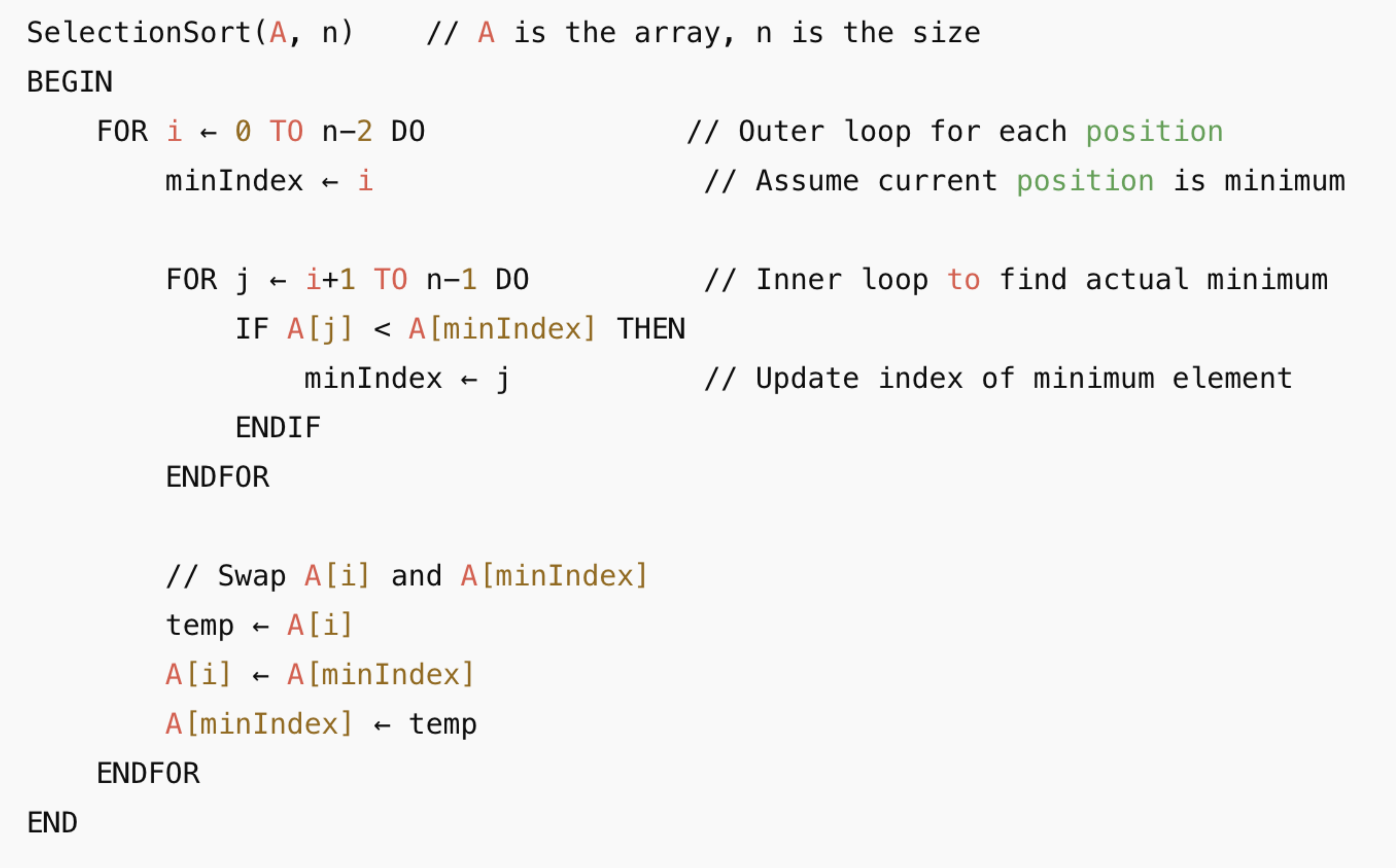
0	1	2	3	4	5	
42	35	12	77	101	5	
5	35	12	77	101	42	Pass 1
5	12	35	77	101	42	Pass 2
5	12	35	77	101	42	Pass 3
5	12	35	42	101	77	Pass 4
5	12	35	42	77	101	Pass 5

N - 1

Complexity

- Worst case: array is in **reverse order**.
- Number of Iterations: N-1
- Number of comparisons in each pass:
  - 1st pass → n-1 comparisons
  - 2nd pass → n-2 comparisons
  - …
  - Last pass → 1 comparison
- Total comparisons = (n-1) + (n-2) + … + 1 = n(n-1)/2
- **Worst-case time complexity:** $O(n^2)$

Pseudocode of Selection Sort



```
SelectionSort(A, n)    // A is the array, n is the size
BEGIN
    FOR i ← 0 TO n−2 DO               // Outer loop for each position
        minIndex ← i                   // Assume current position is minimum

        FOR j ← i+1 TO n−1 DO          // Inner loop to find actual minimum
            IF A[j] < A[minIndex] THEN
                minIndex ← j           // Update index of minimum element
            ENDIF
        ENDFOR

        // Swap A[i] and A[minIndex]
        temp ← A[i]
        A[i] ← A[minIndex]
        A[minIndex] ← temp
    ENDFOR
END
```

Exercises

- Write a program to sort an array of **N** integers in **ascending order** using **bubble sort**.
- Modify the bubble sort program to sort the array in **descending order**.
- Write a program for bubble sort to count the number of swaps needed to sort the array.
- You have a list of students with roll numbers and scores. You have to find the student with the highest score and display their roll number and score.
- Write a program to sort an array of **N** integers in **ascending order** using **selection sort**.

References

- **Chapter 10: Data Structures using C** by E. Balagurusamy
- Visit the site for live visualization: https://visualgo.net/

Thank You