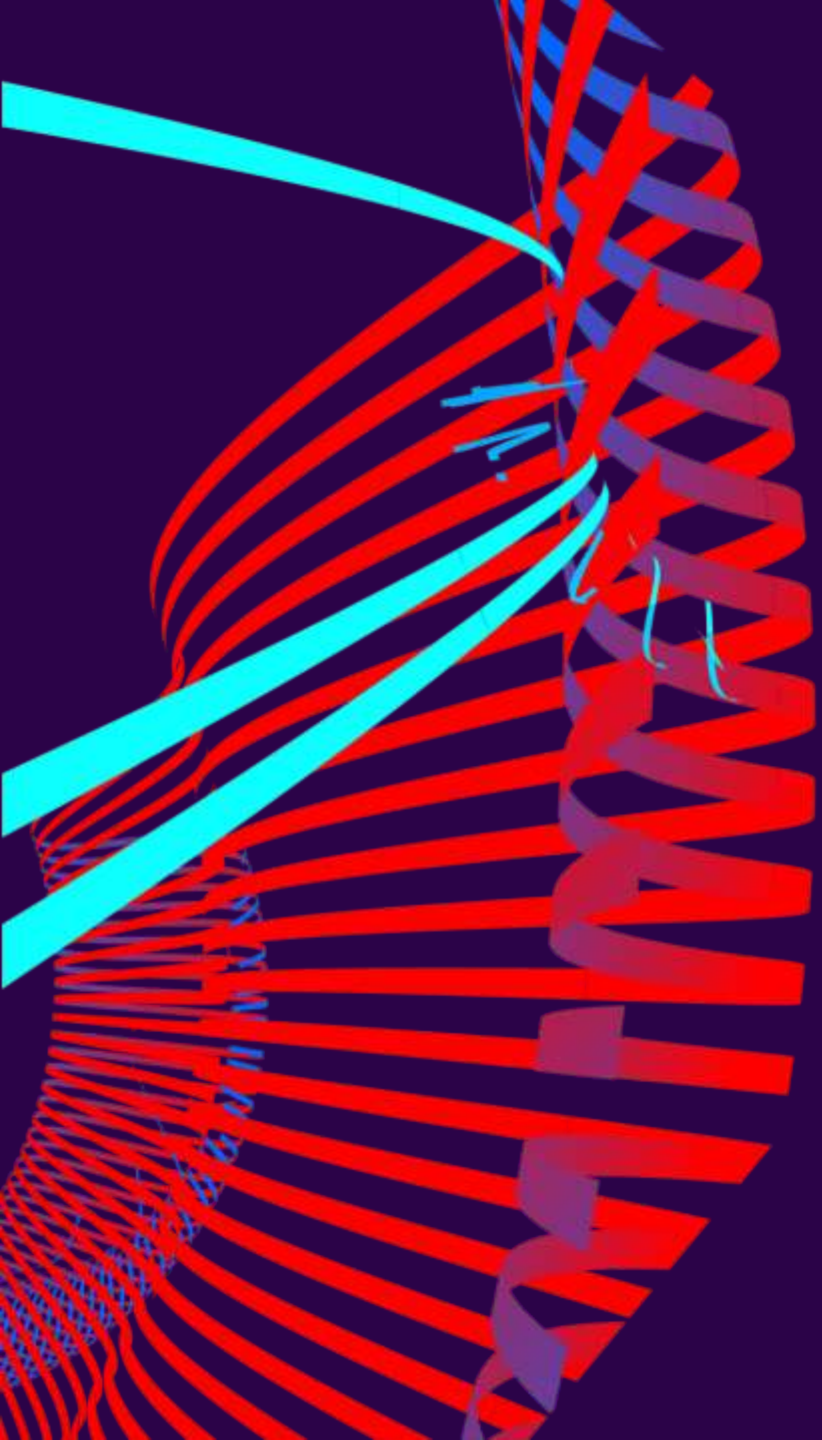
SELECTION & BUBBLE SORT

ABDULLAHI JEYLANI 2111502215

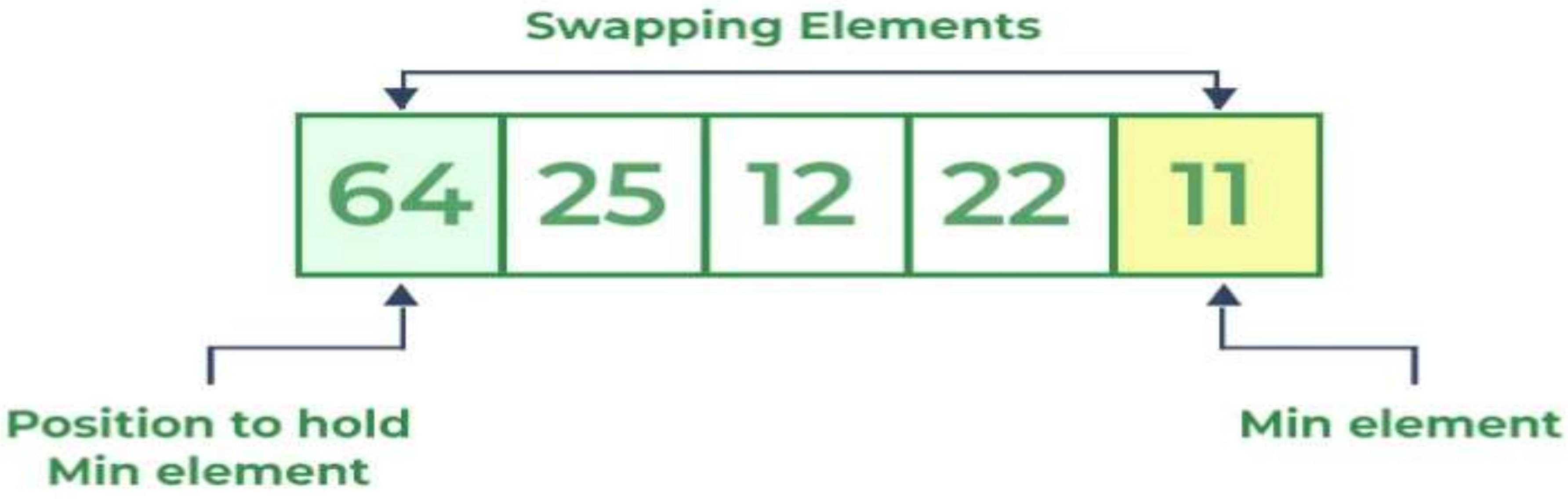
MUSTAPHA YAHAYA 201502220

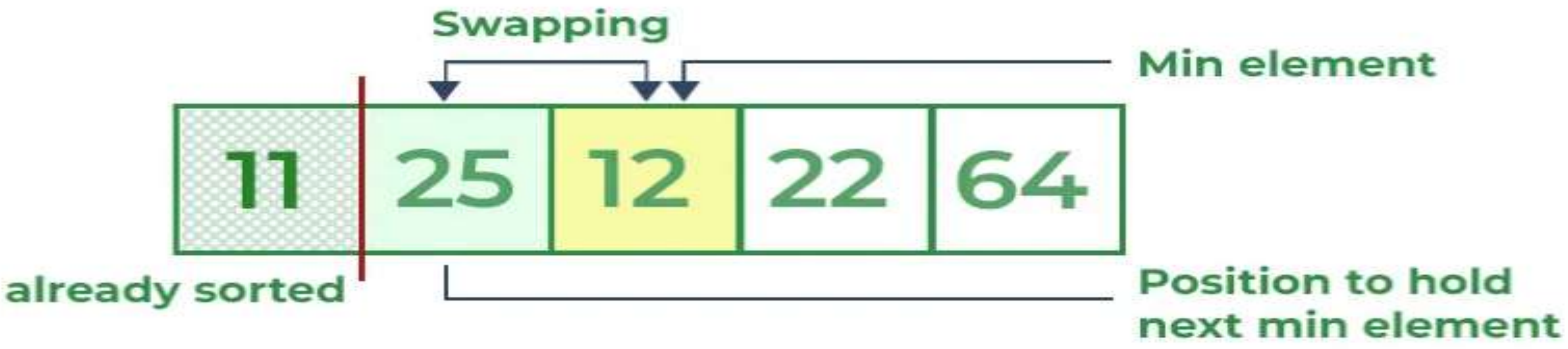


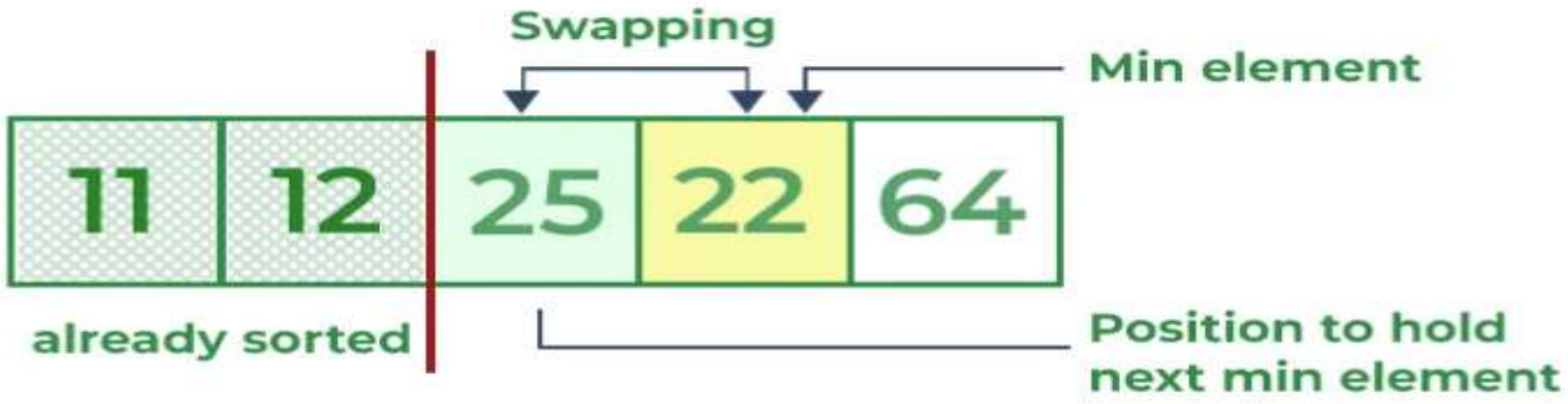
SELECTION SORT ALGORITHM

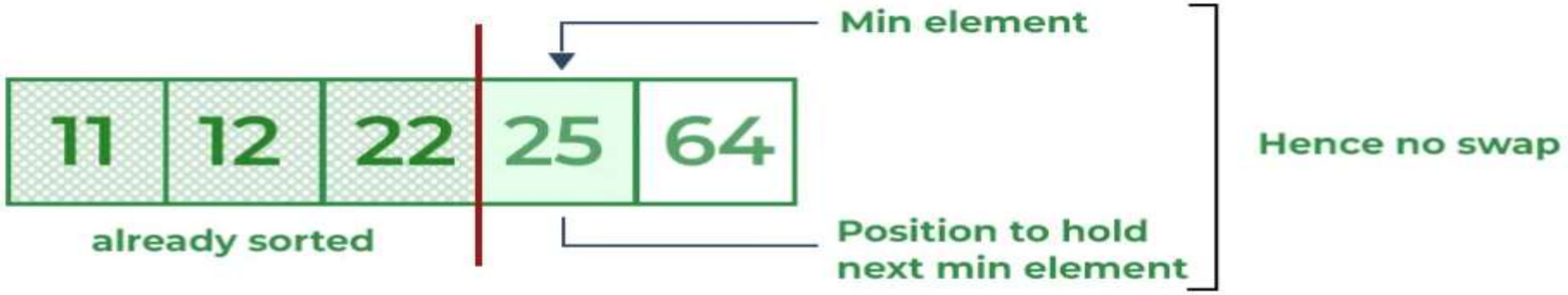
SELECTION SORT

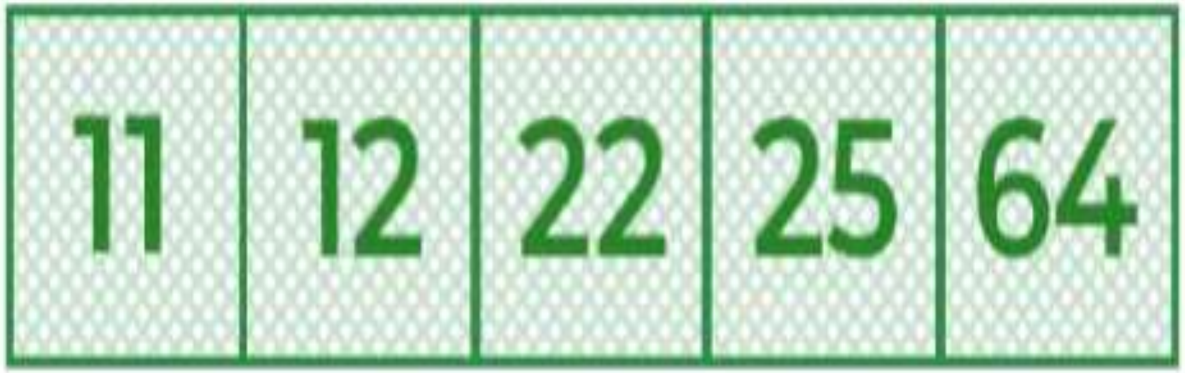
- Selection-sort algorithm is an in-place comparison based algorithm in which the list is divided into two parts the sorted part at the left end and the unsorted at the right end
- The smallest element is selected from unsorted array and swapped with the left most element and the elements become a part of the sorted array


Swapping Elements
64
25
12
22
11
Position to hold Min element
Min element


Swapping
Min element
11
25
12
22
64
already sorted
Position to hold next min element

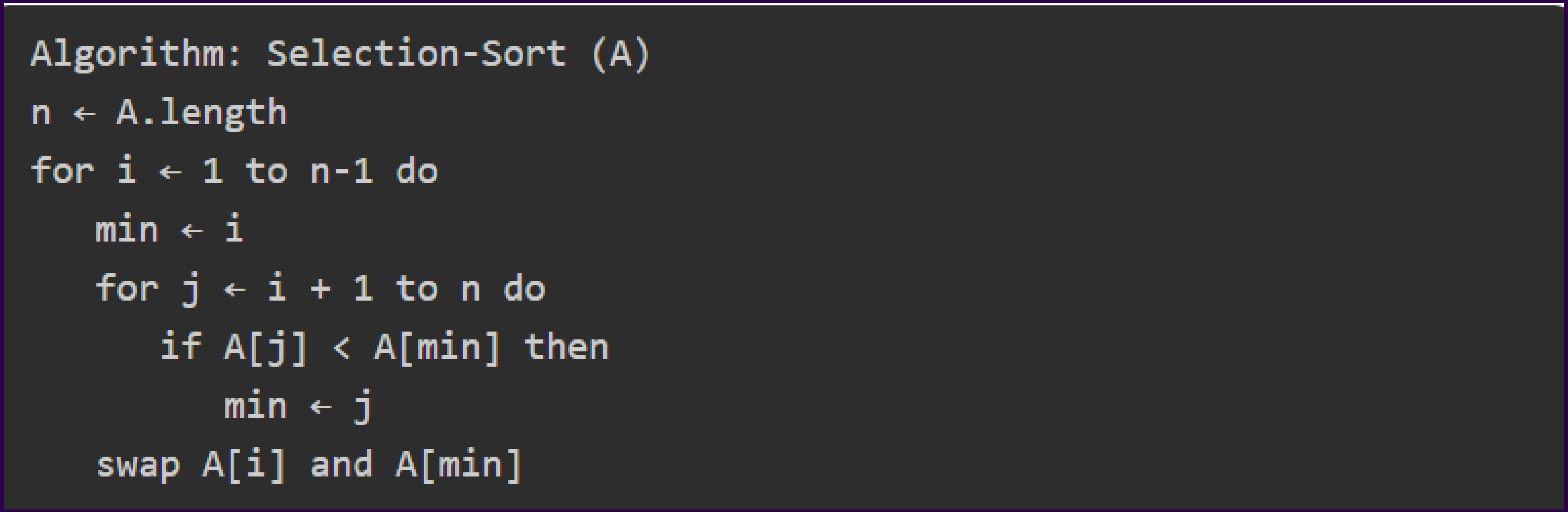

Swapping
Min element
11
12
25
22
64
already sorted
Position to hold
next min element


Min element
11
12
22
25
64
Hence no swap
already sorted
Position to hold
next min element



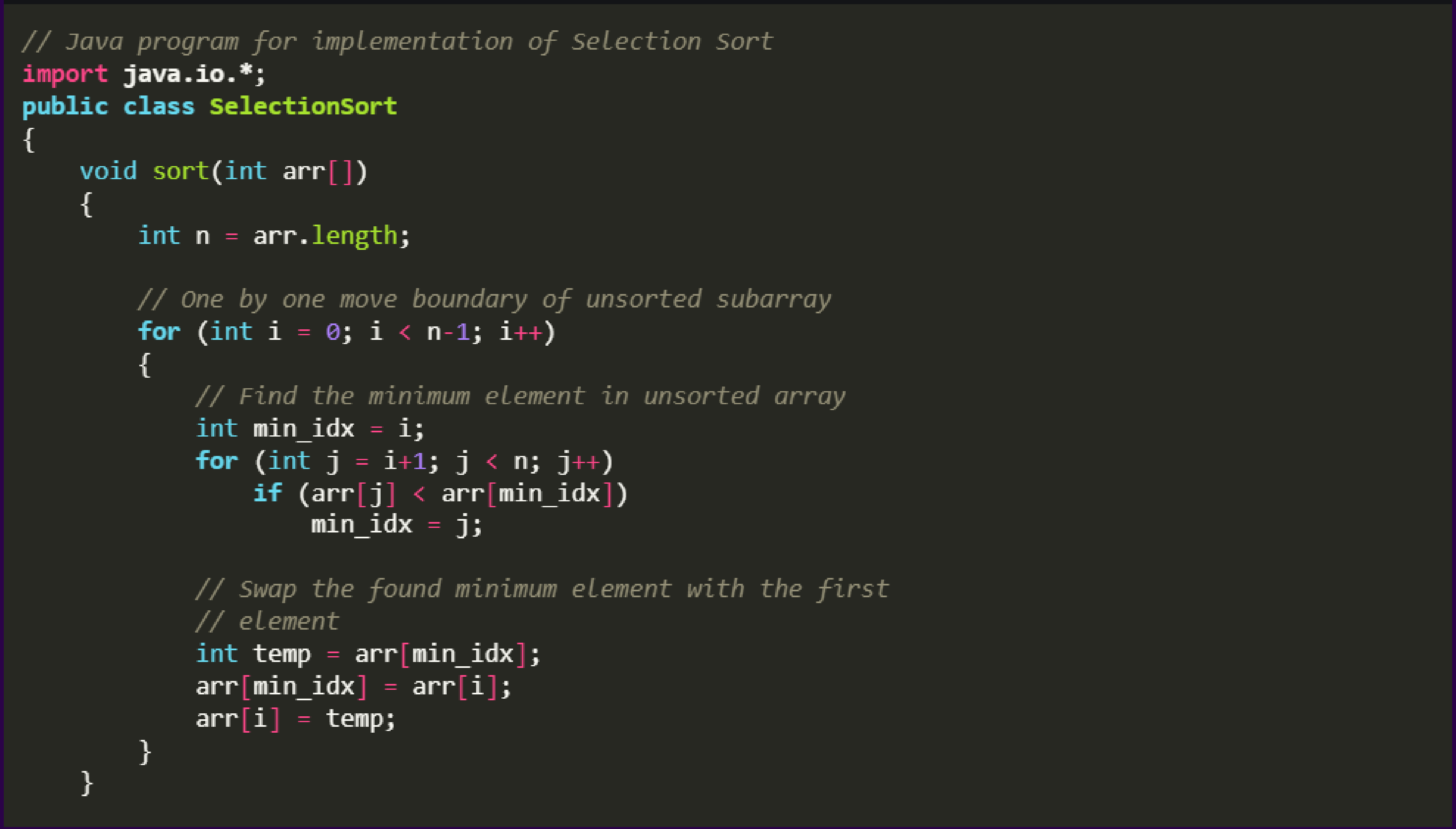
Sorted array

PSEUDO CODE



```
Algorithm: Selection-Sort (A)
n ← A.length
for i ← 1 to n-1 do
   min ← i
   for j ← i + 1 to n do
      if A[j] < A[min] then
         min ← j
   swap A[i] and A[min]
```

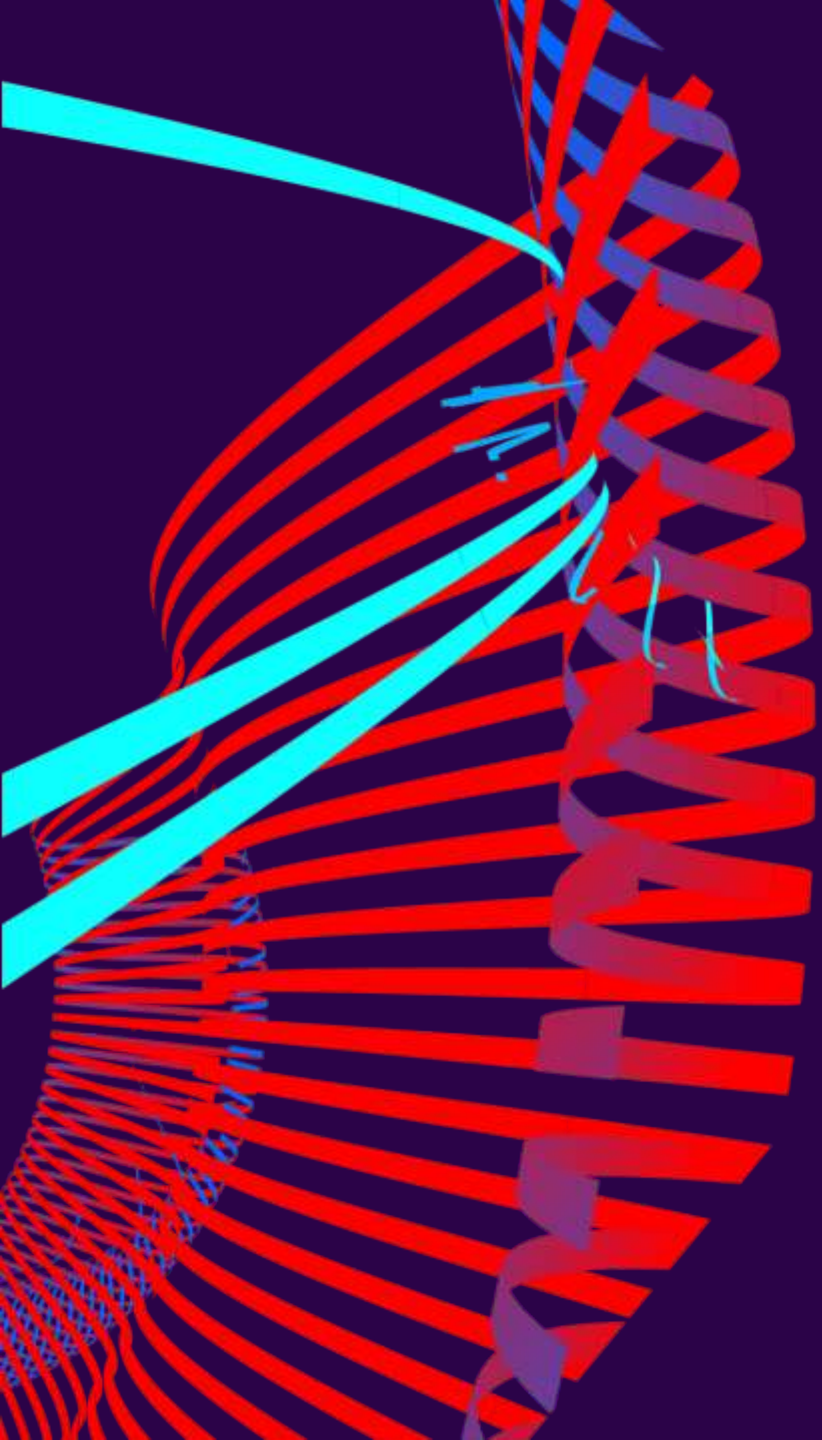
JAVA CODE



```java
// Java program for implementation of Selection Sort
import java.io.*;
public class SelectionSort
{
    void sort(int arr[])
    {
        int n = arr.length;

        // One by one move boundary of unsorted subarray
        for (int i = 0; i < n-1; i++)
        {
            // Find the minimum element in unsorted array
            int min_idx = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            // Swap the found minimum element with the first
            // element
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }
```



BUBBLE SORT ALGORITHM

BUBBLE SORT

HISTORY:

The earlier description of the bubble sort algorithm was in a 1956 paper by mathematician and actuary Edward harry friend, sorting on electronic computer systems, published in the third issue of the third volume of the journal of the association for computing machinery(ACM),as a “sorting exchange algorithm“. Friend described the fundamentals of the algorithm, and, although initially his paper went unnoticed, some years later, it was the rediscovery by many computer scientists, including Kenneth E. Iverson who coined its current name

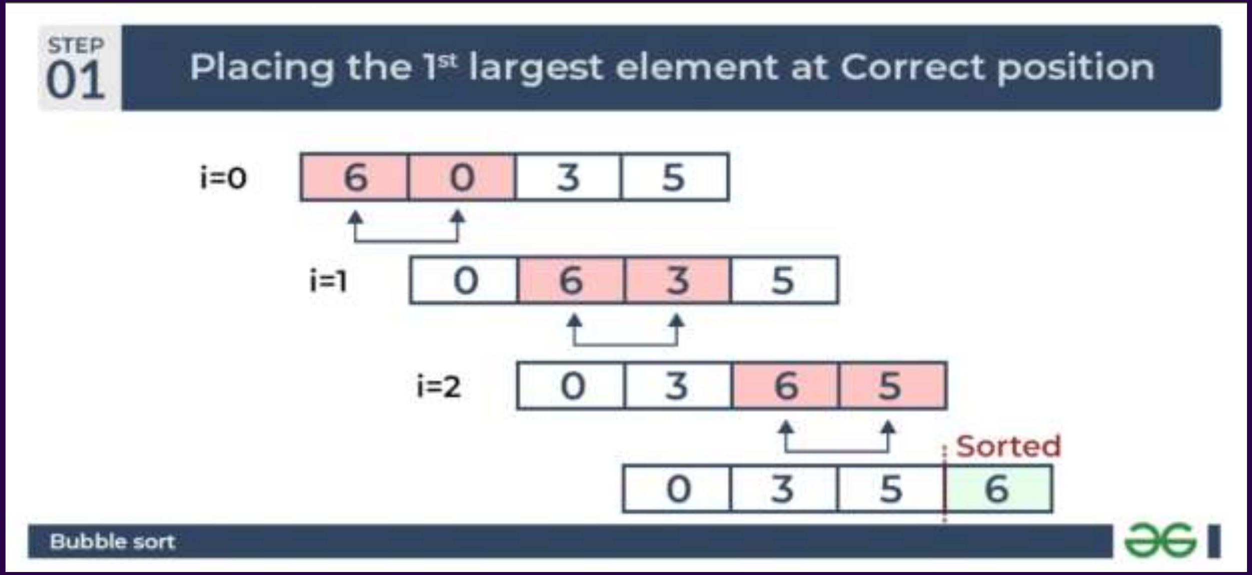
BUBBLE SORT

PERFORMANCE:

Bubble sort has the worst-case and average complexity of $O(n^2)$. Most practical sorting algorithms have substantially better worst-case or average complexity, often $O(n\log n)$. even other $O(n^2)$ sorting algorithms, such as insertion sort, generally run faster than bubble sort and are no more complex. For this reason, bubble sort is rarely used in practice. like insertion sort, bubble sort is adaptive, which can give it an advantage over algorithms like quick sort. This means that it, may outperform those algorithms in cases where the list is already mostly sorted (having a small number of inversions), despite the fact that it has worse average-case time complexity. For example, bubble sort is $O(n)$ on a list that is already sorted, while quicksort would still perform its entire $O(n\log n)$ sorting process. Bubble sort also interacts poorly with modern CPU hardware. It produces at least twice as many writes as insertion sort, twice as many caches’ misses, and asymptotically more branch mispredictions

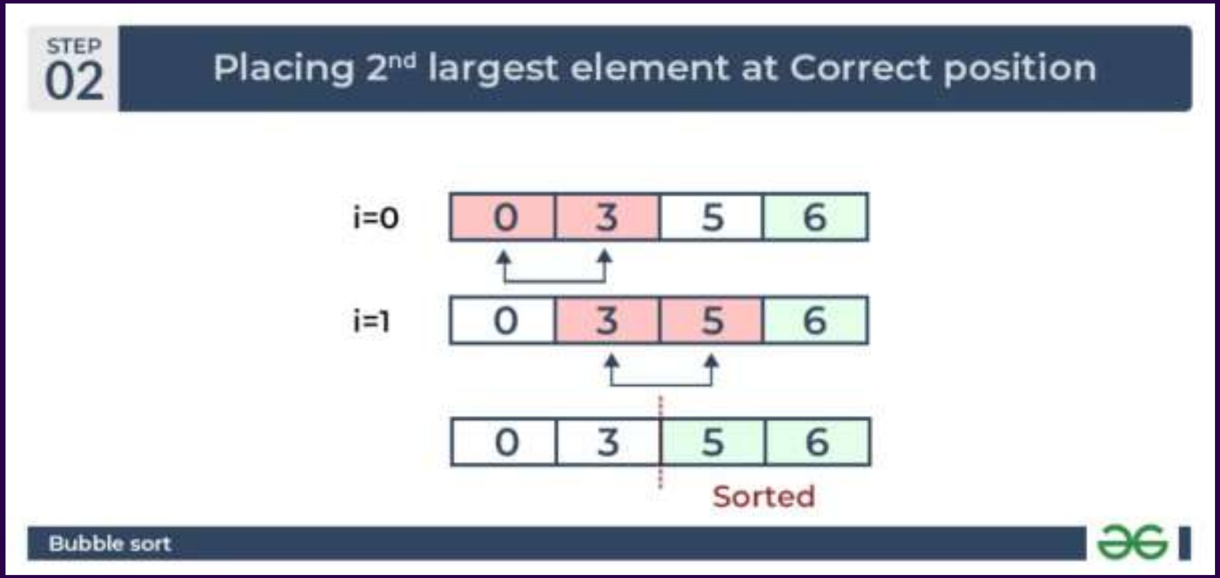
HOW IT WORKS

- FIRST LARGEST ELEMENT IS PLACED AT THE END OF THE ARRAY



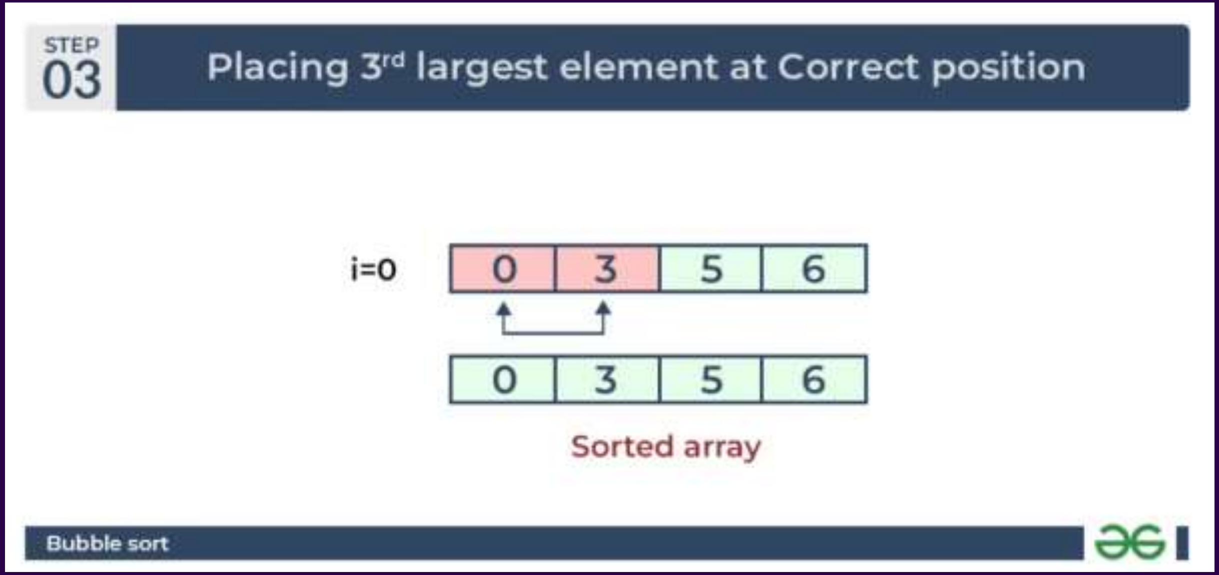
HOW IT WORKS

- SECOND LARGEST ELEMENT IS PLACED AT THE RIGHT POSITION

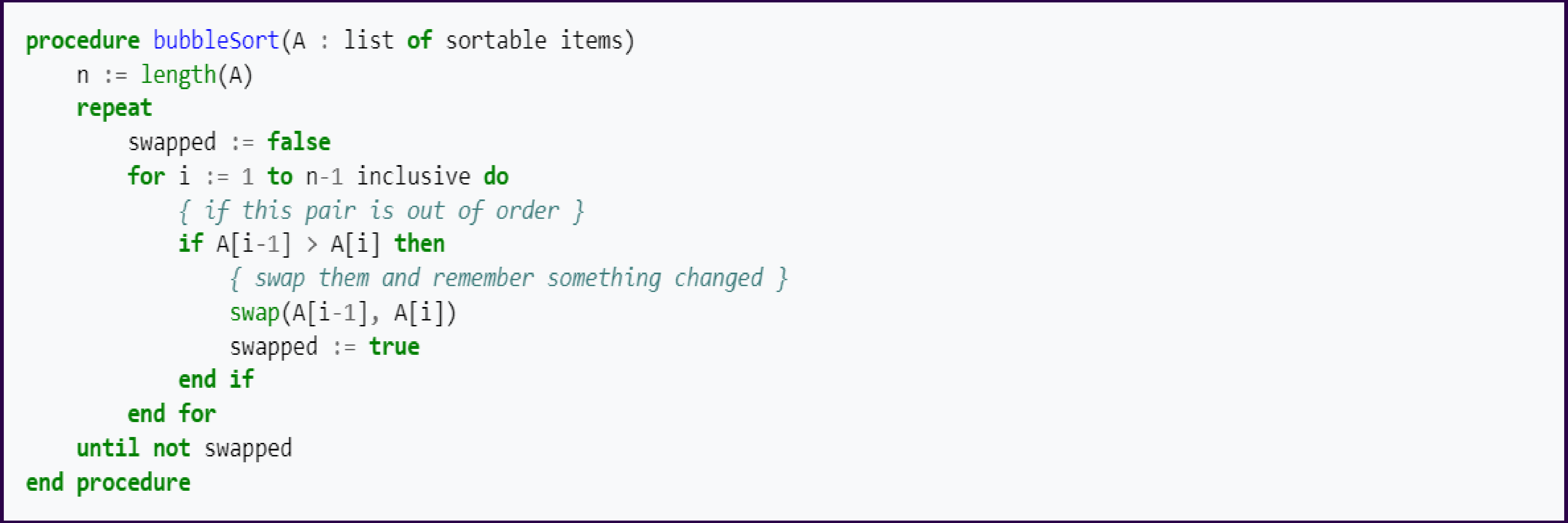


HOW IT WORKS

- THIRD LARGEST

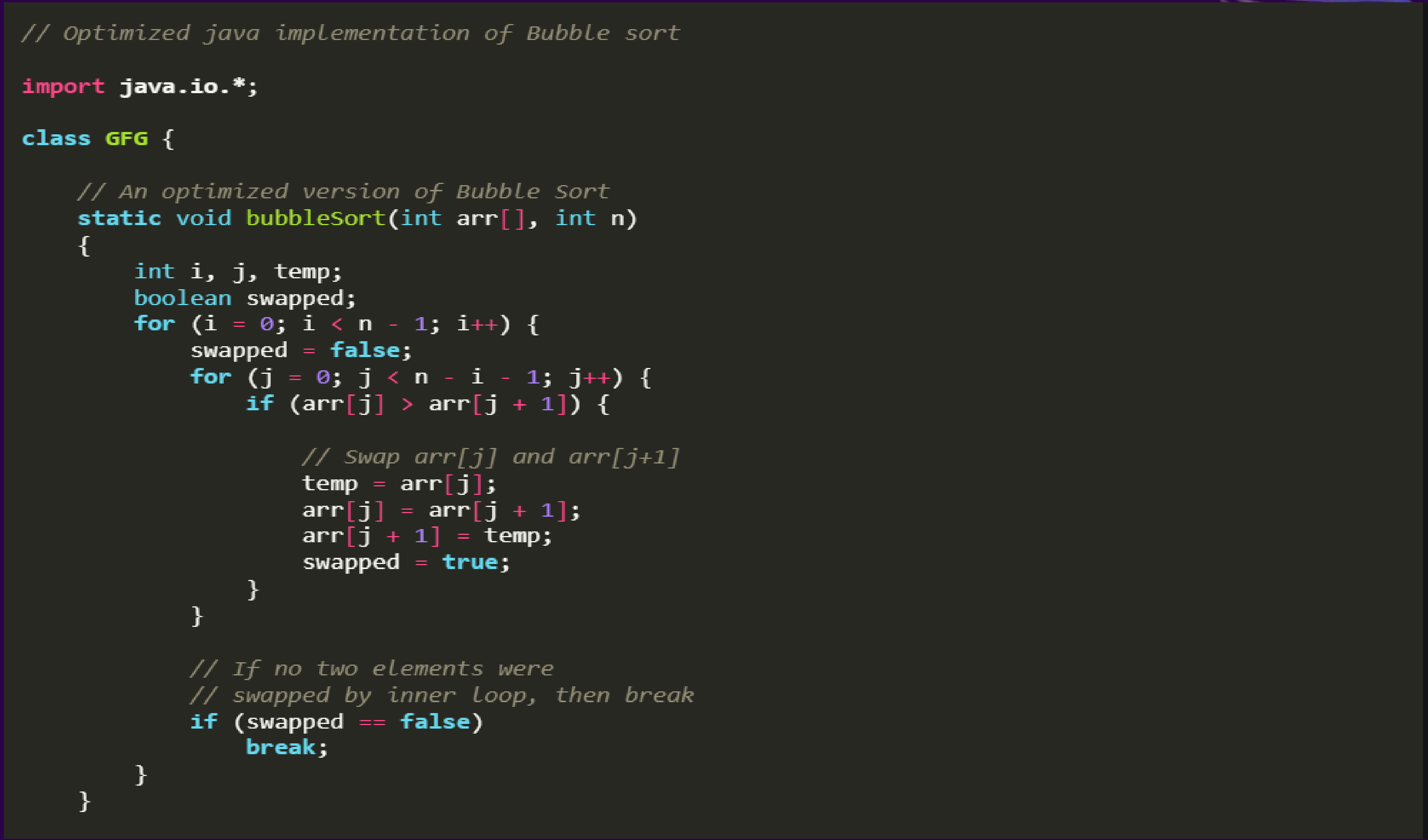


PSEUDO CODE



```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
    repeat
        swapped := false
        for i := 1 to n-1 inclusive do
            { if this pair is out of order }
            if A[i-1] > A[i] then
                { swap them and remember something changed }
                swap(A[i-1], A[i])
                swapped := true
            end if
        end for
    until not swapped
end procedure
```

JAVA CODE



```java
// Optimized java implementation of Bubble sort

import java.io.*;

class GFG {

    // An optimized version of Bubble Sort
    static void bubbleSort(int arr[], int n)
    {
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++) {
            swapped = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {

                    // Swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // If no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }
```

COMPARISON

SELECTION SORT

- Selection sort involves finding the smallest element in the list and swapping it with the first element in the unsorted portion of the list
- Lowest element is moving towards the left most of the array
- The complexity time of the best, average and worst cases are $O(n^2)$.

BUBBLE SORT

- Bubble sort involves comparing and potentially swapping two adjacent elements. If the elements are in the correct order, we move to the next pair.
- The highest element is moving towards the right most of the array.
- The complexity time of the best case is $O(n)$ and the average and worst case are $O(n^2)$.

REFERENCES

https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm

https://www.khanacademy.org/computing/computer-science/algorithms/sorting-algorithms/a/selection-sort-pseudocode

https://www.geeksforgeeks.org/selection-sort/

https://www.geeksforgeeks.org/bubble-sort/

https://en.wikipedia.org/wiki/Bubble_sort#:~:text=10%20External%20links-,History,as%20a%20%22Sorting%20exchange%20algorithm%22

THANK YOU