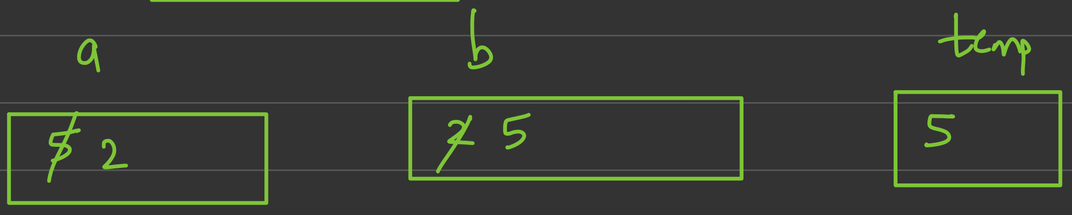
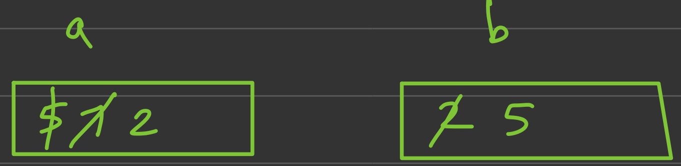
8 Apr 2023



① $temp = a$
$a = b$
$b = temp$
} Using temp variable

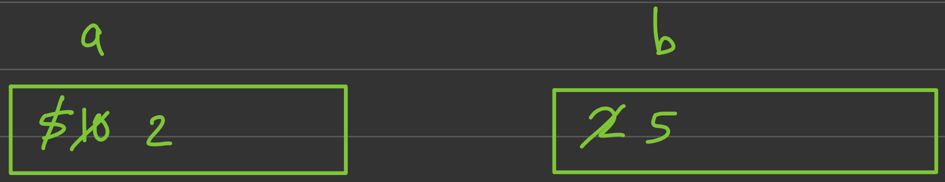
② without using a temporary variable

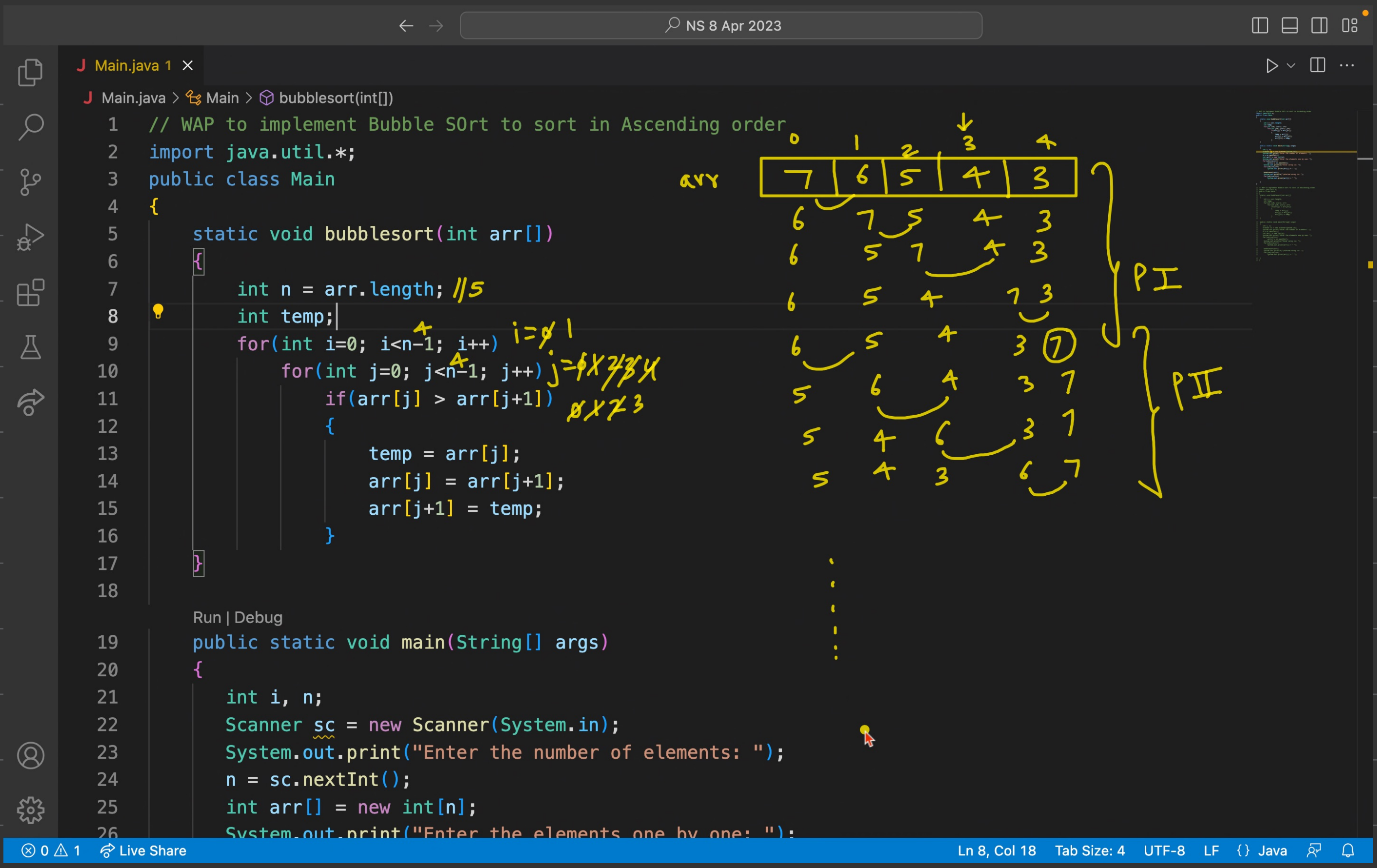
$a = a + b$
$b = a - b$
$a = a - b$



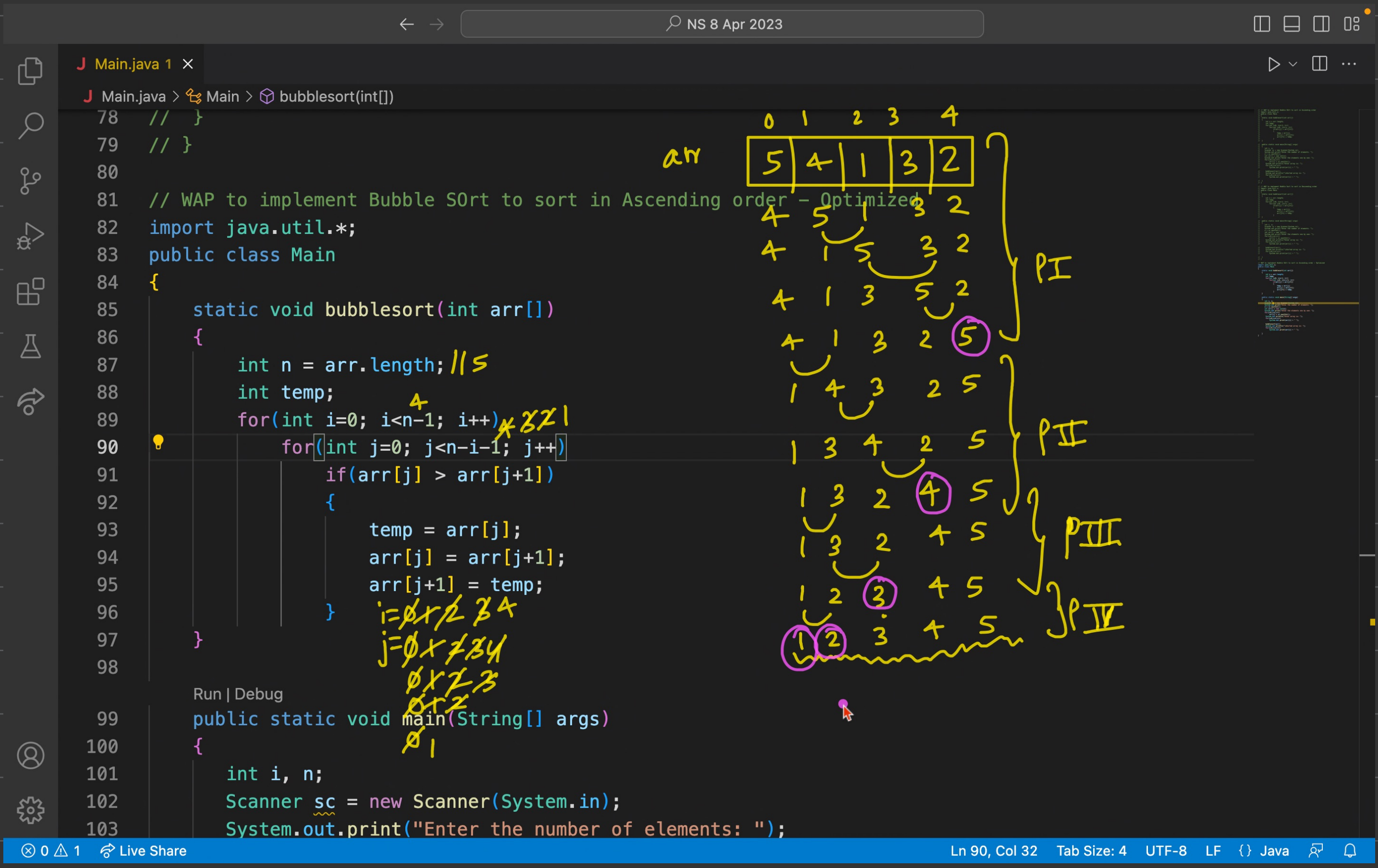
③ — 〃 —

$a = a \times b$
$b = a / b$
$a = a / b$



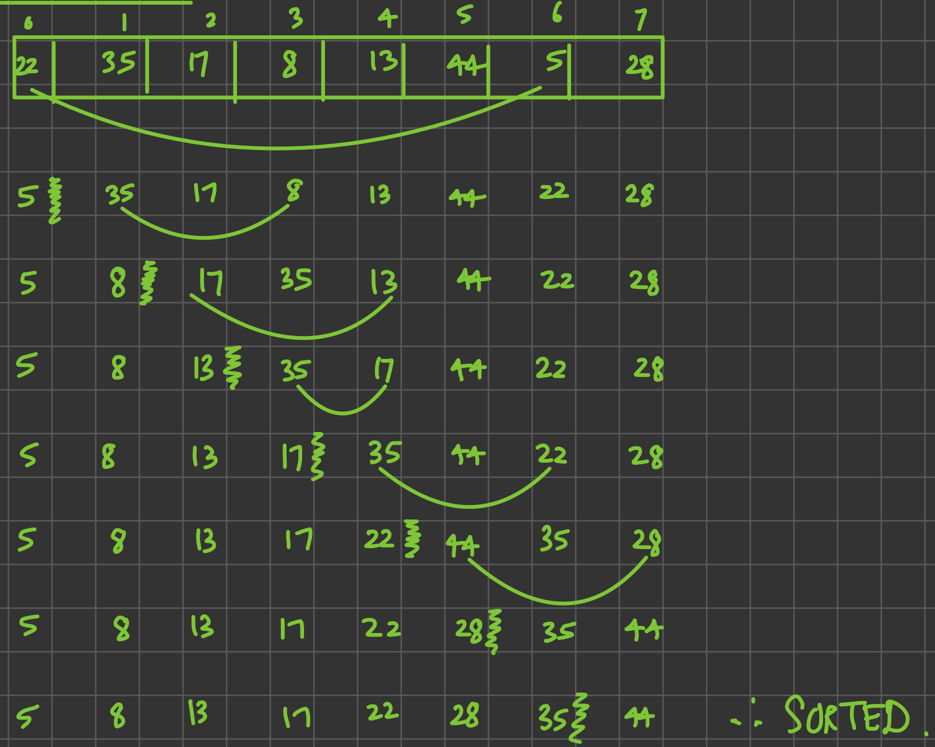


```
// WAP to implement Bubble SOrt to sort in Ascending order
import java.util.*;
public class Main
{
    static void bubblesort(int arr[])
    {
        int n = arr.length;
        int temp;
        for(int i=0; i<n-1; i++)
            for(int j=0; j<n-1; j++)
                if(arr[j] > arr[j+1])
                {
                    temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }

    Run | Debug
    public static void main(String[] args)
    {
       int i, n;
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter the number of elements: ");
       n = sc.nextInt();
       int arr[] = new int[n];
       System.out.print("Enter the elements one by one: ");
```



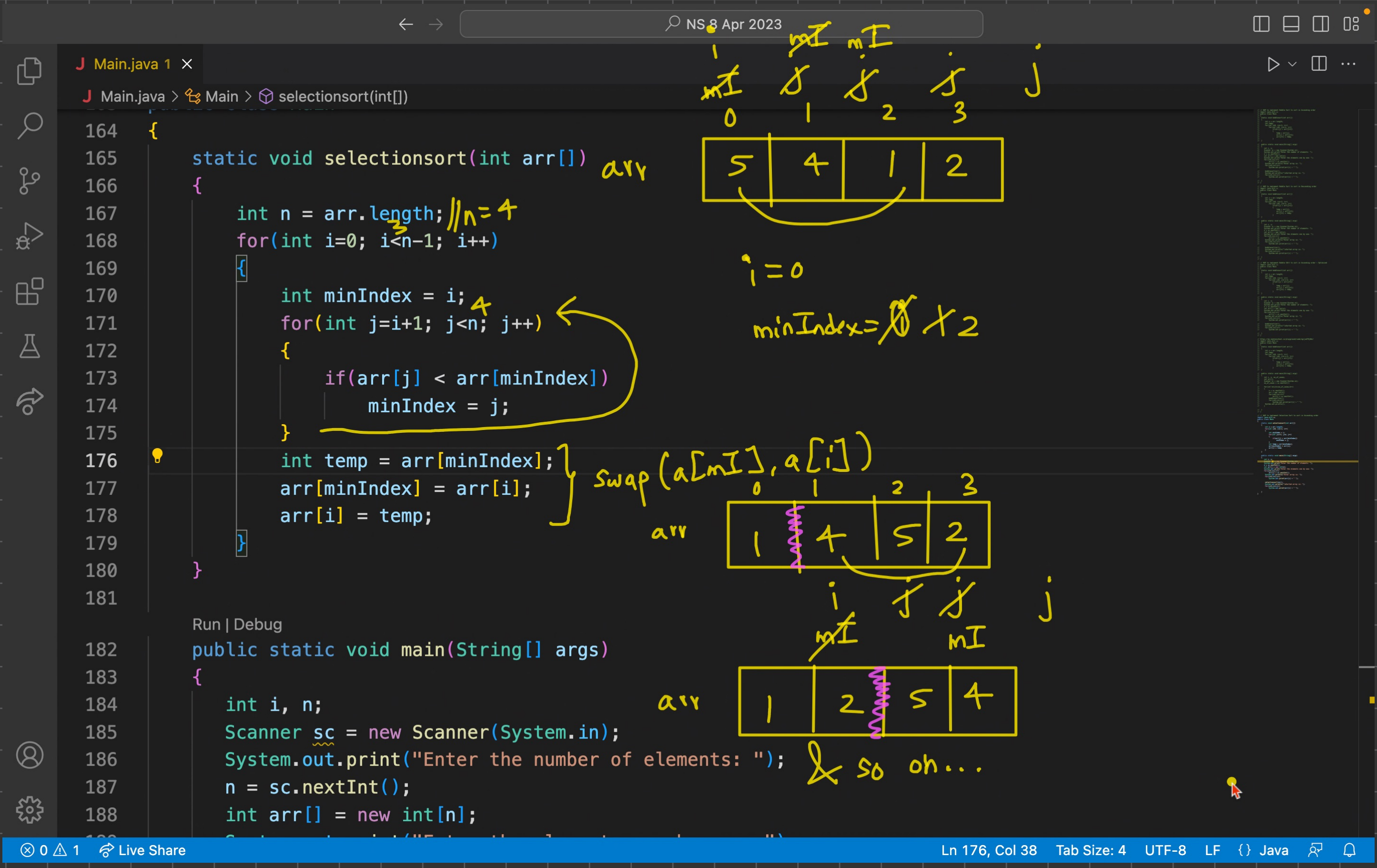
```
//    }
// }

// WAP to implement Bubble SOrt to sort in Ascending order - Optimized
import java.util.*;
public class Main
{
    static void bubblesort(int arr[])
    {
        int n = arr.length;
        int temp;
        for(int i=0; i<n-1; i++)
            for(int j=0; j<n-i-1; j++)
                if(arr[j] > arr[j+1])
                {
                    temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }

    Run | Debug
    public static void main(String[] args)
    {
       int i, n;
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter the number of elements: ");
```

Selection Sort

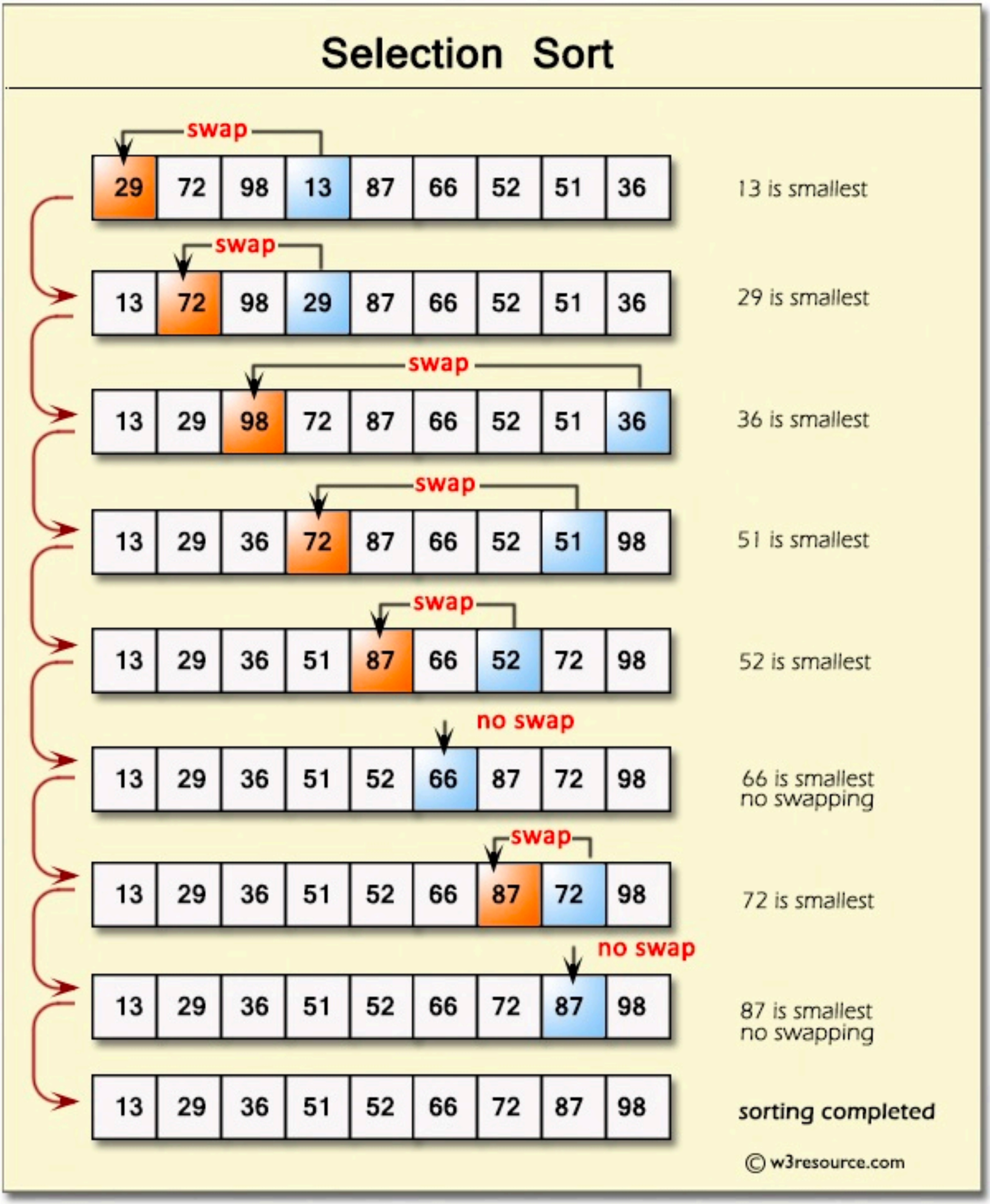


0	1	2	3	4	5	6	7
22	35	17	8	13	44	5	28
5	35	17	8	13	44	22	28
5	8	17	35	13	44	22	28
5	8	13	35	17	44	22	28
5	8	13	17	35	44	22	28
5	8	13	17	22	44	35	28
5	8	13	17	22	28	35	44
5	8	13	17	22	28	35	44

∴ SORTED.



```java
{
    static void selectionsort(int arr[])
    {
        int n = arr.length;
        for(int i=0; i<n-1; i++)
        {
            int minIndex = i;
            for(int j=i+1; j<n; j++)
            {
                if(arr[j] < arr[minIndex])
                    minIndex = j;
            }
            int temp = arr[minIndex];
            arr[minIndex] = arr[i];
            arr[i] = temp;
        }
    }

    public static void main(String[] args)
    {
       int i, n;
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter the number of elements: ");
       n = sc.nextInt();
       int arr[] = new int[n];
```



Selection Sort

29	72	98	13	87	66	52	51	36	13 is smallest
13	72	98	29	87	66	52	51	36	29 is smallest
13	29	98	72	87	66	52	51	36	36 is smallest
13	29	36	72	87	66	52	51	98	51 is smallest
13	29	36	51	87	66	52	72	98	52 is smallest
13	29	36	51	52	66	87	72	98	66 is smallest no swapping
13	29	36	51	52	66	87	72	98	72 is smallest
13	29	36	51	52	66	72	87	98	87 is smallest no swapping
13	29	36	51	52	66	72	87	98	sorting completed

© w3resource.com

Selection sort is a simple sorting algorithm. This sorting 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 part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of O(n2), where n is the number of items.