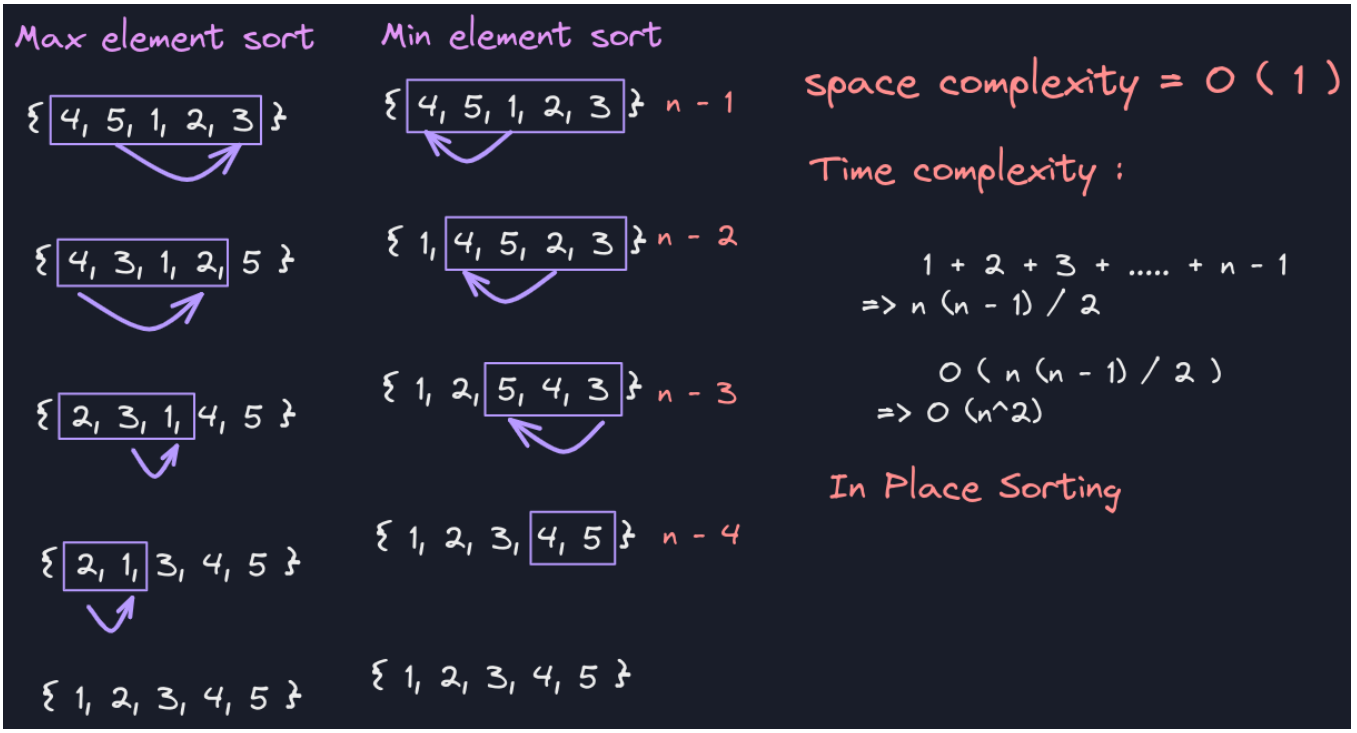
Selection Sort

- In-place Sorting:
  - A Sorting Algorithm which uses constant Space Complexity i.e. zero Auxiliary Space
- Stable Sorting:
  - A Sorting Algorithm without changing indexes of equal elements while sorting.

In Selection Sort, Largest or Smallest element is selected and is placed in correct place.



- Time Complexity :
  - Best Case : O(n^2)
  - Worst Case : O(n^2)
- Space Complexity : O(1) [In-place Sorting]
- Non-Stable Sorting Algorithm
- Use Case : Performs well in small arrays

```java
package com.inclass;

import java.util.Arrays;

public class SelectionSortAlgorithm {
    public static void main(String[] args) {
        int[] arr = {4, 5, 1, 2, 3};
//        maxSelectionSort(arr);
        minSelectionSort(arr);
    }

    static void maxSelectionSort(int[] arr) {
        int count = arr.length - 1;
        while (count > 0) {
            int maxIndex = 0;
            for (int i = 1; i ≤ count; i++) {
                if (arr[i] > arr[maxIndex]) {
```

```
                    maxIndex = i;
                }
            }
            if (count != maxIndex) {
                int temp = arr[count];
                arr[count] = arr[maxIndex];
                arr[maxIndex] = temp;
            }
            System.out.println(Arrays.toString(arr));
            count--;
        }
    }

    static void minSelectionSort(int[] arr) {
        int count = 0;
        while (count < arr.length - 1) {
            int minIndex = count;
            for (int i = count + 1; i < arr.length; i++) {
                if (arr[i] < arr[minIndex]) {
                    minIndex = i;
                }
            }
            if (count != minIndex) {
                int temp = arr[count];
                arr[count] = arr[minIndex];
                arr[minIndex] = temp;
            }
            System.out.println(Arrays.toString(arr));
            count++;
        }
    }
}
```