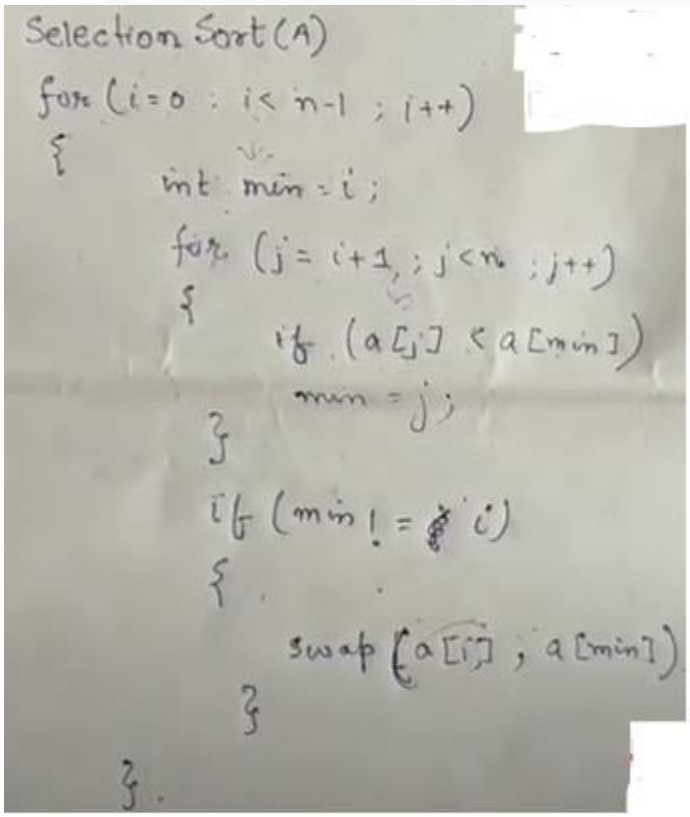
Selection sort *is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.*

The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

Selection Sort Complexity

Time Complexity	
Best	$O(n^2)$
Worst	$O(n^2)$
Average	$O(n^2)$
Space Complexity	O(1)
Stability	No



```
Selection Sort (A)
for (i=0 ; i< n-1 ; i++)
{
    int min = i;
    for (j= i+1; j<n ; j++)
    {
        if (a[j] < a[min])
        min = j;
    }
    if (min != i)
    {
        swap (a[i] , a[min])
    }
}
```

ABDULLAH: 03343127215

Lets consider the following array as an example: ***arr[] = {64, 25, 12, 22, 11}***

First pass:

- *For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where* ***64*** *is stored presently, after traversing whole array it is clear that* ***11*** *is the lowest value.*

64	*25*	*12*	*22*	*11*

- *Thus, replace 64 with 11. After one iteration* ***11****, which happens to be the least value in the array, tends to appear in the first position of the sorted list.*

11	*25*	*12*	*22*	*64*

Second Pass:

- *For the second position, where 25 is present, again traverse the rest of the array in a sequential manner.*

11	***25***	*12*	*22*	*64*

- *After traversing, we found that* ***12*** *is the second lowest value in the array and it should appear at the second place in the array, thus swap these values.*

11	***12***	*25*	*22*	*64*

Third Pass:

- *Now, for third place, where* ***25*** *is present again traverse the rest of the array and find the third least value present in the array.*

11	*12*	***25***	*22*	*64*

- *While traversing, **22** came out to be the third least value and it should appear at the third place in the array, thus swap **22** with element present at third position.*

11	*12*	***22***	*25*	*64*

Fourth pass:

- *Similarly, for fourth position traverse the rest of the array and find the fourth least element in the array*
- *As **25** is the 4th lowest value hence, it will place at the fourth position.*

11	*12*	*22*	***25***	*64*

Fifth Pass:

- *At last the largest value present in the array automatically get placed at the last position in the array*
- *The resulted array is the sorted array.*

11	***12***	***22***	***25***	***64***

Advantages - Selection Sort

- Selection sort algorithm is 60% more efficient than bubble sort algorithm.
- Selection sort algorithm is easy to implement.
- Selection sort algorithm can be used for small data sets, unfortunately Insertion sort algorithm best suitable for it.

Disadvantages - Selection Sort

- Running time of Selection sort algorithm is very poor of 0 (n^2).
- Insertion sort algorithm is far better than selection sort algorithm.