Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.
AOA / Even Sem 2021-22 / Experiment 1.

YASH SARANG.
47 / D6AD.
EXPERIMENT - 1.

Aim: Selection Sort and Insertion Sort

Theory:
SELECTION SORT
The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.
1) The subarray is already sorted.
2) Remaining subarray which is unsorted.
In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

Time Complexity: O(n2) as there are two nested loops.
Auxiliary Space: O(1)

The good thing about selection sort is it never makes more than O(n) swaps and can be useful when memory writing is a costly operation.

INSERTION SORT

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Algorithm

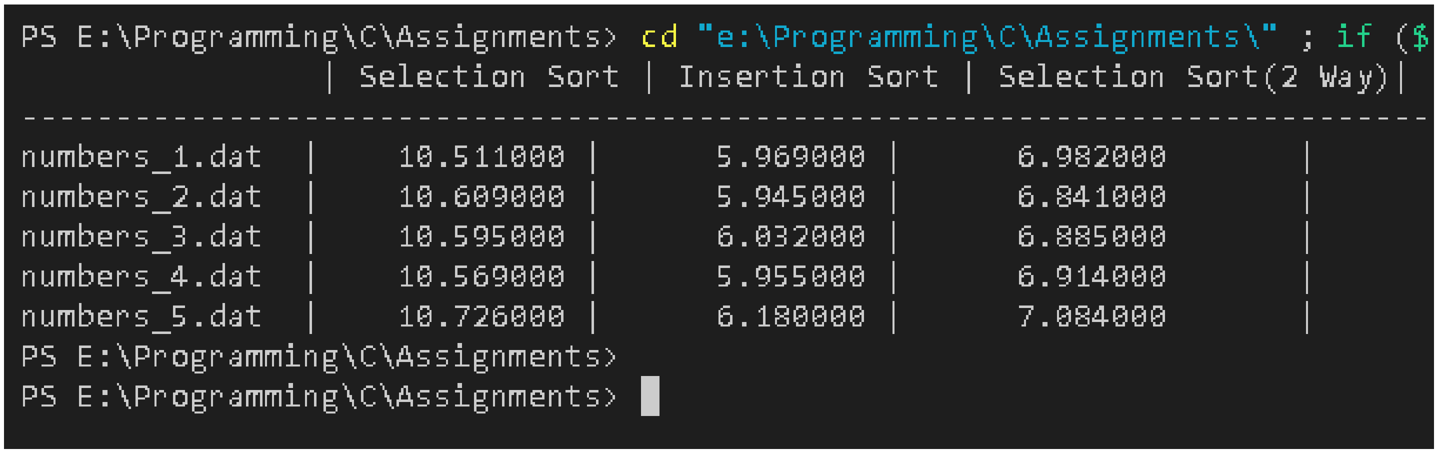
To sort an array of size n in ascending order:
1: Iterate from arr[1] to arr[n] over the array.
2: Compare the current element (key) to its predecessor.
3: If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

Time Complexity: O(n^2)
Auxiliary Space: O(1)
Boundary Cases: Insertion sort takes maximum time to sort if elements are sorted in reverse order. And it takes minimum time (Order of n) when elements are already sorted.
Algorithmic Paradigm: Incremental Approach
Sorting In Place: Yes
Stable: Yes
Online: Yes
Uses: Insertion sort is used when the number of elements is small. It can also be useful when the input array is almost sorted, only a few elements are misplaced in a complete big array.

CODE:

Code in the Final.c file attached along with this doc.

OUTPUT:



```
PS E:\Programming\C\Assignments> cd "e:\Programming\C\Assignments\" ; if ($
                 | Selection Sort | Insertion Sort | Selection Sort(2 Way)|
-------------------------------------------------------------------------------
numbers_1.dat  |     10.511000 |       5.969000 |       6.982000       |
numbers_2.dat  |     10.609000 |       5.945000 |       6.841000       |
numbers_3.dat  |     10.595000 |       6.032000 |       6.885000       |
numbers_4.dat  |     10.569000 |       5.955000 |       6.914000       |
numbers_5.dat  |     10.726000 |       6.180000 |       7.084000       |
PS E:\Programming\C\Assignments>
PS E:\Programming\C\Assignments> 
```

CONCLUSION:

By performing this experiment, I can conclude that Insertion sort is faster in most of the cases due to it's ability of modifying the array while sorting in such a way that the subarray is always kept sorted. In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.
Which eventually help sus to save iterations.
Contrarily In Selection sort, we have to mandatorily go through all the n iterations.
