5 (1) (24)

PAGE:
DATE: / /

* Assignment No 5 *

Aim:-

Write a python program to store second year percentage of student in array write function for sorting array of floating point number in ascending order using.

(a) Insertion sort

(b) Shell sort & display top five scores.

Objectives:-

(1) to understand the concept of sorting method

(2) to find the performance of sorting method.

(3) to apply sorting method to sort the given array list in ascending order and display top 5 scores

outcomes:-

(1) to design & implement sorting techniques.

(2) to calculate time and space complexity of the given sorting methods.

(3) to apply the functions of sorting method on given data for sorting the element in ascending order and display top five elements.

os programming tools used:-

64-bit fedora-17 or latest open-source os or last microsoft windows, Pycham IDE.

Theory:-

In this assignment we will implement the given sorting techniques.

PAGE:
DATE: / /

for this purpose, we need to accept the students data of percentage in array are list, the percentage should be accepted as floating point number thereafter, apply the following sorting operation to sort the percentage in ascending order.

The program should display the sorted element in ascending order as well as display the top five scores amongst them.

Sorting is the process of arranging the the given data in some pre-defined order or sequence, the order can be either ascending or descending.

Insertion Sort:-

- Insertion sort works similar to the sorting of playing cards in hands.
- It is assumed that the first card is already sorted in the card game, & then we select an unsorted card.
- If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise it will be placed at the left side. similary all unsorted cards are taken & put in their exact place
- The same approach is applied in insertion sort. the idea behind the insertion sort is that first take one element, iterate it through the sorted array.

PAGE:
DATE: / /

- Although it is simple to use, it is not appropriate for large data set as the time complexity of insertion sort in the average case & worst case in $O(n^2)$ where n is the number of items.

- Insertion sort is less efficient that the other sorting algorithm like heap sort, quick, sort, merge sort, etc.

Pseudo code for - Insertion Sort:

Algorithm insertion sort (A[I]) {
for i=1 to length (A) inclutive do:
value to insert = A[i].
hole position = A[i].

while hole position > 0 & A (hole position-1]>
value to insert do.
A [hole position] = A [hole position-1]
hole position = hole position-1

end while.
A [hole position] = value to insert.
end for.
end procedure.

time complexity:
worst Case - $O(n^2)$
Average Case - $O(n^2)$.

5 (4)

PAGE:
DATE: / /

Shell sort:-

- shell sort is a highly efficient sorting algorithm & is based on insertion sort algorithm.
- This algorithm avoids large shifts as in case of insertion sort, if the smaller values is to the far right and has to be moved to the far left.
- This algorithm uses insertion sort on a widely spread element first to sort them & the sorts the less widely spaced elements. this spacing is termed as interval.
- This algorithm is quite efficient for medium sized data sets as it average & worst-case Complexity of this algorithm depends upon the gap sequence the best known is O(n), where n is the number of items & the worst case Space Complexity is O(n)

Pseudo code for shell sort:

```
Algorithm ShellSort (A[])
    while interval < A.length/3 {
        interval = interval * 3+1;
    }
    while interval > 0 {
        for outer = interval to A.length {
            value to insert = A[outer]
            inner = outer;
        while inner > interval - 1 && A[inner-interval] >=
        value to insert {
            A[inner] = A[inner - interval]
            inner = inner - interval
    }.
```

PAGE:
DATE: / /

```
A[inner] = value to insert
    }
interval = (interval - 1) / 3;
    }
end procedure.
```

time complexity:

worst case = Average case = O(n)

Conclusion :-

The function for different sorting methods are implemented successfully on student percentage data with efficient time & space complexity & displayed top-five scores amongst them.