Assignment 4

Name:- Harsh Balaprasad Baheti
Div :- A
Roll No.:-SECOA112

Assignment 4

Objective :

To illustrate the use of selection sort and bubble sort.

Outcome :

Students will be able to use various sorting operations using one dimensional array for sorting.

Theory:

1) selection sort :

selection sort is a sorting algorithm, specifically an in place comparision sort, It has O(*n* 2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.
The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

2)bubble sort:

Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and impractical for most problems even when compared to insertion sort . It can be practical if the input is usually in sorted order but may occasionally have some out-of-order elements nearly in position.

Problem Statement :

Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using a) Selection Sort b) Bubble sort and display top five scores.

Algorithm :

1. bubble_sort:-

```
Step 1)    start
Step 2)    input list from user.
Step 3)    for every i range length of (list-1)
    jump to step4 else step7.
Step 4)    for every j range length of (list-i-1)
    jump to step5 else step3.
Step 5)    if list[j]>list[j+1] jump to step 6 else step4.
```

Step 6) swap(list[j],list[j+1]) jump to step 4.

Step 7) exit()

2. selection sort:-

Step 1) start

Step 2) input list from user.

Step 3) for every i range length of list

jump to step4 else step9.

Step 4) set minimum index to i

Step 5) for every j range length of (i+1,lengt)

jump to step6 else step8.

Step 6) if list[j]<list[minimum index]jump to step 7 else

step5.

Step 7) set minimum index to j jump to step 5.

Step 8) swap(list[i],list[minimum index]) jump to step

3.

Step 9) exit()

3. top5:-

```
Step 1)    start
Step 2)    print "Top 5 scores are :"
Step 3)    for i in range (1,6) jump to 4 else step 5.
Step 4)    print(iterable[0 - i], end=" ")  jump to 3.
Step 5)    exit()
```

Program/Code:

```python
list1 = [10, 20, 30, 40, 51, 55, 20, 32, 521, 45, 84, 54, 65, 625]
list2 = [10, 20, 30, 40, 51, 55, 20, 32, 521, 45, 84, 54, 65, 740]


# bubble sort
def bubble_sort(lister):
    for i in range(len(lister) - 1):
        for j in range(len(lister) - i - 1):
            if lister[j] > lister[j + 1]:
                lister[j], lister[j + 1] = lister[j + 1], lister[j]


# selection sort
def selection_sort(array):
    for i in range(len(array)):
        min_index = i
        for j in range(i + 1, len(array)):
            if array[min_index] > array[j]:
                min_index = j
        array[i], array[min_index] = array[min_index], array[i]


# to print top 5 scores from list
```

```
def top5(iterable):
    print("Top 5 scores are :")
    for i in range(1, 6):
        print(iterable[0 - i], end=" ")
    print()


bubble_sort(list1)
selection_sort(list2)
top5(list1)
top5(list2)
```

Output :

```
Top 5 scores are :
625 521 84 65 55
Top 5 scores are :
740 521 84 65 55
```

Time Complexity :

sr no.	Function	Time Complexity
1	**bubble_sort**	**O(n^2)**
2	**selection_sort**	**O(n^2)**
3	**top5**	**O(1)**

Total Time Complexity:= O(n^2)

Conclusion :-

Illustrated the use of selection sort and bubble sort.