Due date: 7th October,2023

Assignment 1

Step 1:

Implement 5 functions in Python that would sort an unsorted list, i.e

defbubble_sort(my_list):

defselection_sort(my_list):

definsertion_sort(my_list):

defquick_sort(mylist):

definsertion_sort(mylist, left, right):

Provide a main that uses each of the above functions to sort a list of length 100 and make sure it works as expected

Step 2:

Add lines of code to the above functions so that apart from sorting the received list, it calculates T(n) and returns it. This means that each of the above functions will have a return value which is the exact number of operations executed to perform the sort.

Use a main to test your T(n) calculation. A good way of testing your T(n) is this. Send a best case scenario, a worst case scenario and an average scenario to your sort function and see what numbers come out for your T(n). Explain what best, worst and average case scenarios will be for a sorting algorithm.

Step 3:

Use the sort functions with T(n) calculation feature to plot T(n) vs. n for a wide range of list sizes. Say 10, 50 , 100, 500, 1000, 5000, 10000, 50000, 100000, 1000000, 10000000. Make sure you use a WORST CASE scenario for your list so that your T(n) is a good reflection of O(n). Do you see your curves aligned with what we learnt about the performance of the sort algorithms

Step 4

Using time library in python, time your sort algorithms for the same list sizes that you plotted in step 3 and this time plot algorithm completion time vs n. Interpret the results while compared to the step 3 plot and compared to your knowledge of the algorithm's performances. Don't forget to use WORST CASE scenario.

Example for Step 2 in bubble sort:

```
defbubble_sort(my_list):
    steps = 0
    for i in range(0, len(my_list) - 1):
        for j in range(0, len(my_list) - 1 - i):
            if my_list[j] >my_list[j + 1]:
                steps += 4 # 4 operations, 3 for the swap and one for the comparison
                my_list[j], my_list[j + 1] = my_list[j + 1], my_list[j]
    return steps
```

Code to create a random list of a specific size for steps 3 and 4 :

```
import random
listSize = 50
rand_list = [random.randint(0,101) for val in range(listSize)]
rand_list.sort(reverse=True)
print("steps needed for sorting: {}".format(bubble_sort(rand_list)))
```