**SSN College of Engineering**

Department of Information Technology

UIT2201 — Programming and Data Structures

2022 – 2023

**Exercise — 05**

U. Pranaav | IT-B | 3122225002093

I. AIM:

To arrange n elements in ascending or order using Bubble sort, Insertion sort and Selection sort and analyze the time complexity of the code and express the same in asymptotic notation. Also find the performance of each sorting algorithm in their average case, best case and worst case scenarios.

II. CODE:

# -\*- coding: utf-8 -\*-

"""

This module provides a series of functions that calculate the

value of a polynomial, given the polynomial coefficient terms

and the value of x in 3 different time complexities, namely

O(n^2), O(nlogn) and O(n). This is a part of the exercises

given under the course UIT2201 (Programming and Data Structures).

In this source code I have executed my own logic. The code

follows good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 03 2023

Revised on Wed May 03 2023

Original Author: U. Pranaav <pranaav2210205@ssn.edu.in>

"""

import random

import time

from math import log2

*def* bubble\_sort(*data*):

    '''

    The given function sorts a given list using

    the bubble sort method and has a time complexity

    of O(n^2) and returns number of comparisons, swappings

    and amount of time taken to run the function inside

    a tuple.

    The given input is modified.

    args:

        data: the list to be sorted.

    Returns:

        A tuple containing sorted list number of comparisons,

        swappings and amount of time taken to run the function

        inside a tuple.

    '''

    comparisons = 0

    swappings = 0

    n = len(*data*)

    start\_time = time.time()

    for i in range(n):

        comparisons += 1

        for j in range(n-i-1):

            comparisons += 1

            if *data*[j] > *data*[j+1]:

                swappings += 1

*data*[j], *data*[j+1] = *data*[j+1], *data*[j]

    end\_time = time.time()

    time\_taken = end\_time - start\_time

    return (*data*, comparisons, swappings, time\_taken)

*def* selection\_sort(*data*):

    '''

    The given function sorts a given list using

    the selection sort method and has a time complexity

    of O(n^2) and returns number of comparisons, swappings

    and amount of time taken to run the function inside

    a tuple.

    The given input is modified.

    args:

        data: the list to be sorted.

    Returns:

        A tuple containing sorted list number of comparisons,

        swappings and amount of time taken to run the function

        inside a tuple.

    '''

    comparisons = 0

    swappings = 0

    n = len(*data*)

    start\_time = time.time()

    for i in range(len(*data*)):

        comparisons += 1

        temp\_ind = i

        for j in range(i+1, len(*data*)):

            comparisons += 1

            if *data*[temp\_ind] > *data*[j]:

                temp\_ind = j

        if i != temp\_ind:

            swappings += 1

*data*[i], *data*[temp\_ind] = *data*[temp\_ind], *data*[i]

    end\_time = time.time()

    time\_taken = end\_time - start\_time

    return (*data*, comparisons, swappings, time\_taken)

*def* insertion\_sort(*data*):

    '''

    The given function sorts a given list using

    the insertion sort method and has a time complexity

    of O(n^2) and returns number of comparisons, swappings

    and amount of time taken to run the function inside

    a tuple.

    The given input is modified.

    args:

        data: the list to be sorted.

    Returns:

        A tuple containing sorted list number of comparisons,

        swappings and amount of time taken to run the function

        inside a tuple.

    '''

    comparisons = 0

    swappings = 0

    n = len(*data*)

    start\_time = time.time()

    for i in range(1,n-1):

        j = i - 1

        temp\_var = *data*[i]

        while j >= 0 and temp\_var < *data*[j]:

                comparisons += 1

                swappings += 1

*data*[j+1] = *data*[j]

                j-=1

*data*[j+1] = temp\_var

    end\_time = time.time()

    time\_taken = end\_time - start\_time

    return (*data*, comparisons, swappings, time\_taken)

*def* random\_list(*size*):

    '''

    The given function generates a random number of values

    and returns the values in a list.

    args:

        size: the number of point objects to be

        generated

    Returns:

        A list of random integer values.

    '''

    random\_list = []

    for case in range(*size*):

        x\_val = random.randint(-1000,1000)

        random\_list.append(x\_val)

    return random\_list

*def* printout\_cases(*random\_lst*, *sorted\_lst*, *comparisons*, *swappings*, *time*):

    '''

    This function provides a concise way of showing the

    original data, sorted data as well as number of

    comparisons, swappings and time taken.

    The input is not modified and there are no side effects.

    args:

        random\_lst: the unsorted data

        sorted\_lst: the sorted data

        comparisons: number of comparisons between the elements

        of the data

        swappings: number of swappings done

        time: time taken to run the function

    Returns:

        None

    '''

    #print(f"Random list is : {random\_lst}")

    print()

    #print(f"Sorted list is : {sorted\_lst}")

    print("Size of data is : ",len(*random\_lst*))

    print(*f*"Number of comparisons is : {*comparisons*}")

    print(*f*"Number of swappings is : {*swappings*}")

    print("Time taken is : ", *time*)

    print()

    print('-'\*100)

    print()

*def* avg\_scenario(*size*):

    '''

    The function provides the parameters required for

    printing an average case scenario for sorting.

    The input is not modified and there are no side effects.

    args:

        size: the size of random list to be generated

    Returns:

        None

    '''

    random\_lst = random\_list(*size*)

    copy\_1 = random\_lst.copy()

    copy\_2 = random\_lst.copy()

    copy\_3 = random\_lst.copy()

    sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel = selection\_sort(copy\_1)

    print("SELECTION SORT")

    printout\_cases(copy\_1,sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel)

    sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub = bubble\_sort(copy\_2)

    print("BUBBLE SORT")

    printout\_cases(copy\_2,sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub)

    sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins = insertion\_sort(copy\_3)

    print("INSERTION SORT")

    printout\_cases(copy\_3,sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins)

*def* best\_scenario(*size*):

    '''

    The function provides the parameters required for

    printing a best case scenario for sorting.

    The input is not modified and there are no side effects.

    args:

        size: the size of random list to be generated

    Returns:

        None

    '''

    random\_lst = [x for x in range(*size*)]

    copy\_1 = random\_lst.copy()

    copy\_2 = random\_lst.copy()

    copy\_3 = random\_lst.copy()

    sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel = selection\_sort(copy\_1)

    print("SELECTION SORT")

    printout\_cases(copy\_1,sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel)

    sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub = bubble\_sort(copy\_2)

    print("BUBBLE SORT")

    printout\_cases(copy\_2,sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub)

    sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins = insertion\_sort(copy\_3)

    print("INSERTION SORT")

    printout\_cases(copy\_3,sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins)

*def* worst\_scenario(*size*):

    '''

    The function provides the parameters required for

    printing a worst case scenario for sorting.

    The input is not modified and there are no side effects.

    args:

        size: the size of random list to be generated

    Returns:

        None

    '''

    random\_lst = [x for x in range(*size*,0,-1)]

    copy\_1 = random\_lst.copy()

    copy\_2 = random\_lst.copy()

    copy\_3 = random\_lst.copy()

    sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel = selection\_sort(copy\_1)

    print("SELECTION SORT")

    printout\_cases(copy\_1,sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel)

    sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub = bubble\_sort(copy\_2)

    print("BUBBLE SORT")

    printout\_cases(copy\_2,sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub)

    sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins = insertion\_sort(copy\_3)

    print("INSERTION SORT")

    printout\_cases(copy\_3,sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins)

*def* main\_calc(*user\_input*, *second\_user\_input*):

    '''

    This function can be used to calculate the value

    of nlogn for finding time complexities.

    args:

        user\_input: the value of number of comparisons

        by a given function.

        second\_user\_input: the value of n

    Returns:

        the value of nlogn.

    '''

    result = *user\_input* / (*second\_user\_input* \* log2(*second\_user\_input*))

    return result

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

    #this part of the code will only be run when the function is called directly

    #it will not be executed when it is imported as a module

    for i in range(2,11):

        lst = random\_list(i)

        copy\_1 = lst.copy()

        copy\_2 = lst.copy()

        copy\_3 = lst.copy()

        sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub = bubble\_sort(copy\_1)

        sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins = insertion\_sort(copy\_2)

        sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel = selection\_sort(copy\_3)

        print("BUBBLE SORT")

        printout\_cases(copy\_1,sorted\_lst\_bub, comparisons\_bub, swappings\_bub, time\_bub)

        print("INSERTION SORT")

        printout\_cases(copy\_2,sorted\_lst\_ins, comparisons\_ins, swappings\_ins, time\_ins)

        print("SELECTION SORT")

        printout\_cases(copy\_3,sorted\_lst\_sel, comparisons\_sel, swappings\_sel, time\_sel)

    print("Now testing their average case for large input size:")

    avg\_scenario(10000)

    print("Now testing their best case for large input size:")

    best\_scenario(10000)

    print("Now testing their worst case for large input size:")

    worst\_scenario(10000)

III. OUTPUT:

BUBBLE SORT

Random list is : [-783, 918]

Sorted list is : [-783, 918]

Size of data is : 2

Number of comparisons is : 3

Number of swappings is : 0

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-783, 918]

Sorted list is : [-783, 918]

Size of data is : 2

Number of comparisons is : 0

Number of swappings is : 0

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-783, 918]

Sorted list is : [-783, 918]

Size of data is : 2

Number of comparisons is : 3

Number of swappings is : 0

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-860, 238, 510]

Sorted list is : [-860, 238, 510]

Size of data is : 3

Number of comparisons is : 6

Number of swappings is : 1

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-860, 510, 238]

Sorted list is : [-860, 510, 238]

Size of data is : 3

Number of comparisons is : 0

Number of swappings is : 0

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-860, 238, 510]

Sorted list is : [-860, 238, 510]

Size of data is : 3

Number of comparisons is : 6

Number of swappings is : 1

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-206, 158, 716, 824]

Sorted list is : [-206, 158, 716, 824]

Size of data is : 4

Number of comparisons is : 10

Number of swappings is : 2

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-206, 158, 824, 716]

Sorted list is : [-206, 158, 824, 716]

Size of data is : 4

Number of comparisons is : 1

Number of swappings is : 1

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-206, 158, 716, 824]

Sorted list is : [-206, 158, 716, 824]

Size of data is : 4

Number of comparisons is : 10

Number of swappings is : 2

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-916, -736, -295, -76, 912]

Sorted list is : [-916, -736, -295, -76, 912]

Size of data is : 5

Number of comparisons is : 15

Number of swappings is : 4

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-916, -295, -76, 912, -736]

Sorted list is : [-916, -295, -76, 912, -736]

Size of data is : 5

Number of comparisons is : 1

Number of swappings is : 1

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-916, -736, -295, -76, 912]

Sorted list is : [-916, -736, -295, -76, 912]

Size of data is : 5

Number of comparisons is : 15

Number of swappings is : 4

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-761, -577, -433, 203, 510, 618]

Sorted list is : [-761, -577, -433, 203, 510, 618]

Size of data is : 6

Number of comparisons is : 21

Number of swappings is : 9

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-761, -577, -433, 203, 618, 510]

Sorted list is : [-761, -577, -433, 203, 618, 510]

Size of data is : 6

Number of comparisons is : 8

Number of swappings is : 8

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-761, -577, -433, 203, 510, 618]

Sorted list is : [-761, -577, -433, 203, 510, 618]

Size of data is : 6

Number of comparisons is : 21

Number of swappings is : 5

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-852, -606, -575, -486, -423, 342, 744]

Sorted list is : [-852, -606, -575, -486, -423, 342, 744]

Size of data is : 7

Number of comparisons is : 28

Number of swappings is : 12

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-852, -606, -575, -486, -423, 744, 342]

Sorted list is : [-852, -606, -575, -486, -423, 744, 342]

Size of data is : 7

Number of comparisons is : 11

Number of swappings is : 11

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-852, -606, -575, -486, -423, 342, 744]

Sorted list is : [-852, -606, -575, -486, -423, 342, 744]

Size of data is : 7

Number of comparisons is : 28

Number of swappings is : 4

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-953, -609, -62, -26, 303, 768, 854, 857]

Sorted list is : [-953, -609, -62, -26, 303, 768, 854, 857]

Size of data is : 8

Number of comparisons is : 36

Number of swappings is : 19

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-953, -609, -62, -26, 768, 854, 857, 303]

Sorted list is : [-953, -609, -62, -26, 768, 854, 857, 303]

Size of data is : 8

Number of comparisons is : 16

Number of swappings is : 16

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-953, -609, -62, -26, 303, 768, 854, 857]

Sorted list is : [-953, -609, -62, -26, 303, 768, 854, 857]

Size of data is : 8

Number of comparisons is : 36

Number of swappings is : 5

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-911, 16, 227, 300, 381, 455, 491, 728, 955]

Sorted list is : [-911, 16, 227, 300, 381, 455, 491, 728, 955]

Size of data is : 9

Number of comparisons is : 45

Number of swappings is : 12

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-911, 16, 227, 300, 381, 455, 728, 955, 491]

Sorted list is : [-911, 16, 227, 300, 381, 455, 728, 955, 491]

Size of data is : 9

Number of comparisons is : 10

Number of swappings is : 10

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-911, 16, 227, 300, 381, 455, 491, 728, 955]

Sorted list is : [-911, 16, 227, 300, 381, 455, 491, 728, 955]

Size of data is : 9

Number of comparisons is : 45

Number of swappings is : 4

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Random list is : [-871, -455, -354, -306, -289, -51, 343, 355, 495, 613]

Sorted list is : [-871, -455, -354, -306, -289, -51, 343, 355, 495, 613]

Size of data is : 10

Number of comparisons is : 55

Number of swappings is : 25

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

INSERTION SORT

Random list is : [-871, -455, -354, -306, -289, 343, 355, 495, 613, -51]

Sorted list is : [-871, -455, -354, -306, -289, 343, 355, 495, 613, -51]

Size of data is : 10

Number of comparisons is : 21

Number of swappings is : 21

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

SELECTION SORT

Random list is : [-871, -455, -354, -306, -289, -51, 343, 355, 495, 613]

Sorted list is : [-871, -455, -354, -306, -289, -51, 343, 355, 495, 613]

Size of data is : 10

Number of comparisons is : 55

Number of swappings is : 9

Time taken is : 0.0

----------------------------------------------------------------------------------------------------

Now testing their average case for large input size:

SELECTION SORT

Size of data is : 10000

Number of comparisons is : 50005000

Number of swappings is : 9990

Time taken is : 8.247478485107422

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Size of data is : 10000

Number of comparisons is : 50005000

Number of swappings is : 25382922

Time taken is : 21.93934679031372

----------------------------------------------------------------------------------------------------

INSERTION SORT

Size of data is : 10000

Number of comparisons is : 25381524

Number of swappings is : 25381524

Time taken is : 8.902828216552734

----------------------------------------------------------------------------------------------------

Now testing their best case for large input size:

SELECTION SORT

Size of data is : 10000

Number of comparisons is : 50005000

Number of swappings is : 0

Time taken is : 5.52709698677063

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Size of data is : 10000

Number of comparisons is : 50005000

Number of swappings is : 0

Time taken is : 7.8830037117004395

----------------------------------------------------------------------------------------------------

INSERTION SORT

Size of data is : 10000

Number of comparisons is : 0

Number of swappings is : 0

Time taken is : 0.003996133804321289

----------------------------------------------------------------------------------------------------

Now testing their worst case for large input size:

SELECTION SORT

Size of data is : 10000

Number of comparisons is : 50005000

Number of swappings is : 5000

Time taken is : 9.55515432357788

----------------------------------------------------------------------------------------------------

BUBBLE SORT

Size of data is : 10000

Number of comparisons is : 50005000

Number of swappings is : 49995000

Time taken is : 23.812073469161987

----------------------------------------------------------------------------------------------------

INSERTION SORT

Size of data is : 10000

Number of comparisons is : 49985001

Number of swappings is : 49985001

Time taken is : 17.835464477539062

----------------------------------------------------------------------------------------------------