Ex 5 - Sorting Algorithm

T SADAKOPA RAMAKRISHNAN | 3122225002109 | IT - B

PART A:

Q1) Arrange n elements either in ascending or descending order using Bubble sort and Selection sort. Write a Python function to sort n numbers and analyze the time complexity of your code and express the same in asymptotic notation. Give your inference on the performance of these sorting algorithms in terms of the number of comparisons and number of swaps performed. Try the best case, worst case and average case scenarios and report your observations.

CODE:

```
# -*- coding: utf-8 -*-
'''This module provides the sorting algorithms,
namely Bubble Sort and Selection Sort. The empirical
analysis is done for both the algorithms and
O(n^2) complexity is verified.This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed May 3 2023

Revised on Wed May 3 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
'''
```

```
import random
from timeit import default_timer as timer

def createLIST():
    '''
    This function creates a sequence of integer using random
module
    '''
    n = int(input("Enter the number of elements: "))
    seq = []
    for i in range(n):
        seq.append(random.randint(-5,1500))
    print("Original List:", seq)
    return seq

def BubbleSort(seq):
    '''
    This function sorts the entered sequence using
BubbleSort which has time complexity of O(n^2)
    '''
    fn = 0
    swap = 0
    n = len(seq)
    start = timer()
    for i in range(0,n-1):
        fn += 1
        for j in range(0,n-i-1):
            fn += 1
            if seq[j] > seq[j+1]:
                swap += 1
                seq[j] , seq[j+1] = seq[j+1], seq[j]
    end = timer()
    print("f(n) =", fn)
    print("No of swapings:", swap)
    print("Exact time:",end - start )
    return seq
```

```
def SelectionSort(seq):
    '''
    This function sorts the entered sequence using Selection
Sort which has time complexity of O(n^2)
    '''
    fn = 0
    swap = 0
    n = len(seq)
    start = timer()
    for i in range(n):
        fn += 1
        minIndex = i
        for j in range(i+1, n):
            fn += 1
            if seq[minIndex] > seq[j]:
                swap += 1
                minIndex = j

        seq[i], seq[minIndex] = seq[minIndex], seq[i]
    end = timer()
    print("f(n) =", fn)
    print("No of swapings:", swap)
    print("Exact time:",end - start )
    return seq

#Test cases for this source code:

if __name__ == "__main__":

    #Bubble Sort
    print("Using Bubble Sort: ")
    print(BubbleSort(createLIST()))
    print()

    #Selection Sort
    print("Using Selection Sort: ")
    print(SelectionSort(createLIST()))
```

```
    print()
```

OUTPUT:

```
Using Bubble Sort:
Enter the number of elements: 5
Original List: [874, 1192, 707, 1129, 473]
f(n) = 14
No of swapings: 7
Exact time: 1.0890000339713879e-05
[473, 707, 874, 1129, 1192]

Using Selection Sort:
Enter the number of elements: 3
Original List: [1069, 1214, 23]
f(n) = 6
No of swapings: 2
Exact time: 3.0359000447788276e-05
[23, 1069, 1214]
```

PART - B

Q2) Write a Python function to sort n numbers using Insertion sort and analyze the time complexity of your code using the number of comparisons and swaps required and express the same in asymptotic notation. Try the best case, worst case and average case scenarios and report your observations.

Code:

```
# -*- coding: utf-8 -*-
'''This module provides the sorting algorithm,
Insertion Sort. The empirical Analysis is
done with Bubble Sort and Selection Sort.The
module provides the total run time of the algorithm
```

```
along with the sorted list. This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed May 3 2023

Revised on Wed May 7 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
'''


import random
from timeit import default_timer as timer

def createLIST():
    '''
    This function creates a sequence of integer using random
module
    '''
    n = int(input("Enter the number of elements: "))
    seq = []
    for i in range(n):
        seq.append(random.randint(-5,1500))
    print("Original List:", seq)
    return seq

def InsertionSort(seq):
    '''
    This function sorts the entered sequence using Insertion
Sort
```

```
    which has time complexity of O(n^2)
    '''
    fn = 0
    swap = 0
    n = len(seq)
    start = timer()
    for i in range(1,n):
        fn += 1
        temp = seq[i]
        j = i - 1
        while (j >= 0 and seq[j] > temp):
            fn += 1
            seq[j+1] = seq[j]
            j = j-1
            swap += 1
        seq[j+1] = temp
    end = timer()
    print("f(n) =", fn)
    print("No of swapings:", swap)
    print("Exact time:",end - start )
    return seq

#Test cases for this source code:

if __name__ == "__main__":

    #Insertion Sort
    print("Using Insertion Sort: ")
    print(InsertionSort([10,6,4,100,22,44,223,4433]))
    print()
```

Output:

```
Using Insertion Sort:
f(n) = 12
No of swapings: 5
Exact time: 1.4736004231963307e-05
[4, 6, 10, 22, 44, 100, 223, 4433]
```