DAA Project Report

Name: Siddhesh Rajesh Rao

UTA ID: 1001666614

NetID: sxr6615

Programming Language Used: Python 3

Project done: 1. Comparing sorting Algorithms. (Merge, Heap, Quick, Insertion, Bubble)

How to run the code: Use Command Line Interface: *python3 sorting_algorithms.py*

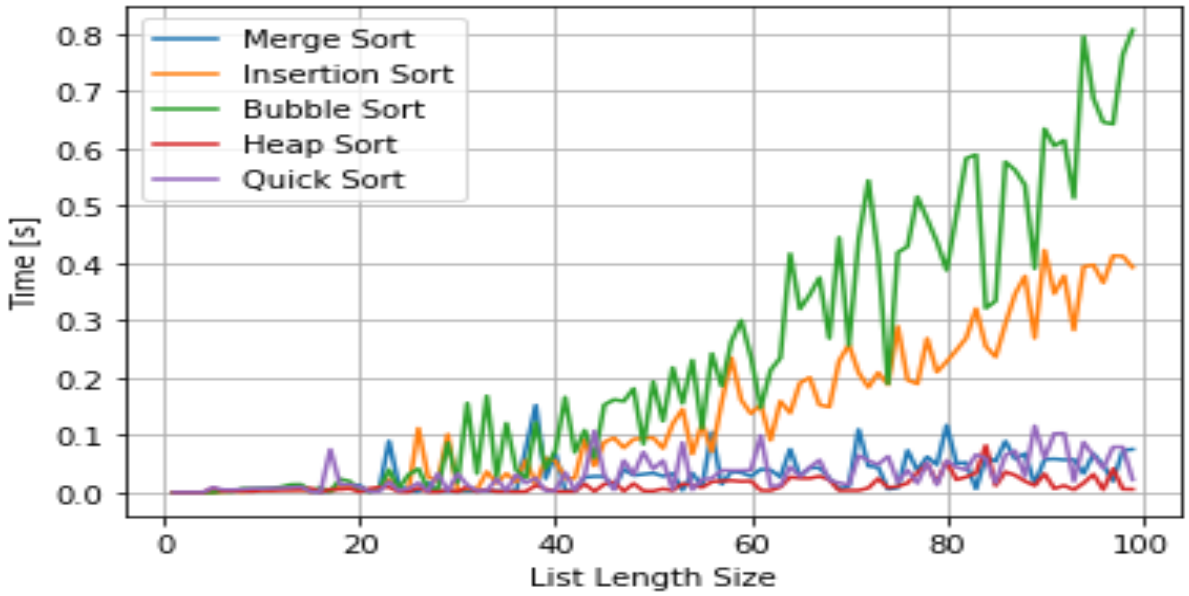
Design of User Interface: After running the code from command line, you will be asked the following:

```
Please enter your choice of sorting algorithm (Enter 'Merge'
/'Heap'/'Quick'/'Insertion'/'Bubble'):
```

Enter one of the following options **(Input is not Case Sensitive)**. Invalid option will by default ask if you want to run a function which compares all algorithms and plots a graph of time vs list size.

```
Do you want to see time vs list size graph for all algorithm
s? (Y/N):
```

If you enter 'y', the function compares all algorithms by generating random list of integers of size (1-1,000) with list size ranging from 1-100 *(Can change list size on line 207 of code)* and plot them on a graph with respect to time. We see that Bubble sort is the worst performing algorithm followed by Insertion sort. Other algorithms follow being better than these two. The Graph signifies the results.



After entering one of the options, you will be asked to enter the size of input list that you want to sort.

```
Enter size of list to be sorted:
```

After you enter the size of list, you will be given an option to generate numbers randomly or enter elements manually.

```
Do you want to randomly generate numbers (Enter Y/N):
```

Your input list will be displayed after you enter elements manually or after generation.

```
Your Input List is:
```

After input list is displayed, the respective algorithm is called, and the list is sorted. Sorted list is printed after the function is executed. Total execution system time is also printed.

Note: An option to see Time vs List Size Graph (for the selected algorithm) will be available only when you generate input list randomly.

```
Do you want to see Time vs List Size Graph? (Y/N):
```

Making an option 'y' will run the respective algorithm by randomly generating integer values in the range of (1-1,000) with a list size ranging from 1-200 *(list size for specific algorithm's time vs list graph can be changed on the following lines of code. Merge = Line 283, Bubble = Line 326, Insertion = Line 369, Quick = Line 454, Heap = Line 411)* and plot them on a graph with respect to time. 'n' will exit the program.

Code Structure:

Class of every sorting algorithm (Example: **Class Bubble, Class Merge, Class Quick, Class Heap, Class Insertion)**, 3 functions used for generating random numbers, getting user input for lists and comparing and plotting all algorithms and a main function.

Objects to call every class is created in main function. (**isort = Insertion(), bsort = Bubble(), msort = Merge(), qsort = Quick(), hsort = Heap()**)

Objects created in main are used to call main functions defined in each class which initiate the sorting algorithm.

Libraries used: *No external libraries are used for developing the algorithms. sys is used for exit function, random is used for generating random numbers, time is used to calculate execution time and use it for graph plotting, matplotlib.pyplot is used for graph plotting.*

Some Test Manual Inputs for every algorithm (Done using random input generation):

Merge Sort:

Please enter your choice of sorting algorithm (Enter 'Merge'/'Heap'/'Quick'/'Insertion'/'Bubble'): merge

Enter size of list to be sorted: 5

Do you want to randomly generate numbers (Enter Y/N): y

Your Input List is: [9561, 5569, 7540, 3183, 2577]

Sorting Via Merge Sort [9561]

Sorting Via Merge Sort [5569]

Sorting Via Merge Sort [5569, 9561]

Sorting Via Merge Sort [7540]

Sorting Via Merge Sort [3183]

Sorting Via Merge Sort [2577]

Sorting Via Merge Sort [2577, 3183]

Sorting Via Merge Sort [2577, 3183, 7540]

Sorting Via Merge Sort [2577, 3183, 5569, 7540, 9561]

Sorted List via Merge Sort is: [2577, 3183, 5569, 7540, 9561]

Total Execution Time: 0.328125

Quick Sort:

Please enter your choice of sorting algorithm (Enter 'Merge'/'Heap'/'Quick'/'Insertion'/'Bubble'): quick

Enter size of list to be sorted: 5

Do you want to randomly generate numbers (Enter Y/N): y

Your Input List is: [2503, 2020, 4460, 5133, 2779]

Sorting via quick Sort: [2503, 2020, 4460, 5133, 2779]

Sorting via quick Sort: [2503, 2020, 2779, 5133, 4460]

Sorting via quick Sort: [2020, 2503, 2779, 5133, 4460]

Sorting via quick Sort: [2020, 2503, 2779, 5133, 4460]

Sorting via quick Sort: [2020, 2503, 2779, 5133, 4460]

Sorting via quick Sort: [2020, 2503, 2779, 5133, 4460]

Sorting via quick Sort: [2020, 2503, 2779, 5133, 4460]

Sorting via quick Sort: [2020, 2503, 2779, 4460, 5133]

Sorting via quick Sort: [2020, 2503, 2779, 4460, 5133]

Sorting via quick Sort: [2020, 2503, 2779, 4460, 5133]

Sorting via quick Sort: [2020, 2503, 2779, 4460, 5133]

Your Sorted List via Quick Sort is: [2020, 2503, 2779, 4460, 5133]

Total Execution Time: 0.34375

Heap Sort:

Please enter your choice of sorting algorithm (Enter 'Merge'/'Heap'/'Quick'/'Insertion'/'Bubble'): heap

Enter size of list to be sorted: 5

Do you want to randomly generate numbers (Enter Y/N): y

Your Input List is: [9955, 5341, 8135, 2795, 3715]

Sorting via Heap Sort: [9955, 5341, 8135, 2795, 3715]

Sorting via Heap Sort: [8135, 5341, 3715, 2795, 9955]

Sorting via Heap Sort: [5341, 2795, 3715, 8135, 9955]

Sorting via Heap Sort: [3715, 2795, 5341, 8135, 9955]

Your Sorted List via Heap Sort is [2795, 3715, 5341, 8135, 9955]

Total Execution Time: 0.328125

Insertion Sort:

Please enter your choice of sorting algorithm (Enter 'Merge'/'Heap'/'Quick'/'Insertion'/'Bubble'): insertion

Enter size of list to be sorted: 5

Do you want to randomly generate numbers (Enter Y/N): y

Your Input List is: [3713, 6813, 3201, 8260, 1633]

[3713, 6813, 6813, 8260, 1633]

[3713, 3713, 6813, 8260, 1633]

[3201, 3713, 6813, 8260, 8260]

[3201, 3713, 6813, 6813, 8260]

[3201, 3713, 3713, 6813, 8260]

[3201, 3201, 3713, 6813, 8260]

Your Sorted List Via Insertion Sort is: [1633, 3201, 3713, 6813, 8260]

Total Execution Time: 0.328125

Bubble Sort:

Please enter your choice of sorting algorithm (Enter 'Merge'/'Heap'/'Quick'/'Insertion'/'Bubble'): bubble

Enter size of list to be sorted: 5

Do you want to randomly generate numbers (Enter Y/N): y

Your Input List is: [9991, 4170, 5665, 3880, 7547]

Sorting via Bubble Sort: [9991, 4170, 5665, 3880, 7547]

Sorting via Bubble Sort: [4170, 9991, 5665, 3880, 7547]

Sorting via Bubble Sort: [4170, 5665, 9991, 3880, 7547]

Sorting via Bubble Sort: [4170, 5665, 3880, 9991, 7547]

Sorting via Bubble Sort: [4170, 5665, 3880, 7547, 9991]

Sorting via Bubble Sort: [4170, 5665, 3880, 7547, 9991]

Sorting via Bubble Sort: [4170, 3880, 5665, 7547, 9991]

Sorting via Bubble Sort: [4170, 3880, 5665, 7547, 9991]

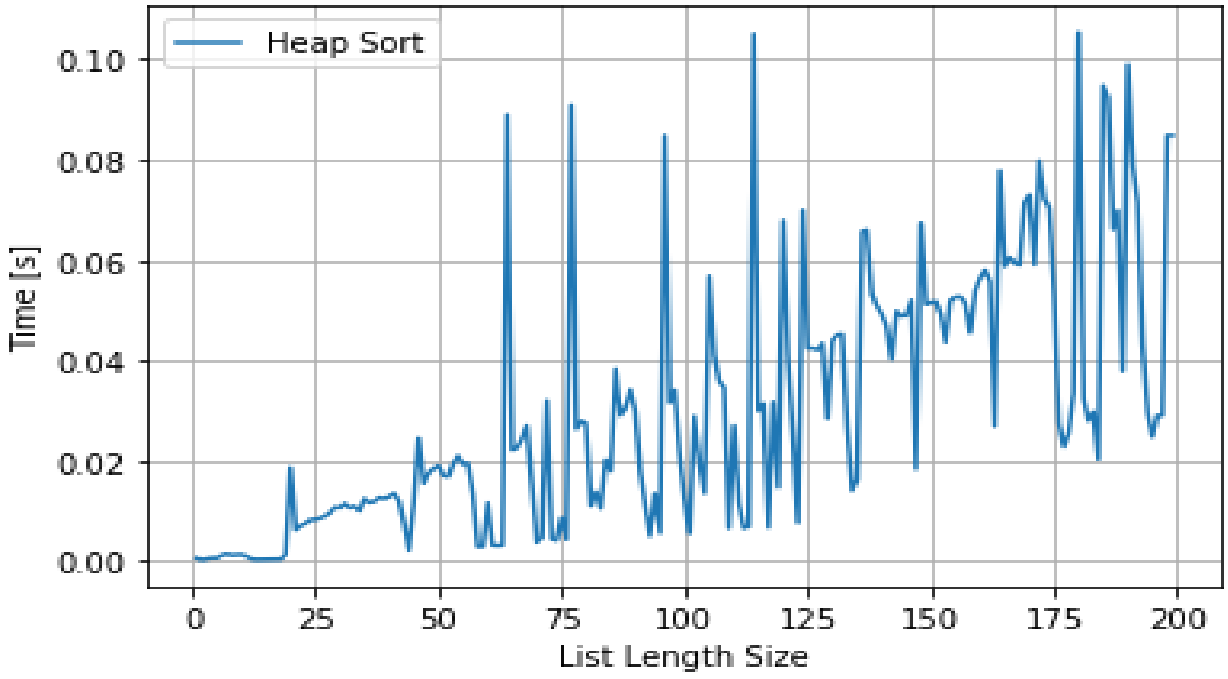
Sorting via Bubble Sort: [3880, 4170, 5665, 7547, 9991]

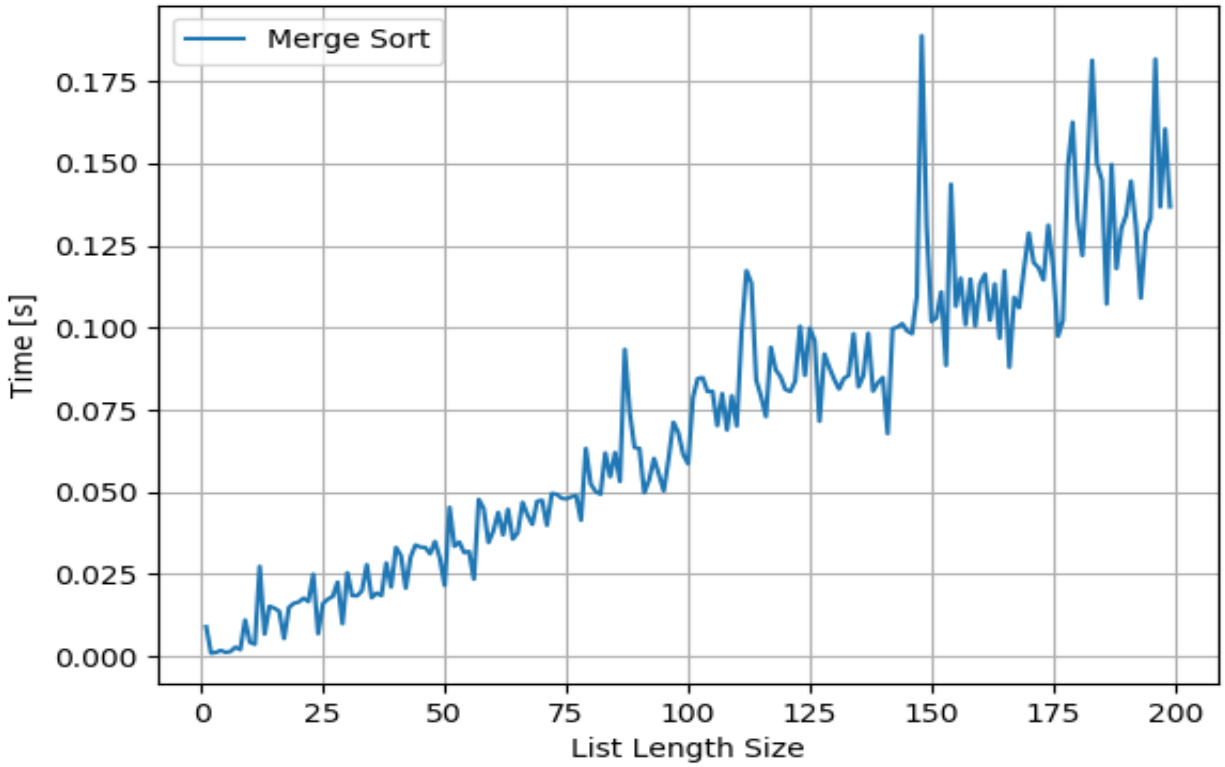
Sorting via Bubble Sort: [3880, 4170, 5665, 7547, 9991]

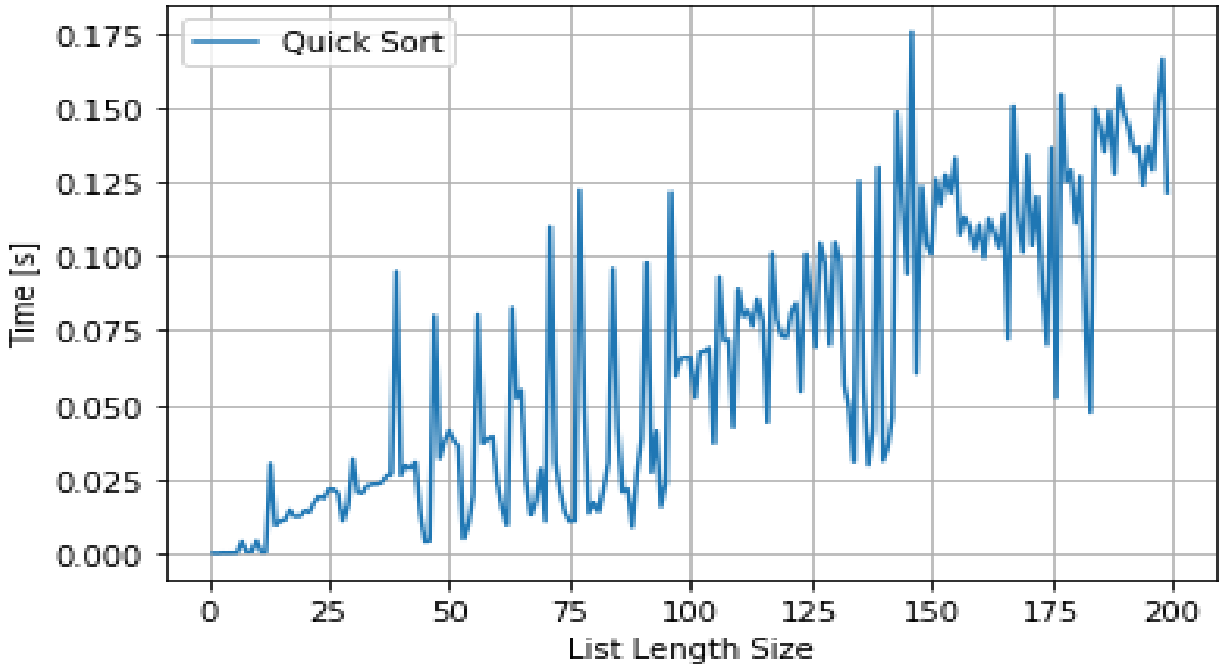
Your Sorted List via Bubble Sort is: [3880, 4170, 5665, 7547, 9991]

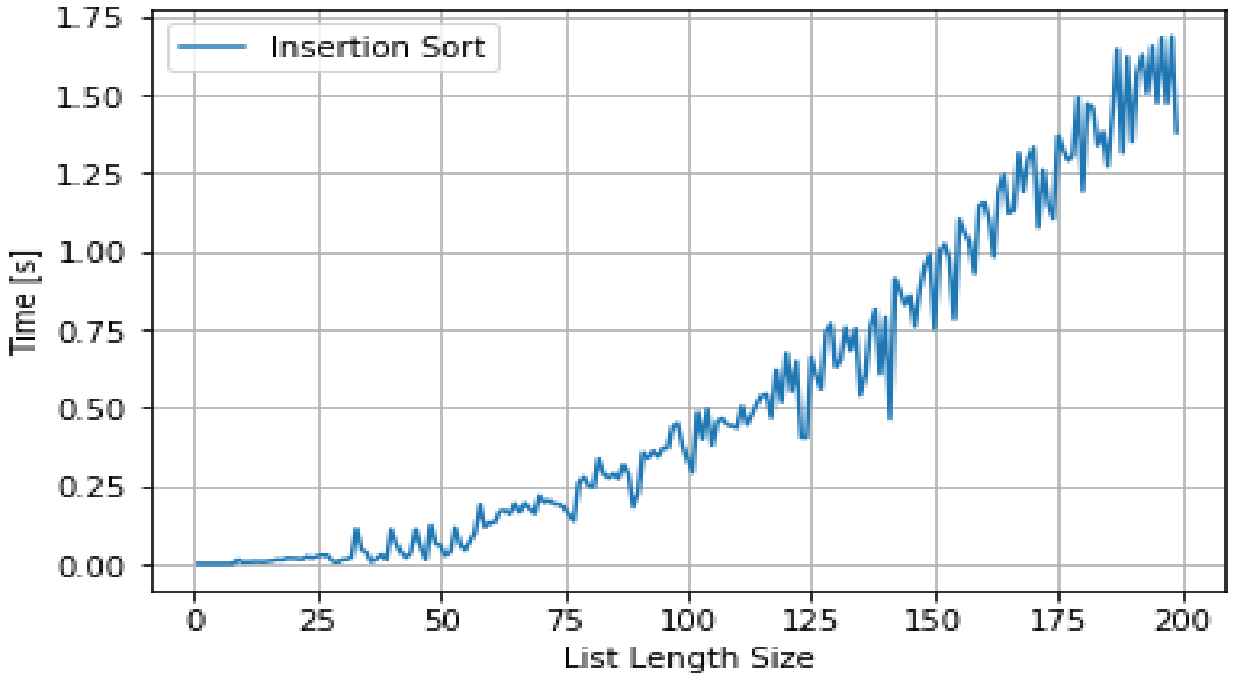
Total Execution Time: 0.359375

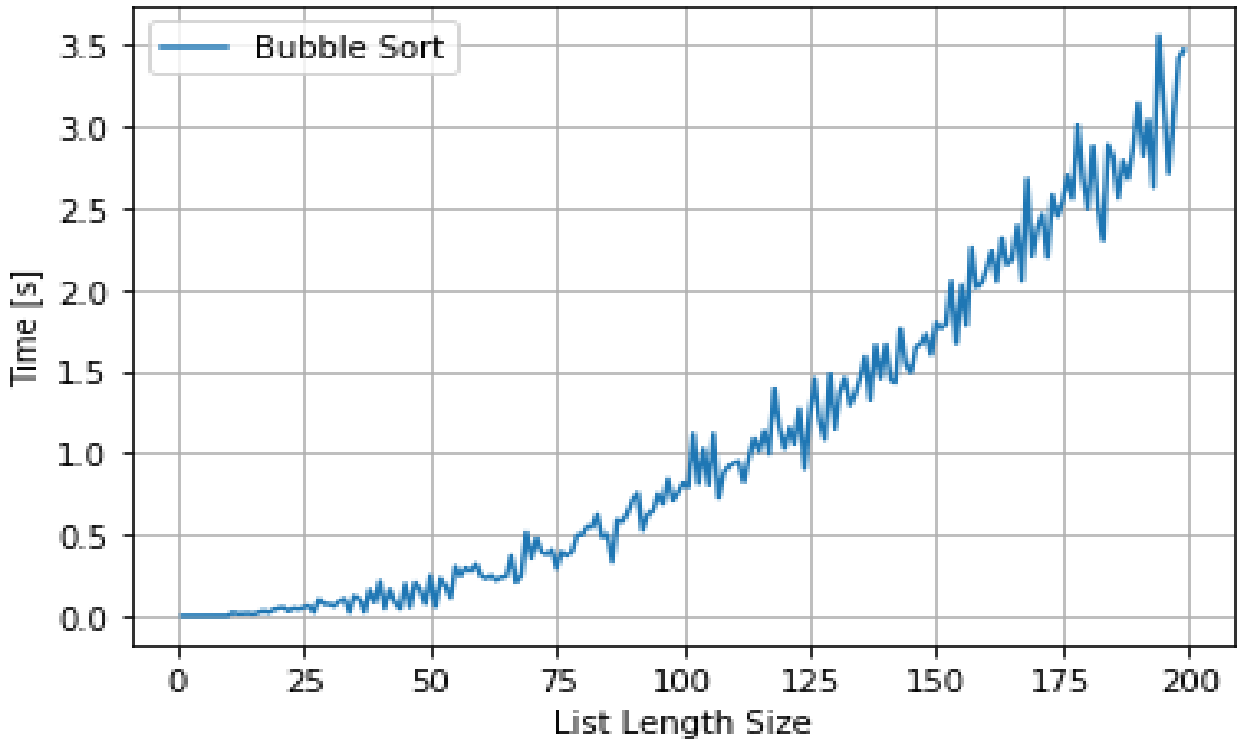
Graphs of Time vs List Length (For List Size 1-200):




Merge Sort
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Time [s]
0
25
50
75
100
125
150
175
200
List Length Size


Quick Sort
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Time [s]
0
25
50
75
100
125
150
175
200
List Length Size


Insertion Sort
Time [s]
List Length Size
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0
25
50
75
100
125
150
175
200


Bubble Sort
Time [s]
List Length Size
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
25
50
75
100
125
150
175
200