National University of Computer & Emerging Sciences (FAST-NU)

Course: CS2009:Design and Analysis of Algorithms

Project Report

Implementation and Evaluation of Sorting Algorithms

Course Instructor: Dr. Muhammad Atif Tahir

Prepared by:

Sabah Mawani (20k-393)

Arooba Moin (20k-0213)

Abstract:

The following project was created by evaluating sorting algorithms based on its own unique algorithm and then calculating its unique time complexities. This benchmarking has been achieved by implementing and testing the algorithms in an IDE and a programming language in order to simulate and evaluate the working of the algorithms.

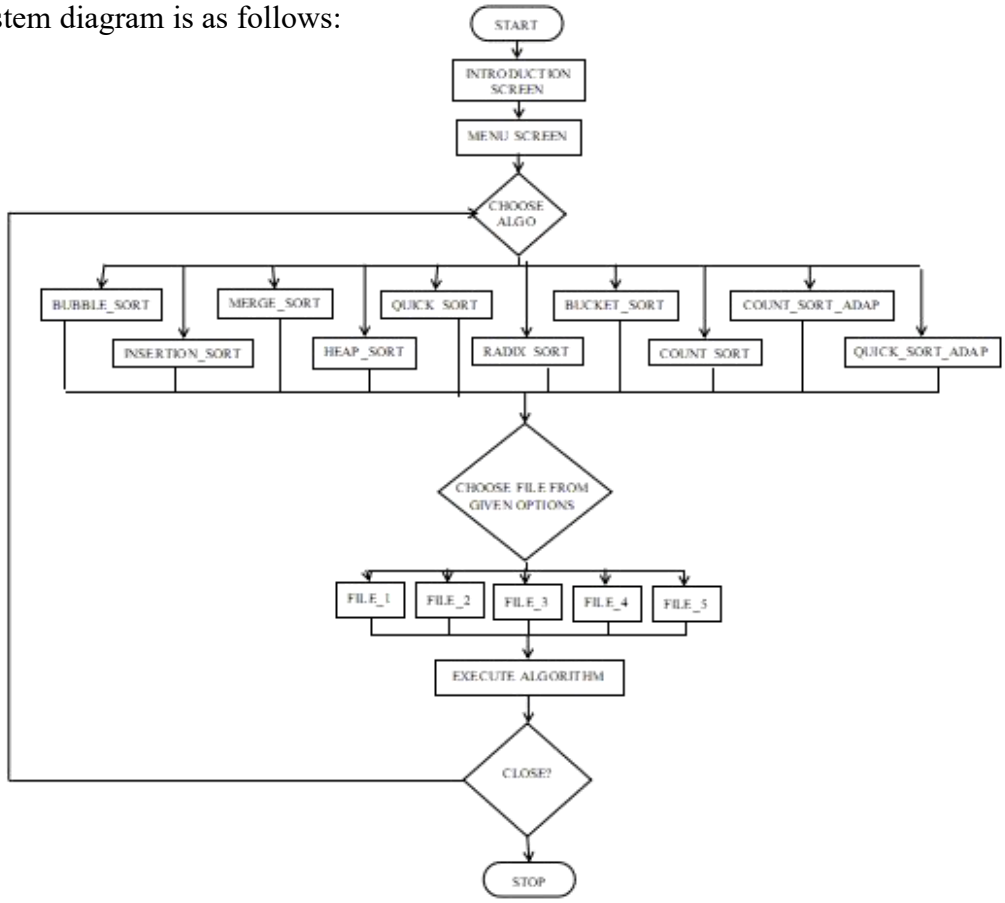
Introduction:

The problem at hand was to evaluate the output of various sorting algorithms including:

- Insertion Sorting Algorithm
- Bubble Sorting Algorithm
- Merge Sorting Algorithm
- Heap Sorting Algorithm
- Quick Sorting Algorithm
- Radix Sorting Algorithm
- Bucket Sorting Algorithm
- Counting Sorting Algorithm
- An Adaptation of the Quick Sort Algorithm {7.4.5. from book}
- An Adaptation of the Count Sort Algorithm {8.2.4. from book}

Programming Design:

The system diagram is as follows:



Experimental Setup:

The system takes input in the form of ".csv" files that ensures a certain format to be followed. The user will be able to select any . .csv file in their sorting algorithm
Further details of the software and frameworks
used include:
• IDE: VSCODE
• Programming Language: Python 3
• Framework for Graph visualization: Pygame

Results:

Input Files

	A	B	C	D	E	F	G	H	I	J	K
1	463	109	284	65	119						
2	123	211	440	150	207						
3	394	27	382	318	144						
4	467	183	375	265	351						
5	258	283	464	493	474						
6	99	242	199	387	148						
7	274	36	239	312	50						
8	312	129	113	140	478						
9	379	155	169	240	278						
10	433	47	497	326	276						
11	280	193	74	175	299						
12	218	456	203	224	285						
13	51	44	396	427	218						
14	446	245	369	314	248						
15	119	421	87	88	97						
16	164	370	76	432	64						
17	194	208	48	176	466						
18	105	340	28	447	461						
19	459	131	289	398	305						
20	477	344	173	66	488						
21	91	342	324	458	367						
22	493	389	366	208	192						
23	404	142	384	466	128						
24	52	229	173	191	410						
25	292	386	477	494	414						
26											
27											
28											

Insertion | Bubble | Merge | Heap | Quick | Radix | Bucket | Counting | Qadap | Cadap

```
Time taken by insertion sort:  6.786275625228882
Time taken by bubble sort:  33.595118045806885
Time taken by merge sort:  13.999305248260498
Time taken by heap sort:  29.355466604232788
Time taken by quick sort:  40.709811210632324
Time taken by radix sort:  9.341968297958374
Time taken by bucket sort:  4.202390432357788
Time taken by count sort:  3.866499900817871
Time taken by quick sort adaptation:  43.61421227455139
Enter lower bound: 5
Enter upper bound: 400
Time taken by count sort adaptation:  11.768250942230225
```

Conclusion:

We were able to visualize how sorting algorithms can be implemented, as well as their time complexities and determine which ones were faster, as a result of implementing this system.

We were also able to evaluate and confirm the complexities of algorithms using a diverse set of inputs.

References:

https://www.geeksforgeeks.org **{Sorting Algorithms}**
https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/ **{Time Complexities}**