Assignment 3 - WRITEUP.pdf

Introduction:

For the write-up, I will be focusing on my code and data collection. Firstly I want to mention that some code assignments that produce the sorting algorithm were given to me in the form of pseudocode. My assignment consisted of different parts but was mostly concentrated on the sorting algorithms below:

Insertion Sort

Heap Sort

Bash Sort

Quick Sort

Those functions were used to sort an array of integers from the smallest to the largest number.

Makefile:

The Makefile below was used to create only 1 executable file, which would be called - ***sorting*** *(usage explained in README.md)*

```
CC = clang
CFLAGS = -Wall -Wextra -Werror -Wpedantic -g

.PHONY: all clean

all: sorting

sorting: sorting.o batcher.o heap.o insert.o quick.o stats.o
	$(CC) sorting.o batcher.o heap.o insert.o quick.o stats.o -o sorting

bathcher.o: batcher.c batcher.h stats.h
	$(CC) $(CFLAGS) -c batcher.c -o bathcher.o

heap.o: heap.c heap.h stats.h
	$(CC) $(CFLAGS) -c heap.c -o heap.o

insert.o: insert.c insert.h stats.h
	$(CC) $(CFLAGS) -c insert.c -o insert.o

quick.o: quick.c quick.h stats.h
	$(CC) $(CFLAGS) -c quick.c -o quick.o

stats.o: stats.c stats.h
	$(CC) $(CFLAGS) -c stats.c -o stats.o

clean:
	rm -f sorting sorting.o
```

Sorting + Data Collection

In the table below I describe the sorting algorithm used and data collection. I was, unfortunately, unable to collect accurate data, that would allow me for graph creation, therefore I decided not to make one as it would not be accurate.

My Algorithm

Below is the print output of my algorithm

```
Insertion Sort, 10 elements, 41 moves, 29 compares,
 34732749     42067670     104268822     134750049     182960600
538219612     954916333     966879077     989854347     994582085

Heap Sort, 10 elements, 0 moves, 1 compares,
 34732749     42067670     104268822     134750049     182960600
538219612     954916333     966879077     989854347     994582085

Quick Sort, 10 elements, 162 moves, 45 compares,
 34732749     42067670     104268822     134750049     182960600
538219612     954916333     966879077     989854347     994582085

Batcher Sort, 10 elements, 39 moves, 16 compares,
 34732749       42067670       104268822       134750049       182960600
538219612       954916333       966879077       989854347       994582085
```

Sorting Algorithm	elements	First Value	Last Value	Moves	Compares
Insertion	10	34732749	994582085	**41**	**29**
Heap	10	34732749	994582085	**0**	**1**
Quick	10	34732749	994582085	**162**	**45**
Batcher	10	34732749	994582085	**39**	**16**

Given Algorithm

Below is the print output of the given algorithm. I was predicting my algorithm to come out this way.

```
Insertion Sort, 10 elements, 41 moves, 29 compares
     34732749     42067670    104268822    134750049    182960600
    538219612    954916333    966879077    989854347    994582085
Batcher Sort, 10 elements, 39 moves, 31 compares
     34732749     42067670    104268822    134750049    182960600
    538219612    954916333    966879077    989854347    994582085
Heap Sort, 10 elements, 93 moves, 44 compares
     34732749     42067670    104268822    134750049    182960600
    538219612    954916333    966879077    989854347    994582085
Quick Sort, 10 elements, 51 moves, 22 compares
     34732749     42067670    104268822    134750049    182960600
    538219612    954916333    966879077    989854347    994582085
```

Sorting Algorithm	elements	First Value	Last Value	Moves	Compares
Insertion	10	34732749	994582085	**41**	**29**
Heap	10	34732749	994582085	93	44
Quick	10	34732749	994582085	51	22
Batcher	10	34732749	994582085	**39**	32

Data Description:

I was have implemented the algorithms successfully. All the algorithms sort sequences from the smallest to the largest value. In addition, the moves and compares have been successfully implemented in the Insertion algorithm. However, I have not implemented the data collection in the other ones successfully, which made it unable for me to be able to compare them to a sufficient degree enough. Therefore I would not be able to assess my hypothesis of the execution of the different algorithms.

Conclusion:

From my data shown I was unfortunately unable to point out the most efficient algorithm, however, I have realized certain points:

- The optimal sorting algorithm would be ‘n’ therefore the sorting through the array only once, however since that is not possible, we are trying to get as close to the number as possible
- The simplest algorithm runs n^2 and therefore the goal of building an algorith would be to have it between n^2 and n

- Algorithm is more important than the processor speed. I learned that some sorting algorithms can be executed in parallel, making them even faster, however, a good algorithm is crucial to the speed of the overall program, when dealing with a lot of data