Bicol University
College of Science
CSIT Department
2ND semester
AY 2022-2023

PROGRAMMING PROJECT 1
CS111 – DESIGN ANALYSIS OF ALGORITHMS

Erjay Cloefe

Ian Rhey Banares

Ignacio Tabug III

Aragorn Muncal

BSCS-2A

Use the different sorting algorithms **(Selection Sort, Bubble Sort, Insertion Sort, Mergesort, Quicksort, and Heapsort)** to sort **N randomly generated integer**s and then **analyze their run time.** The project will have two parts, one for the basic implementation of the program that sorts the integers, and the second for analyzing what you see in terms of performance.

Structure of the code

Line	
9-12	header files
14-15	constant definition
18-36	prototype function declaration
49-242	main function
40-42	declaration of variables
44-45	initialization of arrays
48-52	getting input N from user
53-56	generating random integers and displaying
59-63	getting input X from user
64-69	generating sorted numbers and displaying
72-77	initialization of arrays for sorting
78-85	copying contents of arrays
87-94	clocking, executing, and displaying insertion sort and time taken
96-103	clocking, executing, and displaying bubble sort and time taken
105-112	clocking, executing, and displaying selection sort and time taken
114-121	clocking, executing, and displaying merge sort and time taken
123-130	clocking, executing, and displaying quick sort and time taken
132-140	clocking, executing, and displaying heap sort and time taken
143-148	initialization of arrays
149-156	copying contents of arrays
158-160	finding runtime of array sorted with insertion sort algorithm

162-164	finding runtime of array sorted with bubble sort algorithm
166-168	finding runtime of array sorted with selection sort algorithm
170-175	finding runtime of array sorted with merge sort algorithm
177-182	finding runtime of array sorted with quick sort algorithm
184-189	finding runtime of array sorted with heap sort algorithm
193-231	file printing
235-239	free allocated memory
246-265	function for generating random numbers
266-272	function for generating sorted numbers
274-293	function for insertion sort
296-309	function for selection sort
312-323	function for bubble sort
325-335	function for merge sort
336-382	function for merging subarrays for merge sort
384-390	function for quick sort
392-404	function for partition used in quick sort
406-417	function for heap sort
418-435	function for heapify used in heap sort
438-442	function for swapping two elements
444-469	function for displaying array
470-496	function for printing array in the file

I. Purpose of the Program

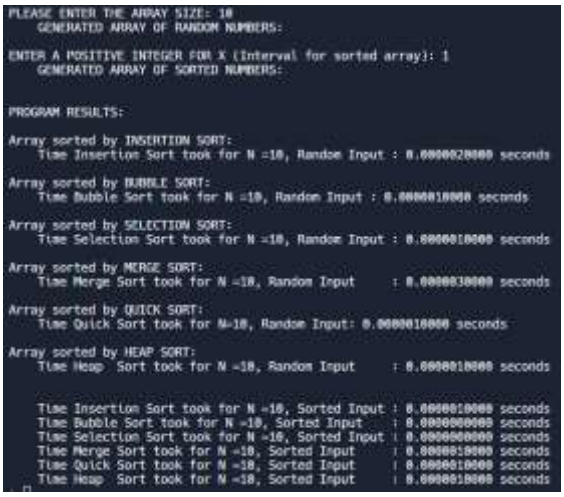
The program contains several Sorting Algorithms such as Selection Sort, Bubble Sort, Insertion Sort, Merge Sort, Quicksort, and Heapsort. It is instructed to calculate the running time of all the sorting algorithms using only two types of an array which are the randomly generated array and the sorted array. Given outputs are expressed in seconds to sort the array.

II. Screenshots and Actual Execution

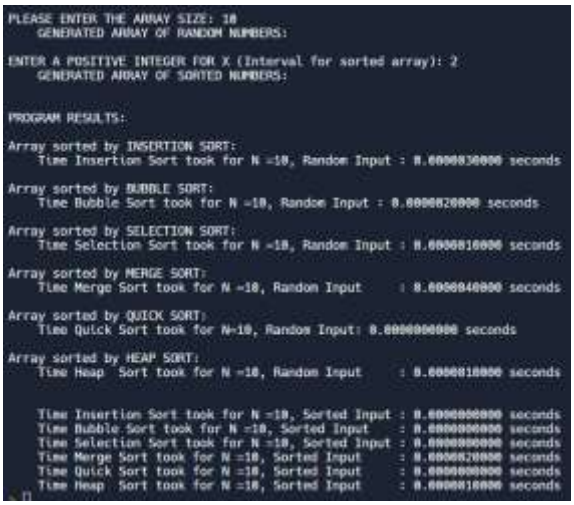
showing the actual execution of the program. (Random at the top - Sorted at the bottom)

N =10 X = 1 , 2, 3

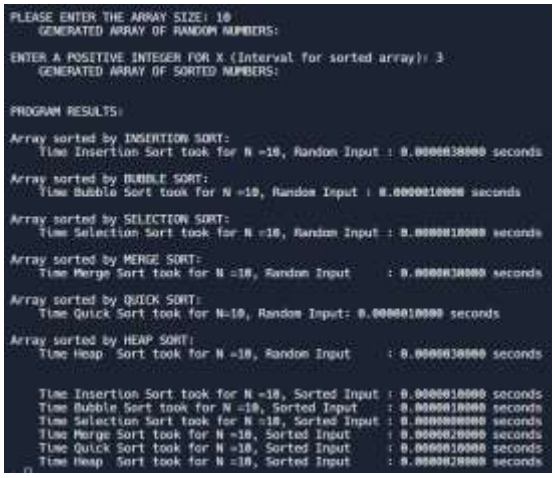
TEST I



TEST II



TEST III



N =100 X = 1 , 2, 3

TEST I

```
PLEASE ENTER THE ARRAY SIZE: 100
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 1
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =100, Random Input : 0.0000050000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =100, Random Input : 0.0000120000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =100, Random Input : 0.0000090000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =100, Random Input     : 0.0000130000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=100, Randoe Input: 0.0000040000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =100, Random Input     : 0.0000060000 seconds


    Time Insertion Sort took for N =100, Sorted Input : 0.0000010000 seconds
    Time Bubble Sort took for N =100, Sorted Input    : 0.0000010000 seconds
    Time Selection Sort took for N =100, Sorted Input : 0.0000010000 seconds
    Time Merge Sort took for N =100, Sorted Input     : 0.0000090000 seconds
    Time Quick Sort took for N =100, Sorted Input     : 0.0000110000 seconds
    Time Heap  Sort took for N =100, Sorted Input     : 0.0000050000 seconds
```

TEST II

```
PLEASE ENTER THE ARRAY SIZE: 100
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 2
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =100, Random Input : 0.0000060000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =100, Random Input : 0.0000130000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =100, Random Input : 0.0000100000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =100, Random Input     : 0.0000140000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=100, Random Input: 0.0000040000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =100, Random Input     : 0.0000060000 seconds


    Time Insertion Sort took for N =100, Sorted Input : 0.0000010000 seconds
    Time Bubble Sort took for N =100, Sorted Input    : 0.0000010000 seconds
    Time Selection Sort took for N =100, Sorted Input : 0.0000010000 seconds
    Time Merge Sort took for N =100, Sorted Input     : 0.0000080000 seconds
    Time Quick Sort took for N =100, Sorted Input     : 0.0000110000 seconds
    Time Heap  Sort took for N =100, Sorted Input     : 0.0000060000 seconds
```

TEST III

```
PLEASE ENTER THE ARRAY SIZE: 100
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 3
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =100, Random Input : 0.0000060000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =100, Random Input : 0.0000640000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =100, Random Input : 0.0000150000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =100, Random Input     : 0.0000100000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=100, Random Input: 0.0000130000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =100, Random Input     : 0.0000140000 seconds


    Time Insertion Sort took for N =100, Sorted Input : 0.0000010000 seconds
    Time Bubble Sort took for N =100, Sorted Input    : 0.0000010000 seconds
    Time Selection Sort took for N =100, Sorted Input : 0.0000090000 seconds
    Time Merge Sort took for N =100, Sorted Input     : 0.0000140000 seconds
    Time Quick Sort took for N =100, Sorted Input     : 0.0000100000 seconds
    Time Heap  Sort took for N =100, Sorted Input     : 0.0000130000 seconds
```

N =1000 X = 1 , 2, 3

TEST I

```
PLEASE ENTER THE ARRAY SIZE: 1000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 1
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =1000, Random Input : 0.0001880000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =1000, Random Input : 0.0005830000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =1000, Random Input : 0.0005910000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =1000, Random Input     : 0.0001260000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=1000, Random Input: 0.0000420000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =1000, Random Input     : 0.0000880000 seconds


    Time Insertion Sort took for N =1000, Sorted Input : 0.0000020000 seconds
    Time Bubble Sort took for N =1000, Sorted Input    : 0.0000020000 seconds
    Time Selection Sort took for N =1000, Sorted Input : 0.0000020000 seconds
    Time Merge Sort took for N =1000, Sorted Input     : 0.0000030000 seconds
    Time Quick Sort took for N =1000, Sorted Input     : 0.0000470000 seconds
    Time Heap  Sort took for N =1000, Sorted Input     : 0.0000610000 seconds
```

TEST II

```
PLEASE ENTER THE ARRAY SIZE: 1000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 2
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =1000, Random Input : 0.0002700000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =1000, Random Input : 0.0006540000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =1000, Random Input : 0.0004670000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =1000, Random Input     : 0.0001040000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=1000, Random Input: 0.0000470000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =1000, Random Input     : 0.0000720000 seconds


    Time Insertion Sort took for N =1000, Sorted Input : 0.0000020000 seconds
    Time Bubble Sort took for N =1000, Sorted Input    : 0.0000020000 seconds
    Time Selection Sort took for N =1000, Sorted Input : 0.0000020000 seconds
    Time Merge Sort took for N =1000, Sorted Input     : 0.0000760000 seconds
    Time Quick Sort took for N =1000, Sorted Input     : 0.0000640000 seconds
    Time Heap  Sort took for N =1000, Sorted Input     : 0.0000750000 seconds
```

TEST III

```
PLEASE ENTER THE ARRAY SIZE: 1000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 3
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =1000, Random Input : 0.0002250000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =1000, Random Input : 0.0006490000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =1000, Random Input : 0.0004750000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =1000, Random Input     : 0.0001320000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=1000, Random Input: 0.0000460000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =1000, Random Input     : 0.0000770000 seconds


    Time Insertion Sort took for N =1000, Sorted Input : 0.0000010000 seconds
    Time Bubble Sort took for N =1000, Sorted Input    : 0.0000020000 seconds
    Time Selection Sort took for N =1000, Sorted Input : 0.0000020000 seconds
    Time Merge Sort took for N =1000, Sorted Input     : 0.0000770000 seconds
    Time Quick Sort took for N =1000, Sorted Input     : 0.0000650000 seconds
    Time Heap  Sort took for N =1000, Sorted Input     : 0.0000630000 seconds
```

N =10000 X = 1 , 2, 3

TEST I

```
PLEASE ENTER THE ARRAY SIZE: 10000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 1
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =10000, Random Input : 0.0163060000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =10000, Random Input : 0.0436070000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =10000, Random Input : 0.0380900000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =10000, Random Input     : 0.0013510000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=10000, Random Input: 0.0005860000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =10000, Random Input     : 0.0009460000 seconds


    Time Insertion Sort took for N =10000, Sorted Input : 0.0000130000 seconds
    Time Bubble Sort took for N =10000, Sorted Input    : 0.0000140000 seconds
    Time Selection Sort took for N =10000, Sorted Input : 0.0000160000 seconds
    Time Merge Sort took for N =10000, Sorted Input     : 0.0008690000 seconds
    Time Quick Sort took for N =10000, Sorted Input     : 0.0476730000 seconds
    Time Heap  Sort took for N =10000, Sorted Input     : 0.0005920000 seconds
```

TEST II

```
PLEASE ENTER THE ARRAY SIZE: 10000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 2
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =10000, Random Input : 0.0218660000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =10000, Random Input : 0.0572620000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =10000, Random Input : 0.0409700000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =10000, Random Input     : 0.0012670000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=10000, Random Input: 0.0006580000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =10000, Random Input     : 0.0010350000 seconds


    Time Insertion Sort took for N =10000, Sorted Input : 0.0000130000 seconds
    Time Bubble Sort took for N =10000, Sorted Input    : 0.0000130000 seconds
    Time Selection Sort took for N =10000, Sorted Input : 0.0000140000 seconds
    Time Merge Sort took for N =10000, Sorted Input     : 0.0007910000 seconds
    Time Quick Sort took for N =10000, Sorted Input     : 0.0474190000 seconds
    Time Heap  Sort took for N =10000, Sorted Input     : 0.0005080000 seconds
```

TEST III

```
PLEASE ENTER THE ARRAY SIZE: 10000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 3
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =10000, Random Input : 0.0151660000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =10000, Random Input : 0.0491120000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =10000, Random Input : 0.0468180000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =10000, Random Input     : 0.0014450000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=10000, Random Input: 0.0006140000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =10000, Random Input     : 0.0011150000 seconds


    Time Insertion Sort took for N =10000, Sorted Input : 0.0000120000 seconds
    Time Bubble Sort took for N =10000, Sorted Input    : 0.0000120000 seconds
    Time Selection Sort took for N =10000, Sorted Input : 0.0000160000 seconds
    Time Merge Sort took for N =10000, Sorted Input     : 0.0007980000 seconds
    Time Quick Sort took for N =10000, Sorted Input     : 0.0607940000 seconds
    Time Heap  Sort took for N =10000, Sorted Input     : 0.0006810000 seconds
```

N =10000 X = 1 , 2, 3

TEST I

```
PLEASE ENTER THE ARRAY SIZE: 100000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 1
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =100000, Random Input : 1.5888880000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =100000, Random Input : 12.9510680000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =100000, Random Input : 3.6960280000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =100000, Random Input     : 0.0157940000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=100000, Random Input: 0.0074010000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =100000, Random Input     : 0.0125660000 seconds


    Time Insertion Sort took for N =100000, Sorted Input : 0.0001260000 seconds
    Time Bubble Sort took for N =100000, Sorted Input    : 0.0001160000 seconds
    Time Selection Sort took for N =100000, Sorted Input : 0.0001310000 seconds
    Time Merge Sort took for N =100000, Sorted Input     : 0.0031500000 seconds
    Time Quick Sort took for N =100000, Sorted Input     : 4.9745740000 seconds
    Time Heap  Sort took for N =100000, Sorted Input     : 0.0062090000 seconds
> 
```

TEST II

```
> make -s
> ./main

PLEASE ENTER THE ARRAY SIZE: 100000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 2
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =100000, Random Input : 1.7493520000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =100000, Random Input : 15.0265210000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =100000, Random Input : 3.9613470000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =100000, Random Input     : 0.0153540000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=100000, Random Input: 0.0085400000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =100000, Random Input     : 0.0140480000 seconds


    Time Insertion Sort took for N =100000, Sorted Input : 0.0004260000 seconds
    Time Bubble Sort took for N =100000, Sorted Input    : 0.0006260000 seconds
    Time Selection Sort took for N =100000, Sorted Input : 0.0007530000 seconds
    Time Merge Sort took for N =100000, Sorted Input     : 0.0066970000 seconds
    Time Quick Sort took for N =100000, Sorted Input     : 6.6323080000 seconds
    Time Heap  Sort took for N =100000, Sorted Input     : 0.0087360000 seconds
> 
```

TEST III

```
PLEASE ENTER THE ARRAY SIZE: 100000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 3
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =100000, Random Input : 2.0296390000 seconds

Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =100000, Random Input : 14.0648280000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =100000, Random Input : 4.2384030000 seconds

Array sorted by MERGE SORT:
    Time Merge Sort took for N =100000, Random Input     : 0.0141650000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=100000, Random Input: 0.0067090000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =100000, Random Input     : 0.0126820000 seconds


    Time Insertion Sort took for N =100000, Sorted Input : 0.0003270000 seconds
    Time Bubble Sort took for N =100000, Sorted Input    : 0.0005470000 seconds
    Time Selection Sort took for N =100000, Sorted Input : 0.0005480000 seconds
    Time Merge Sort took for N =100000, Sorted Input     : 0.0082680000 seconds
    Time Quick Sort took for N =100000, Sorted Input     : 5.3205270000 seconds
    Time Heap  Sort took for N =100000, Sorted Input     : 0.0083100000 seconds
> 
```

N =1000000 X = 1

```
PLEASE ENTER THE ARRAY SIZE: 1000000
    GENERATED ARRAY OF RANDOM NUMBERS:

ENTER A POSITIVE INTEGER FOR X (Interval for sorted array): 1
    GENERATED ARRAY OF SORTED NUMBERS:


PROGRAM RESULTS:

Array sorted by INSERTION SORT:
    Time Insertion Sort took for N =1000000, Random Input : 188.6421520000 seconds


Array sorted by BUBBLE SORT:
    Time Bubble Sort took for N =1000000, Random Input : 1570.9249410000 seconds

Array sorted by SELECTION SORT:
    Time Selection Sort took for N =1000000, Random Input : 423.3866860000 seconds


Array sorted by MERGE SORT:
    Time Merge Sort took for N =1000000, Random Input     : 0.2004340000 seconds

Array sorted by QUICK SORT:
    Time Quick Sort took for N=1000000, Random Input: 0.1062880000 seconds

Array sorted by HEAP SORT:
    Time Heap  Sort took for N =1000000, Random Input     : 0.2645460000 seconds


    Time Insertion Sort took for N =1000000, Sorted Input : 0.0014030000 seconds
    Time Bubble Sort took for N =1000000, Sorted Input    : 0.0015310000 seconds
    Time Selection Sort took for N =1000000, Sorted Input : 0.0018490000 seconds
    Time Merge Sort took for N =1000000, Sorted Input     : 0.0884250000 seconds
signal: segmentation fault (core dumped)
```

III. Analysis of the Output

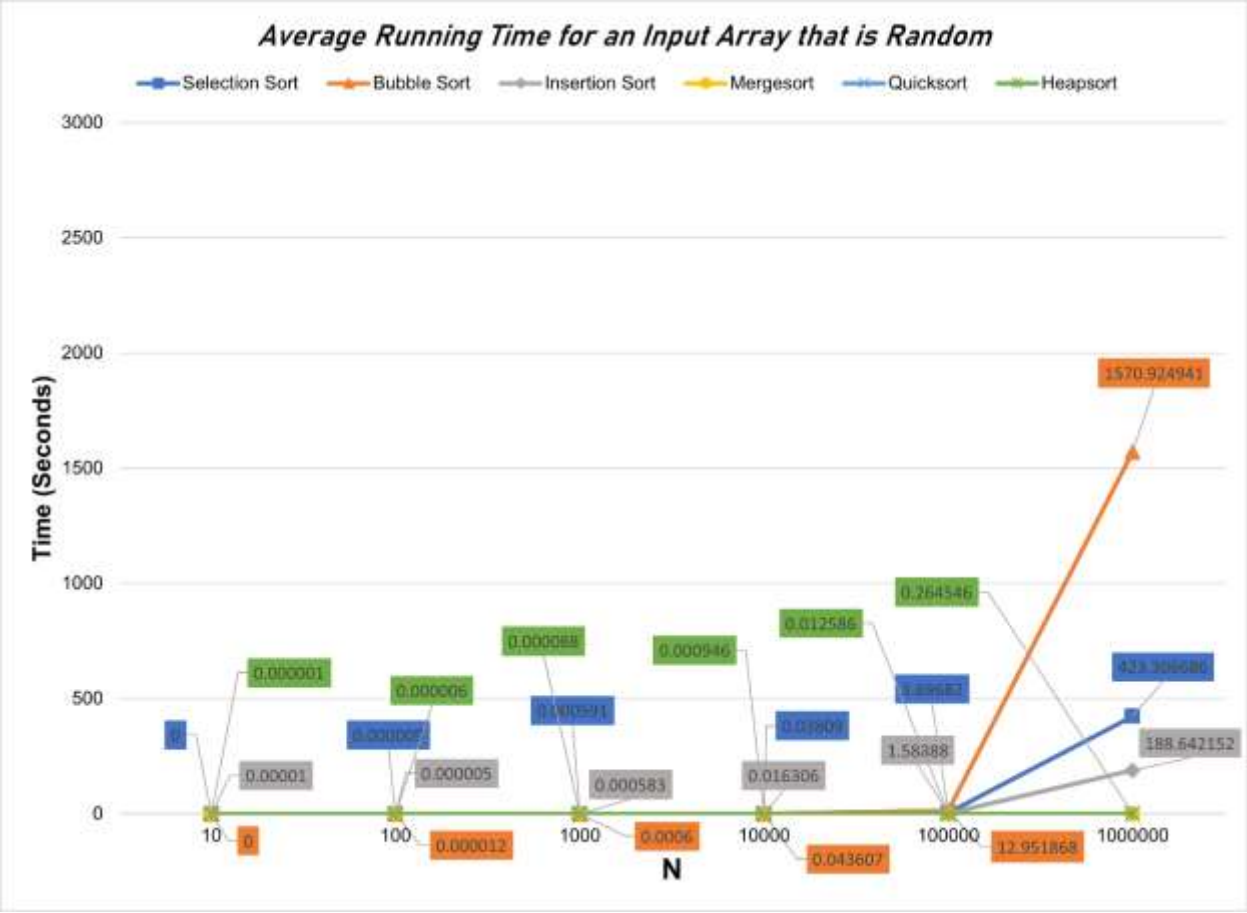
Analysis of the outputs as specified above. Discuss the results of your analysis and provide your conclusion.

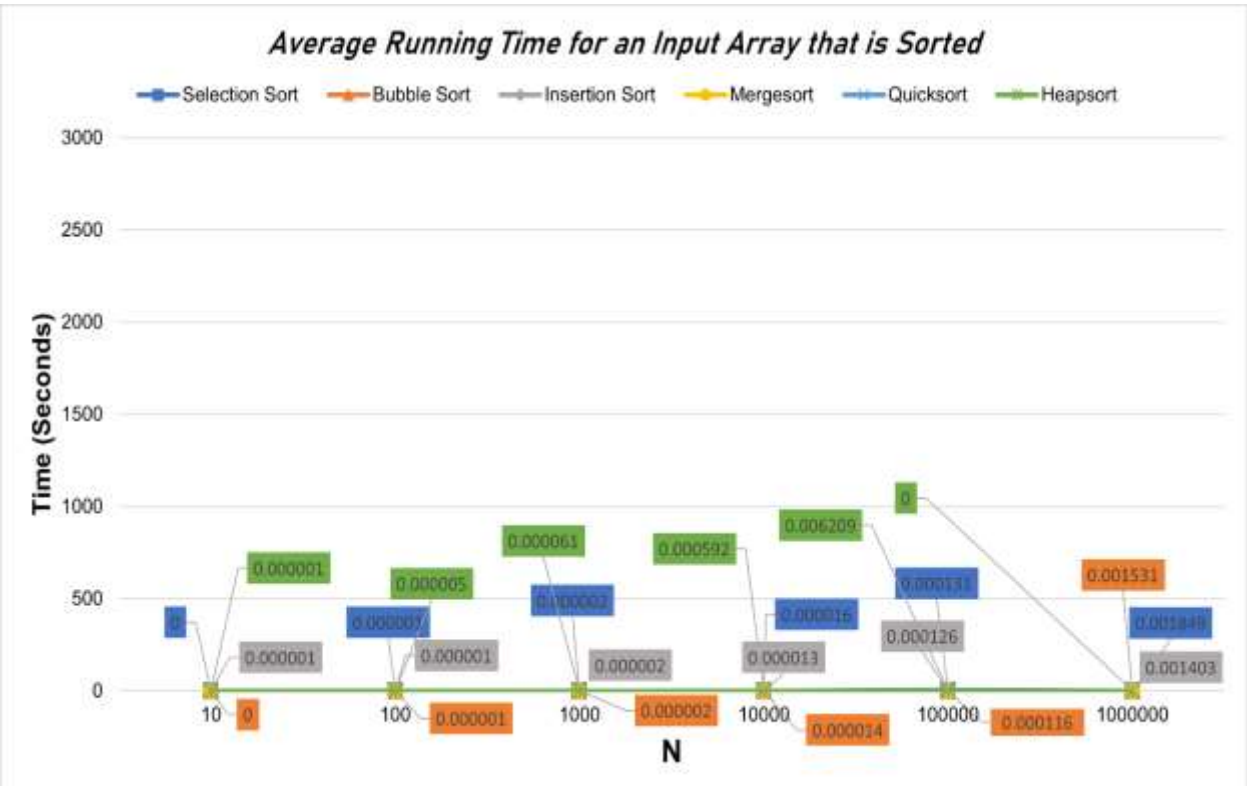
Average Running Time for an Input Array that is Random

N	Selection Sort	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Heap Sort
10	0.0000010000	0.0000010000	0.0000020000	0.0000030000	0.0000010000	0.0000010000
100	0.0000090000	0.0000120000	0.0000050000	0.0000130000	0.0000040000	0.0000060000
1000	0.0005910000	0.0005830000	0.0001880000	0.0001260000	0.0000420000	0.0000880000
10000	0.0380900000	0.0436070000	0.0163060000	0.0013510000	0.0005860000	0.0009460000
100000	3.6968200000	12.951868000	1.5888800000	0.0157940000	0.0074010000	0.0125860000
1000000	423.30668600	1570.9249410	188.64215200	0.2004340000	0.1062880000	0.2645460000

Average Running Time for an Input Array that is Sorted

N	Selection Sort	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort	Heap Sort
10	0.000000000	0.000000000	0.0000010000	0.0000010000	0.0000010000	0.0000010000
100	0.0000010000	0.0000010000	0.0000010000	0.0000090000	0.0000110000	0.0000050000
1000	0.0000020000	0.0000020000	0.0000020000	0.0000680000	0.0008470000	0.0000610000
10000	0.0000160000	0.0000140000	0.0000130000	0.0008690000	0.0476730000	0.0005920000
100000	0.0001310000	0.0001160000	0.0001260000	0.0081500000	4.9745740000	0.0062090000
1000000	0.0018490000	0.0015310000	0.0014030000	0.0884250000	0.0000000000	0.0000000000


Average Running Time for an Input Array that is Random
Selection Sort
Bubble Sort
Insertion Sort
Mergesort
Quicksort
Heapsort
Time (Seconds)
3000
2500
2000
1500
1000
500
0
1570.924941
0.264546
0.012586
0.000946
0.000688
0.000001
0.000006
0.00001
0.000005
0.000583
0.016306
1.58388
188.642152
10
100
1000
10000
100000
1000000
0
0.000012
0.0006
0.043607
12.951868
N


Average Running Time for an Input Array that is Sorted
Selection Sort
Bubble Sort
Insertion Sort
Mergesort
Quicksort
Heapsort
Time (Seconds)
3000
2500
2000
1500
1000
500
0
0.006209
0.000592
0.000061
0.000001
0.000005
0.001531
0.000126
0.001403
0.000002
0.000013
10
100
1000
10000
100000
1000000
0
0.000001
0.000002
0.000014
0.000116
N

IV. Challenges Encountered

- We had difficulties figuring out organizing the generated array into a neat and organized table.
- Upon running our program there were two sorting algorithms that gave us problems which are Quick sort and Heap sort. The array for heap and quicksort for randomly generated numbers works fine as it should, on the other hand, the results for the sorted numbers array don't show.
- Implementing of Array elements – we had difficulties figuring out how to increase the array size since our project requires a huge amount of data storage. We later found out the better implementation of an array is by using malloc() since malloc could store a high amount of data storage. By using it, we were able to store larger elements in the array and we were even able to run 1M input as a user.

V. Conclusion

Generally, considering the time taken and the large input data, it becomes clear that every sorting algorithm has its own efficiencies and deficiencies dealing with tremendous amounts of data. It is obviously proven on our table results that gave us insights to distinguish the capability of every algorithm. On the other hand, Quick Sort is the most sufficient and the fastest sorting algorithm to use when sorting randomized generated numbers.

VI. Contribution of Members

Aragorn Muncal

- Fixed challenges encountered in the program
- Coded
- Program Tester
- Helped with the code documentation

Erjay Cleofe

- Coded & Contributed Sorting algorithm
- Wrote the code documentation
- Program Tester
- Fixed challenges encountered in the program

Ignacio Tabug III

- Coded
- Program Tester
- Fixed challenges encountered in the program
- Wrote the code documentation

Ian Rhey Banares

- Fixed challenges encountered in the program
- Coded & add comments in the program
  Helped the code documentation (Graphs & Tables)
- Program Tester