

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

**ALGORITHMS AND DATA STRUCTURES**

Homework No. 3

Prepared by: ITfuc-21 stud.

UFUK SAYLAN

Accepted by: lect. Urtė Radvilaitė

Vilnius, 2022

**Task No. 13**

Algorithms:

* Quicksort. Pivot – the median of the first, middle and last element.
* Mergesort.

The goal –

The goal is to determine and analyze the factors influencing the sorting speed of algorithms.

Tests:

1. Data – randomly generated numbers. Analyze 5 different sizes of data. Generate 5 sets of random numbers with each size of data and perform 10 tests on each of them.
2. Data – numbers generated in descending order. Analyze 5 different sizes of data. Perform at least 10 tests with the same size of data.

**Testing environment.** *(0,2 point)*

Parameters of a computer:

Asus Tuf laptop with a Ryzen 7 CPU, NVIDIA GeForce GTX 1650 graphics card, 16 GB of Ram Memory.

Programming language: C

Data structure used to store data: ARRAY

Other: The operating system (OS) running on the computer can also influence the performance and behavior of a program. In this case, I am using Ubuntu as the OS.

**Study No. 1.**

**Evaluating the Performance of Quick Sort and Merge Sort on Random Data**

**Testing process.** *(0,2 point)*

1. The first size of data selected – 5,000. Using a C program, numbers were generated 5 sets of random numbers and stored in different text files. They were stored in a array.
2. Each set of random numbers was sorted in both sorting algorithms 10 times. Sorting times were saved in a table and the average of all sorting times was calculated.
3. Steps 1 and 2 were repeated with the following sizes of data: 5,000; 50,000; 500,000; 5,000,000; 50,000,000.
4. The overall test results were presented in a graph and the results were analyzed.

**Results**.

The results are presented in tables and graphs.

**The results presented in the table.** *(0,6 point)*

**Table 1**. Results when n=5,000;50,000;500,000;5,000,000;10,000,000

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Time of tests in Seconds | | | | | | | | | |  |
| **Data list 1** n:5,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | Avg. |
| Quicksort | 0.001475 | 0.001483 | 0.001372 | 0.001407 | 0.001499 | 0.000740 | 0.000740 | 0.000650 | 0.000613 | 0.000650 | **0.001074** |
| Mergesort | 0.001658 | 0.001060 | 0.000764 | 0.000709 | 0.000690 | 0.000682 | 0.000735 | 0.000694 | 0.000643 | 0.000735 | **0.000837** |
| **Data list 2**  n:50,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.007465 | 0.008161 | 0.007535 | 0.007884 | 0.007477 | 0.007874 | 0.007517 | 0.007902 | 0.008244 | 0.007475 | **0.007753** |
| Mergesort | 0.012735 | 0.008137 | 0.009072 | 0.009279 | 0.008969 | 0.009141 | 0.008948 | 0.008604 | 0.008053 | 0.008626 | **0.009156** |
| **Data list 3**  n:500,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.081283 | 0.081850 | 0.092144 | 0.099762 | 0.081821 | 0.088144 | 0.084925 | 0.089463 | 0.085043 | 0.079023 | **0.086346** |
| Mergesort | 0.115971 | 0.110757 | 0.095270 | 0.116394 | 0.119320 | 0.096058 | 0.095905 | 0.095516 | 0.108807 | 0.096898 | **0.105090** |
| **Data list 4**  n:750,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.120674 | 0.130712 | 0.147754 | 0.135213 | 0.123653 | 0.148836 | 0.149245 | 0.122883 | 0.148304 | 0.125301 | **0.135257** |
| Mergesort | 0.146477 | 0.178179 | 0.152651 | 0.152377 | 0.147918 | 0.147145 | 0.148102 | 0.147347 | 0.154569 | 0.155116 | **0.152988** |
| **Data list 5**  n:1,000,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.164418 | 0.165159 | 0.164071 | 0.177865 | 0.191831 | 0.183071 | 0.166402 | 0.204378 | 0.168716 | 0.175110 | **0.176102** |
| Mergesort | 0.246169 | 0.245776 | 0.214918 | 0.196808 | 0.239519 | 0.202795 | 0.197091 | 0.202148 | 0.198354 | 0.196184 | **0.213976** |

**Table 2**. General results.

**The results presented in the graph.** *(0,6 point)*

**Figure 1**. Random number sorting results.

**Analysis of results.**

**Statement 1 – theoretical evaluation.** *(0,2 point)*

The obtained testing results of Quicksort corresponds to theoretical evaluation.

The obtained testing results of Mergesort corresponds to theoretical evaluation.

**Justification. *(0,2 point –* *derivation of increase from theoretical estimate, 0,3 point – calculations, 0,3 point – analysis)***

Quicksort has a worst-case time complexity of O(n^2), although it has an average-case time complexity of O(n log n). This means that in the worst case, the algorithm will take O(n^2) time to sort an array of size n. However, the average-case time complexity is O(n log n), which means that on average, the algorithm will take O(n log n) time to sort an array of size n.

The QuickSort algorithm took 0.001074 seconds to sort an array of 5000 elements, according to our calculations. If we would considered the wors case, it should have taken 0.001074 \* n^2 to complete other datas. We see it is not the case here. For comparison, it took 0.135257 seconds to sort an array of 750,000 elements, which is much closer to the average-case performance of QuickSort than the worst-case performance. If it’d been worst case scenoria, it would take 0.001074 \* 750,000^2 seconds, which is a lot larger than what we see.

<provide calculations>   
<Here is an example when the theoretical estimate of the first algorithm is O(n2). You need to look at the theoretical estimates of both of your algorithms and perform analogous calculations >

Theoretical estimate of an <algorithm> when data is random: O(n2).

The size of data was increased twice each time, therefore:

Conclusion: Doubling the size of data should increase the sorting time by a factor of about 4.

Calculations with testing results:

.........................................

.........................................

<Provide calculations for all sizes and examine whether the time always changed ~4 times. Is 3.618 close enough to 4 to be said to be within the theoretical estimate? If not, why?>

<Then you provide the calculations for the second algorithm>.

**Statement 2 – speed comparison.** *(0,2 point)*

Results showed that <algorithms> is faster than <algorithm> ..........................................< describe when faster: always, only with specific size and so on>

Justification. *(0,3 point)*

**......................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................** <based on theory (how algorithms work) explain why one algorithm is faster than another>

**Statement 3 –** <Think of it yourself. Based on the properties of algorithms, trends noticed >**.** *(0,2 point)*

..............................................................................................................................................................................................................................................................................................................................................................................

Justification. *(0,3 point)*

<base your statement on the operation of algorithms and theory >

.................................................................................................................................................................................................................................................................................................................................................................................................................................................

**Study No. 2**

**Efficiency Comparison of Merge Sort and Quick Sort on Descending Order Data**

**Testing process.** *(0,2 point)*<describe step by step what to do in the testing>

1. .......................................................................................
2. .......................................................................................
3. .......................................................................................
4. .......................................................................................

**Results**.

The results are presented in tables and graphs.

**The results presented in the table.** *(0,6 point)*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Time of tests of descending data sets in Seconds | | | | | | | | | |  |
| **Data list 1** n:5,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | Avg. |
| Quicksort | 0.000862 | 0.000733 | 0.000738 | 0.000727 | 0.000732 | 0.000735 | 0.000738 | 0.000469 | 0.000400 | 0.000366 | **0.000650** |
| Mergesort | 0.000404 | 0.000397 | 0.000444 | 0.000496 | 0.000484 | 0.000509 | 0.000556 | 0.000568 | 0.000559 | 0.000555 | **0.000497** |
| **Data list 2**  n:50,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.003444 | 0.004412 | 0.003861 | 0.004389 | 0.004661 | 0.005083 | 0.005027 | 0.004683 | 0.004822 | 0.003277 | **0.004366** |
| Mergesort | 0.013563 | 0.005759 | 0.005816 | 0.005883 | 0.006265 | 0.005743 | 0.005766 | 0.005707 | 0.005743 | 0.005095 | **0.006534** |
| **Data list 3**  n:500,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.034779 | 0.035933 | 0.035933 | 0.034948 | 0.040549 | 0.042442 | 0.044040 | 0.037059 | 0.035094 | 0.034817 | **0.037559** |
| Mergesort | 0.058178 | 0.057837 | 0.060389 | 0.063925 | 0.067406 | 0.071071 | 0.070329 | 0.070420 | 0.059938 | 0.067990 | **0.064748** |
| **Data list 4**  n:750,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.052983 | 0.052585 | 0.054253 | 0.054960 | 0.064910 | 0.067057 | 0.059224 | 0.056095 | 0.057541 | 0.054103 | **0.057371** |
| Mergesort | 0.090180 | 0.088850 | 0.088486 | 0.089443 | 0.088879 | 0.088255 | 0.091546 | 0.088968 | 0.106205 | 0.112697 | **0.093351** |
| **Data list 5**  n:1,000,000 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |  |
| Quicksort | 0.074144 | 0.089151 | 0.088498 | 0.092786 | 0.070539 | 0.072345 | 0.070858 | 0.070879 | 0.070934 | 0.071320 | **0.077145** |
| Mergesort | 0.117929 | 0.146225 | 0.145398 | 0.144719 | 0.133487 | 0.127474 | 0.146810 | 0.144861 | 0.131968 | 0.119075 | **0.135795** |

**The results presented in the graph.** *(0,6 point)*

***Figure 1****. Descending number sorting results.*

**Analysis of results.**

**Statement 1 – theoretical evaluation.** *(0,2 point)*

The obtained testing results of <1 algorithm> corresponds to/doesn’t match theoretical evaluation.

The obtained testing results of <2 algorithm> corresponds to/doesn’t match theoretical evaluation.

Justification. *(0,2 point –* *derivation of increase from theoretical estimate, 0,3 point – calculations, 0,3 point – analysis)*

<provide calculations and analysis>   
<If the theoretical estimate is the same as for the first test, then proceed to the calculations of how time changes.>

**Statement 2 – speed comparison.** *(0,2 point)*

....................................................................................................................................................................................................................................................................................................................................................

Justification. *(0,3 point)*

...................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................... <based on theory (how algorithms work) explain why one algorithm is faster than another>

**Statement 3 –algorithm speed dependence on data set.** *(0,2 point)*

**Quicksort sorts** the data in study of *Evaluating the Performance of Quick Sort and Merge Sort on Random Data* ***slower*** than *the data in the study of Efficiency Comparison of Merge Sort and Quick Sort on Descending Order Data*

**Merge sorts** the data in study of *Evaluating the Performance of Quick Sort and Merge Sort on Random Data* ***slower*** than *the data in the study of Efficiency Comparison of Merge Sort and Quick Sort on Descending Order Data*

Justification. *(0,3 point)*

Merge sort is a divide and conquer algorithm that works by dividing the input array into smaller subarrays, sorting those subarrays, and then merging them back together. The merge step of the algorithm is typically the most efficient part, because it involves combining two sorted arrays into a single sorted array in a single pass through the data.

In the case of an array that is already in descending order, the subarrays produced by the divide step will already be partially sorted, which means that the merge step can more efficiently combine them into a single sorted array. This will lead to faster overall runtime for merge sort on such arrays.

Quick sort does not generally perform better on arrays that are already partially sorted or in descending order. In fact, quick sort tends to perform worse on such arrays because the pivot selection process is less effective at partitioning the array into two balanced subarrays. However, our choice of pivot element was median. That’s why it is more likely to partition the array into two subarrays of approximately equal size.

**General conclusions.**

1. conclusion. *(0,3 point)*

...........................................................................................................................................................

1. conclusion. *(0,3 point)*

...........................................................................................................................................................

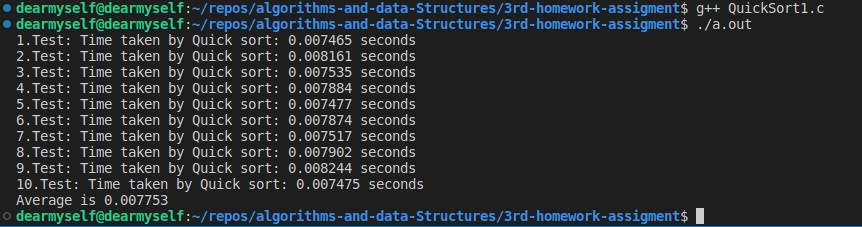
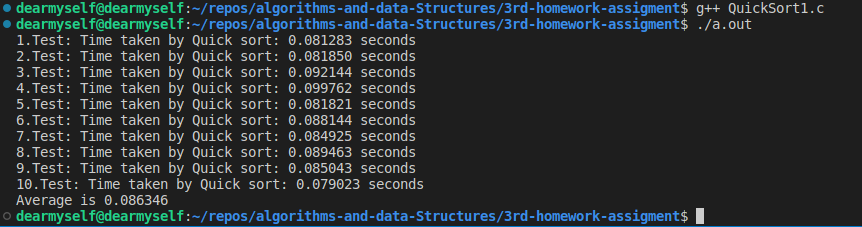
1. conclusion. *(0,3 point)*

...........................................................................................................................................................

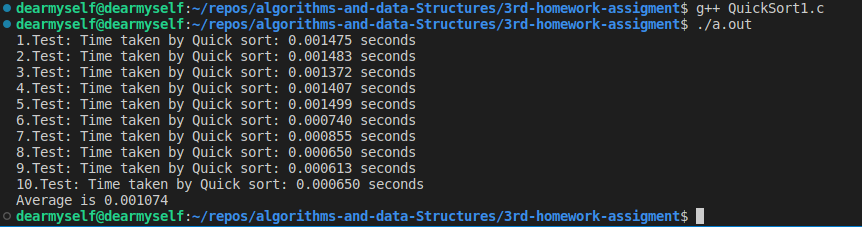
<provide GENERALIZED conclusions of both studies>

**ANNEXES.**

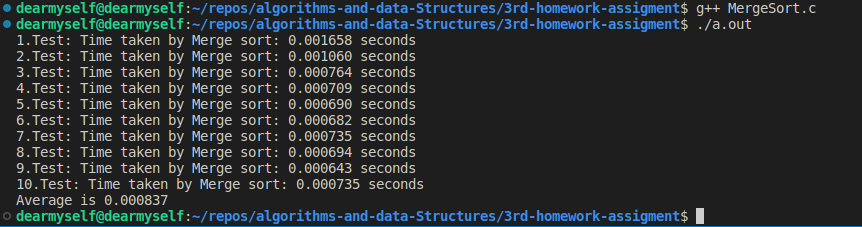
Quicksort with 500,000 random numbers data:

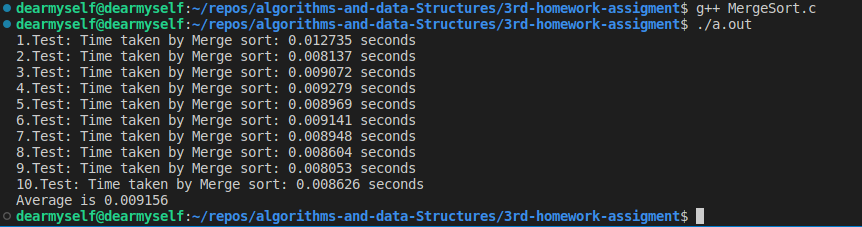
Quicksort with 50,000 random numbers data:

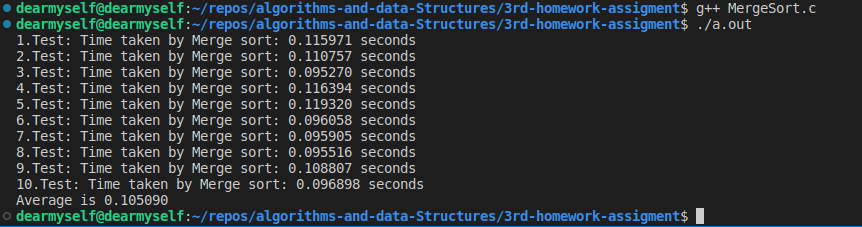
Quicksort with 5,000 random numbers data:

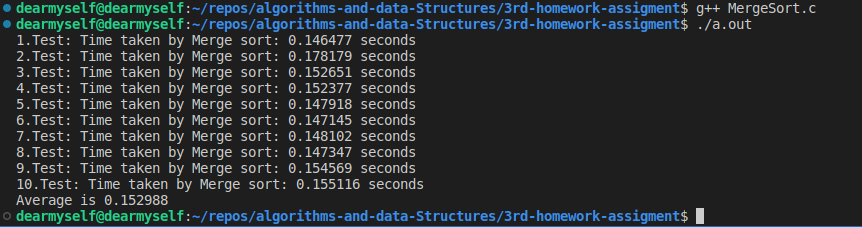


Mergesort with 5,000 random numbers data:

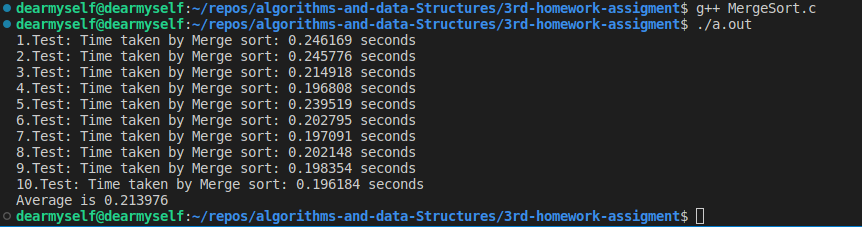
Mergesort with 50,000 random numbers data:

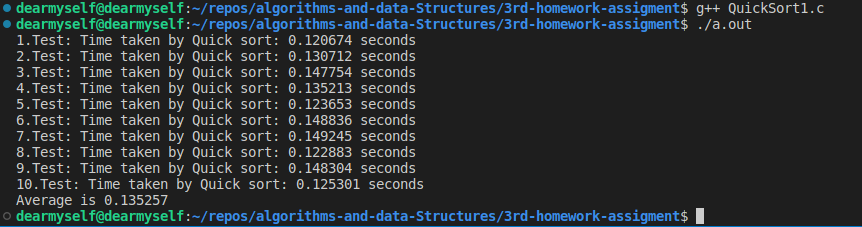
Mergesort with 500,000 random numbers data:

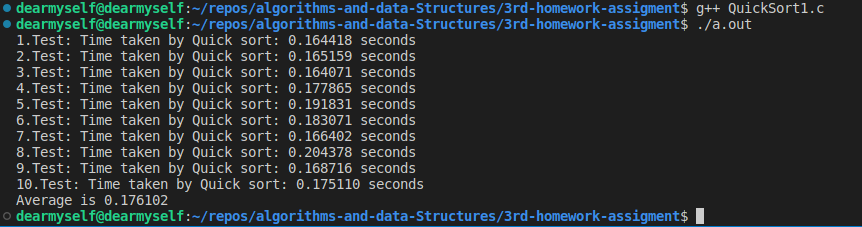
Mergesort with 750,000 random numbers data:

a

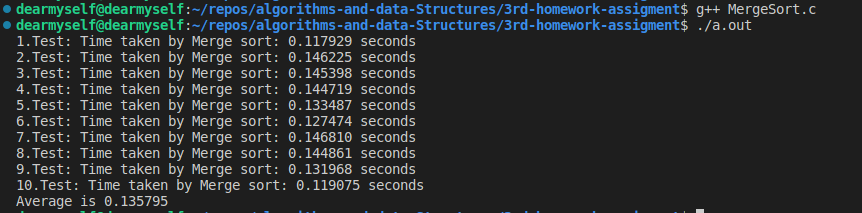
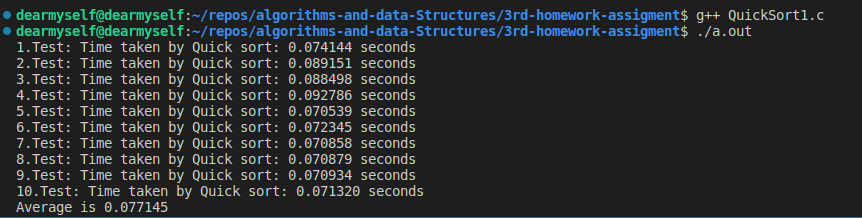
Mergesort with 1,000,000 random numbers data:

Quicksort with 750,000 random numbers data:

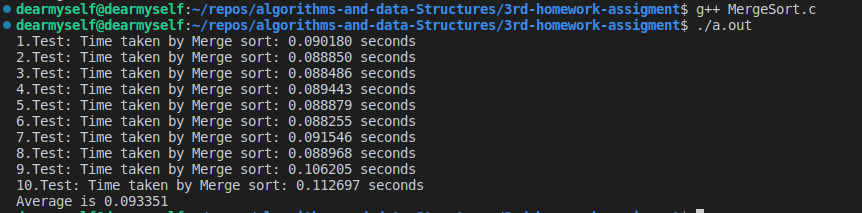
Quicksort with 1,000,000 random numbers data:

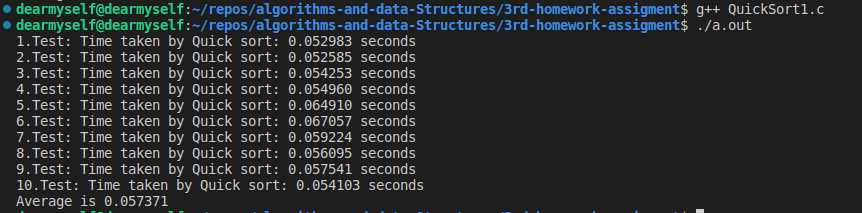


Quicksort with 1,000,000 descending numbers data:

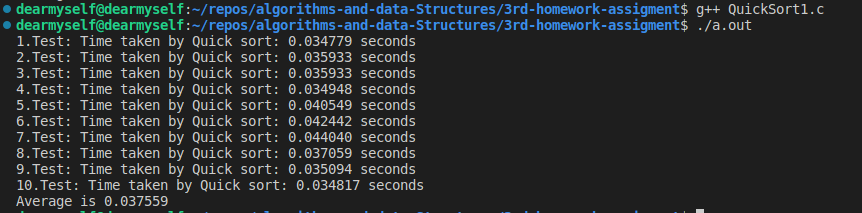
Mergesort with 1,000,000 descending numbers data:

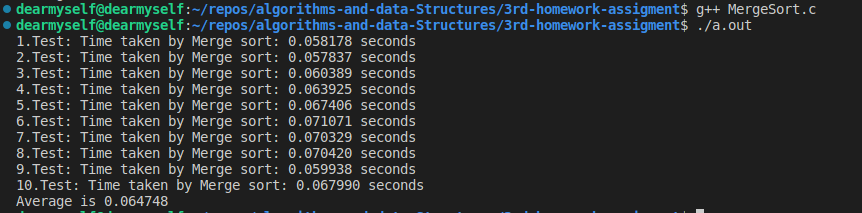
Mergesort with 750,000 descending numbers data:

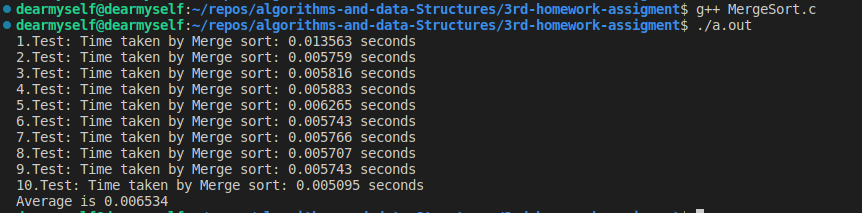
Quicksort with 750,000 descending numbers data:



Quicksort with 500,000 descending numbers data:

Mergesort with 500,000 descending numbers data:

Mergesort with 50,000 descending numbers data:

Quicksort with 5,000 descending numbers data:

