A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms using Three Programming Languages II: Energy Consumption Analysis

Oluwakemi Sade Ayodele
Department of Computer Science, Kogi State Polytechnic,
Lokoja, Nigeria
Email: Kemtemmy2009@gmail.com
&
Bamidele Oluwade
Dewade Systems Consult, Nigeria
Email: deleoluwade@dewadeconsult.com

ABSTRACT

Energy conservation is a major concern today prompting individuals, companies, and organizations to seek for energy-efficient products and services as the energy cost to run equipment has grown to be a major factor. Therefore, software engineers, hardware engineers and system designers explore new directions to reduce the energy consumption of their products and services. The aim of this study is therefore to carry out a comparative experimental analysis of the energy consumption of Quick, Merge and Insertion sort algorithms using three programming languages. The three sorting algorithms were implemented in three programming languages (C, Java and Python) and two algorithm implementation styles (Iterative and Recursive). Time stamp was used to capture the execution time of the sorting algorithm. Power consumed was measured using Joule meter. From the experiments, it was observed, among others, that the data size, programming language used and algorithm implementation styles affect the total energy consumption. It was also observed that the energy consumption is dependent greatly on the time performance of the Algorithm. Using scripting language has clear drawbacks in terms of energy efficiency which should be taken into consideration when using them. Therefore, this paper provides information for choice of sorting algorithm type and its algorithm implementation style in order to minimize energy consumption, most importantly in the current period of explosive growth in the use of resource-constrained devices that run majorly on battey life. This gives developers knowledge on energy efficiency in software leading to choosing some code over others based on their energy performance. The paper is a follow-up to the analysis carried out on execution time in a preceding paper.

Keywords: *Energy Efficiency, Sorting, Comparative Analysis*

Reference Format:

Ayodele, Oluwakemi Sade and Oluwade, Bamidele (2019), A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms Using Three Programming Languages II: Energy Consumption Analysis, *Afr. J. MIS*, Vol.1, Issue 2, pp. 44 - 63.

1. INTRODUCTION

Energy conservation is a major concern today with so many programs and bodies providing incentives to save energy and promote the use of renewable energy resources. As the energy cost for running equipment has grown to be a major factor, individuals, companies, and organizations seek energy-efficient products and services.

This Energy consumption is also critical in computer devices, in terms of both cost and availability. Google engineers, maintaining thousands of servers, warned that if power consumption continues to grow, power costs can easily overtake hardware costs by a large margin (Barroso, 2005).

Power dissipation is also a major concern in portable, battery-driven devices that have proliferated rapidly in recent years with battery of laptop or mobile phone easily depleted. Also, in the case of distributed devices like sensor networks where the charging of batteries is difficult or impossible, energy issue is even more serious and noticeable. Furthermore, energy dissipation causes thermal problems. Most of the energy consumed by a system is converted into heat, resulting in wear and reduced reliability of hardware components.

For these reasons, energy has become a leading design constraint for computing devices. Therefore, the goal of software engineers is to design energy-efficient algorithms that reduce energy consumption while minimizing compromise to service.

Therefore understanding energy usage/ consumption of an algorithm has become a major issue in determining and improving the energy efficiency of any algorithm.

More so, estimating energy consumption is a laborious task. High energy consumption affects the environment which necessitate for greenness in software development.

Energy conservation which is one of the areas of Green computing is now a major concern in today's world with its conservation of uppermost importance due to its high cost and limited availability. The exponential growth in computing activity and the ever increasing energy consumption of data centers which doubles between year 2000 and 2006 (http://www1-eere.energy.gov/Femp/Program/dc_energy_consumption.html) with adverse effect of energy emission on the environment and rising concern for energy conservation which has made energy efficiency of algorithm of uppermost importance. It has been found that ICT is responsible for 2% of total world emissions (www-03.ibm.com/press/attachments/GreenIT-final-mar4.pdf). Therefore, protecting the environment by saving energy and thus reducing cabondioxide emissions is one of today's hottest and most challenging topics (Bunse et. al., 2009A). From a technological point of view, the realization especially for mobile systems still falls behind expectations.

In all of these, energy efficiency studies have mostly focused on the hardware side of energy efficiency solution, the software role still requires deepening in terms of methods and techniques. Algorithmic techniques can provide effective solutions for energy savings, complementing hardware and system-based approaches.

This paper is therefore in that direction, studying and comparing the energy efficiency of sorting algorithms based on Programming language, Algorithm implementation styles and Data size.

This work is a continuation of an earlier paper which focused on executime time analysis (Ayodele and Oluwade, 2019).

2. RELATED WORK

Deepthi *et al.*, *(*2018) conducted experiments to study how different sorting algorithms have an impact on energy consumption using C language implementation. It was discovered that both time and energy have an impact on the efficiency of these sorting algorithms with quick sort, merge sort and shell sort found to be in the same range of time and energy consumption, followed by insertion and selection sort which is far better than Bubble sort. However, the implementation is only the C language with a non-varying small data size of 10,000. The effect of sorting large data sizes was not considered.

Rivoire *et al*., (2007) proposed an external sorting benchmark for evaluating the energy efficiency of

sorting for a wide range of system but focused more on hardware rather than the software domain.

Bunse *et al.*, (2009), in their study focused on sorting algorithms, which are not only used directly by the user of a device but also very often implicitly by other algorithms and show that different sorting algorithms have different energy consumptions however this work is limited to just comparison with Sorting algorithm and programming language.

Aaron *et al.*, (2010) performed a detailed analysis of energy consumption of a smart phone, based on measurements of a physical device. How the different components of the device contribute to overall power consumption was analysed. A model of the energy consumption for different usage scenarios was developed, and showed how these translate into overall energy consumption and battery life under a number of usage patterns. The measurement of energy consumption was on smart phone as hardware but not on the installed software.

Ahmad Elkahlout *et al*., (2017) conducted a comparative study to evaluate the performance of three algorithms; comb, cocktail and count sorting algorithms in terms of execution time. Java programming was used to implement the algorithms using numeric data on the same platform conditions. Among the three algorithms, it was found out that the cocktail algorithm has the shortest execution time; while counting sort comes in the second order. Furthermore, Comb comes in the last order in term of execution time. In the work, different sorting algorithms were used, only Java was used for comparison and energy consumptions of the sorting algorithms were not determined.

Aremu *et al*., (2016) presented a comparative study of three sorting algorithm median, heap, and quick sort using CPU time and memory space as performance index to adopt the most efficient sorting technique in the development of job scheduler for grid computing community. These were implemented with C-language; while the profile of each technique was obtained with G-profiler. However, energy efficiency of these sorting algorithms was not considered.

Ali *et al*., [15] studied the working of five sorting algorithms (bubble, selection, insertion, merge, quick sorts) and compared them on the basis of computational complexities, usage of memory and other computer resources, stability, number of swaps and number of comparisons. Quick sort was said to be the faster sorting algorithm while bubble sort is the slowest algorithm. Nothing was discussed on energy efficiency as parameter for comparing these sorting algorithms.

3. METHODOLOGY

3.1 Research Questions (RQ)

The following research questions have been formulated for this study to help the stated objectives above:

RQ1: Does different dataset size consume different amount of energy for executing the same algorithm?
RQ2: Does different programming languages consume different amount of energy for executing the same algorithm?
RQ3: Does different Algorithm implementation style (Iterative and Recursive) have effect on energy consumption?
RQ4: Is there a relationship between Energy Usage, DataSize and Programming language used?

3.2 Implementation/Experimentations

3.2.1 Algorithm Implementation Languages

The three sorting algorithms (Quick, Merge and Insertion Sorts) were run using three different programming languages (Java, C and Python Languages). This is to:

i. Determine how energy is consumed when changing the implementation language.
ii. Give a clue as to why energy consumption varies between languages.

The sorting algorithms used in the study, Merge, Quick and Insertion sort algorithms were implemented using Java, C and Python Programming Languages. Merge and Quick sort were implemented using both recursive and iterative programming styles while Insertion was implemented using only iterative programming structure. The choices of programming languages were categorized based on:

- Virtual machine based languages: Java (compiled and interpreted language)

- Native Languages: C(Compiled language)
- Scripting Languages: Python (Interpreted language)

The Experiments were conducted on the same computer configuration, and running Window 7. The sorting algorithms were run on a laptop with the following configuration, HP 630 Notebook PC, 4GB RAM, Intel(R) Core(TM) i3 @2.53 processor.

3.2.2 Algorithm implementation styles

Two different algorithm implementation styles were used on the same Algorithm for the three Sorting Algorithms (Quick, Merge and Insertion) to compare its energy consumption:

i. Iterative Algorithm
ii. Recursive Algorithm

3.3 Data Size

Data sizes ranges from one hundred thousand (100,000) to five hundred thousand (500,000) at an interval of one hundred thousand. The data was randomly generated by importing random function from the language(s) libraries. The power consumption and energy efficiency of random generator are measured. This is to ensure that the power consumption and energy efficiency captured is solely for the algorithm.

To reduce measurement error, each data size being considered was executed five times, the average captured and recorded for use.

3.4 Measurement

TimeStamp and Joulemeter

Time stamp was placed directly above the called function containing the sorting algorithm and another time stamp was placed directly below the called sorting function. This is to ensure that the execution time captured is solely for the sorting algorithm. Execution time was derived by subtracting the start time from end time. Power consumed was measured using Joule meter.

Execution Time

The Execution Time T is the time that it takes for an algorithm to execute.

Execution time for sorting was measured using system clock imported from programming language's libraries.

System.cuurentTimeMillis(), time(Null) and datatime, timeit.default_timer() were time functions used for Java, C and Python Programming Languages respectively. Execution Time was derived by subtracting Start time from end time. Measured times were in seconds.

3.5 Software Setup

Merge sort, Insertion sort and Quick sort algorithms were implemented using C, Java and Python. Dev C++ and Net beans Integrated Development Environments were used for C and Java programming languages respectively. Python 3.6.0 was used in the implementation. The sorting algorithms were implemented using recursive and iterative styles. Each of the algorithms was implemented in three programming languages, Merge sort and Quick sort algorithms were implemented using the iterative and recursive version for each programming languages. Data sizes ranges from one hundred thousand (100,000) to five hundred thousand (500,000) at an interval of one hundred thousand. The integers for each data size were randomly generated using the rand() function. The sorting algorithms were placed in functions and classes and called in the main function and class. Time stamp was placed directly above the called function containing the sorting algorithm and another time stamp was placed directly below the called sorting function. This is to ensure that the execution time captured is solely for the sorting algorithm. Execution time was derived by subtracting the start time from end time. Power consumed was measured using Joule meter.

To carry out the experiment, launch Joule meter and the IDE (Dev C++ or Net beans). Run the program, a request to input data size was displayed. Click on browse on Joule meter and specify file name to store power consumed by sorting algorithm per millisecond and recorded. Click on the start button on the Joule meter to record power measurement. Immediately, input data size and click ok to start sorting. The power measurement was stop immediately sorting is completed. The execution time was displayed in millisecond. The start time and end time was used to trace power consumption for the sorting. The experiment is repeated five times for each algorithm using one data size and one programming structure (iterative or recursive). The average of the five experiments was recorded as the value for the data size.

Algorithm Energy

$$E = P\ X\ T \quad \ldots(1)$$

where P = CPU power utilization and T = the execution time

Microsoft Excel was used to compute the Energy from the formula above. (See Table 3.2 as screen shot)

4. RESULTS

Below are the lists of tables developed from the results of the experiments. The results are on:

i. Table 4.1: Energy Comparison of Iterative sorting Algorithms Implementations in C
ii. Table 4.2: Energy Comparison of Iterative sorting Algorithms Implementations in Java
iii. Table 4.3: Energy Comparison of Recursive sorting Algorithms Implementations in Python
iv. Table 4.4: Energy Comparison Of Recursive Quick Algorithms Implementations in C++, Java And Python
v. Table 4.5: Energy Comparison Of Recursive Merge Algorithms Implementations In C++, Java And Python
vi. Table 4.6: Energy Comparison of Iterative Insertion Algorithms Implementations In C++, Java and Python
vii. Table 4.7: Energy Comparison Of Iterative Merge Algorithms Implementations In C++, Java and Python
viii. Table 4.8: Energy Comparison Of Iterative Quick sort Algorithms Implementations In C++, Java and Python
ix. Table 4.9: Power (Watt) of three Iterative and Recursive sorting Algorithms implemented in Java
x. Table 4.10: Power(Watt) of three Iterative and Recursive sorting Algorithms implemented in C
xi. Table 4.11: Power (Watt) of three Iterative and Recursive sorting Algorithms implemented in Python

5. DISCUSSION OF RESULTS

All corresponding figures are in the Appendix

5.1 Impact of Data Size on Energy Consumption (Energy versus Data Size)

From figure 1, which is the figure of Data Size Analysis of the three Sorting Algorithms implemented in two algorithm implementation styles?
As the data size increases from 100,000 to 500,000 at a range of 100,000, the energy also increases.

Checking the trend in the shape of the different graphs from each of these graph lines, the higher the data size the higher the energy consumption. From this figure, there is continuous upward flow of the graph lines in all cases.
Therefore the following conclusions were made:

1. There is an increase in Energy usage as data size increases in the three programming languages used
2. Also there is an increase in energy consumption in the two algorithm implementation styles used as the data size increases.
3. Generally, from these observations, the table and figures indicate decrease in energy efficiency (increased energy consumption) with increase in data size.
4. This implies that both data size and energy have strong impact on the efficiency of the algorithm.

5.2 Impact of Data Size on Execution Time (Execution Time versus Data Size)

From the previous work of same authors (Ayodele and Oluwade, 2019), the following conclusions were drawn:

1. The Average Execution Time increases with increase in data size
2. Irrespective of the Algorithm implementation style used, statement in (1) above is upheld.
3. Irrespective of the Programming language used, statement in (1) above is upheld.

4. Average Execution time of recursive implementations of all the sorting Algorithms and in all programming languages used is higher than its equivalent iterative implementations.
5. From figure 4.10C – 4.10E, both the execution time and energy have an impact on the efficiency of the algorithms.

5.3 Impact of Data Size on Power Usage (Power versus Data Size)

From Figures 3 – 6,

Observations:

The Power consumption over various levels of data sizes was plotted. The power consumption of four different sorting algorithms, their relative implementation in three distinct languages and two algorithm implementation styles were analysed.

The power consumption for iterative merge and quick sort is relatively stable with increasing data size. However, insertion sort consumes high quantity of power with increase in data size. From the graphs, insertion sort is not suitable for large data size. The KSU hybrid sort has the lowest power consumption.

Power consumed by Merge sort and Quick sort implemented in C fluctuates with increasing data size while insertion sort is relatively stable with increasing data size though with high power consumption. From the graph, Quick sort is more power friendly.

Power consumed by iterative insertion, merge and quick sorts implemented in python is high compared to other programming languages. Merge sort consumed less power compared to quick and insertion sort. Insertion sort had the highest power consumption.

Power consumption for recursive merge and quick sorts implemented in Java, recursive quick sort consumed lesser power compared to recursive merge sort. Hence, recursive merge sort is more power efficient.

Power consumption for recursive merge and quick sort implemented in C, quick sort is unstable with smaller data size but consumed lesser power compared with merge sort.

Recursive merge sort implemented in python consumed very high power compared to recursive quick. The result shows that recursive merge implemented in python is not suitable for platforms where energy is highly prioritized.

Insertion sort implemented in Java consumed a noticeable high power compared to iterative merge sort, recursive merge, iterative quick and recursive quick sorts. Insertion sort implemented with java is not suitable for power critical devices.

Data size, algorithm, programming language and algorithm implementation style all affect the power consumption of an Algorithm.

In all programming languages used and implementation styles, KSU hybrid sort has the lowest power consumption.

5.4 Impact of Programming Languages on energy consumption

Observations:

From Figure 1-figure 10, Checking the trend in the shape of the graph, from figure 10 which has the plotting of the three sorting algorithms implemented in the all the programming languages used and also the two algorithm implementation styles. In all cases the energy consumption also increases with increase in data size.

The following were observed:

1. The Energy consumed by same algorithm varies from one programming language to the other.
2. As data size increases, the energy consumption increasing in the three programming languages.
3. The Energy consumed by the scripting language (Python) is significantly higher than the virtual based language and also to the native code language.

4. It was noted the high energy consumption of the scripting language which can be attributed to the fact that there is the need to interpret then execute the algorithm. This additional step clearly has a higher value in terms of energy consumption.
5. Programming languages are diverse in terms of design and specifications. The experiment on programming languages with this diversity gives us hints as to why energy consumption varies from one programming language to another.

5.5 Impact of Algorithm implementation Style (Iterative and Recursive)

Observations:

From figure 4 the Insertion sorting algorithm using the Iterative algorithm implementation style implemented in python programming language has the highest energy consumption. As the data size increases, the energy consumption also increases. It should be noted as earlier on stated that insertion sort algorithm was only implemented using the iterative algorithm implantation style. In all other sorting algorithms used implemented in the three programming languages, the energy consumption of the recursive version is higher than the iterative.
Checking the trend in the shape of the graph,

The following were observed:

1. The algorithm under review determines the energy efficiency of a particular algorithm implementation style.
2. Generally, iterative implementation of sorting algorithms are more energy efficient than equivalent recursive implementation.
3. The recursive insertion algorithm implemented in Python has the highest energy consumption. This implies that this sorting algorithm implemented in this language is not energy efficient.

5.6 Relationship between Energy Usage, Data Size, Algorithm Implementation Style and Programming language used.

It is worth noting that Sorting Algorithms have looping structures. If the loops iterate n times, then the time for all n iterations is T1. N where T1 indicates number of times the computation takes place in one loop iteration. The value of T1 is dependent on programming language used, compiler or interpreter used in translating source code to machine code, programming structure, algorithm implementation style and other factors. While the T1 has negligible effect on computing the asymptotic complexity, the impact is evident in actual implementation of the algorithms.

The impact of T1 on the execution time for each algorithm implemented accounts for differences in behavior. Same algorithm implemented with different programming languages and algorithm implementation styles have slight differences in execution time. Obviously, T1 is not negligible but key factors to be considered in algorithm implementation.

However, it is interesting to see the differences in execution time between iterative and recursive variants of the same algorithms. These lead to the assumption that the use of recursion might improve performance, but will definitively increase energy consumption. An Iterative algorithm will be faster than the Recursive algorithm because of overheads like calling functions and registering stacks repeatedly. Many times the recursive algorithms are not efficient as they take more space and time. The results show that the iterative version of the Sorting Algorithms used are more energy efficient than the recursive version. In conclusion we can say that energy consumption is strongly related to execution time.

From all the Figures presented it can be seen that Data size, Algorithm implementation Style and programming language are factors that impact the energy consumption of Algorithm. It can be noted that the same sorting algorithm has different energy consumption with varying data sizes, when implemented in different programming language or/and when implemented in different algorithm implementation styles. The higher the data size, the higher the power consumption, the higher the time taken for sorting and invariably the higher the energy consumption. Also, how software is written and the programming language used are important factor for energy efficiency.

Varying the parameters values (Data size, algorithm implementation style, programming language) impacts the energy consumption with different evolution patterns based on which parameters are varied.

6. CONCLUSION

Energy management has become an important issue in computing environments. Therefore, measuring the energy consumption of software is the first step in order to produce an energy efficient code to complement other hardware and system-based approaches.

Developers over the years made significant effort to optimize hardware component in pursuant of an energy efficient devices treating the algorithm as a black box.

The energy, time execution and power consumption of different sorting algorithms were analysed, the relative implementation in three distinct languages, and two algorithm implementation styles over an average of five data sizes.

The Data size, Programming language, the implementation algorithm style are factors that impact the energy consumption of software. Varying the parameters values (Data size, Programming language, Algorithm implementation style) impacts the Energy consumption with different evolution patterns based on which parameters are varied.

Our work therefore provides the basic information to choose a specific sorting implementation to minimize energy consumption depending on the usage of the sorting algorithm and type of computing devices in which the sorting algorithm will be used most especially in any handheld battery driven devices.

REFERENCES

[1] Barroso, L.A (2005). The price of performance. *ACM Queue 3*. [Online] http://www.techopedia.com>definition Retrieved on 13/01/2018

[2] Call for paper for Green Computing, In Software Engineering Aspects of Green Computing In the *28th Annual ACM Symposium on Applied Computing*, March 18-22, 2013, Coimbra, Portugal. [online]. Available: http://trese.ewi.utwente.ni/workshops/SEAGC. Retrieved 07/11/2016

[3] Data center energy consumption trends. http://www1.eere.energy.gov/Femp/Program/dc_energy_consumption.html. Retrieved 11/11/2017

[4] IBM, *Green IT: Why Mid-Size Companies are investing Now*- IBM. [online]. Available: www-03.ibm.com/press/attachments/GreenIT-final-mar4.pdf. Retrieved 14/11/2017

[5]Bunse C., Höpfner H., Roychoudhury S., Mansour E (2009). Energy Efficient Data Sorting Using Standard Sorting Algorithms. In International Conference on Software and Data Technologies ICSOFT 2009: Software and Data Technologies pp. 247-260 [ONLINE] https://link.springer.com/chapter/10.1007/978-3-642-20116-5_19. Retrieved on 4/11/2017

[6]Rivoire, S., Sharh, M. A., Ranganathan, P. and Kozyrakis, C. (2007), "Joulsort: A balanced energy efficiency benchmark", in proceedings of the 2007 ACM SIGMOD international conference on management of Data, ser.SIGMOD '07. New York,, USA: ACM, 2007, pp. 365-376. [Online]. Available: http://doi.acm.org/10.1145/1247480.1247522. Retrieved 18/11/2016

[7] Bunse, Christian; Hopfner, Hagen; Mansour, Essam and Roychoudhury, Suman (2009), "Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments" In *Mobile data Management Systems, Services and Middleware*, 2009. MDM '09 Tenth International Conference on May 2009, pp 600-607. Retrieved [Online] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.543.8109&rep=rep1&type=pdf on 29/09/2017.

[8] Carroll, Aaron and Heiser, Gernot (2010). An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010.USENIX Association, pp 4,36. Retrieved [online] https://ori-nuxeo.univlille1.fr/nuxeo/site/esupversions/5dd79154-0514-469fbd60-899cf9f57035

[9] Jain, R., Molnar, D., and Ramzan, Z. (2005). "Towards understanding algorithmic factors affecting energy consumption: Switching complexity, randomness and preliminary experiments," In *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '05, New York, NY, USA:

ACM 2005, pp 70-79, [online]. Available :https://pdfs.semanticscholar.org/.../249c47bdf85538b03b314c3f0fb70b0518c4.pdf. Retrieved 15/01/2017

[10] Rashid, M., Ardito, L., and Torchiano, M. (2015), "Energy Consumption Analysis of Algorithm Implementations", In *2015 proceedings of ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM). [online]. Available: http://doi.ieee/computer society.org/10.1109/esem.2015.7321198. Retrieved 04/11/2016

[11] Elkahlout, Ahmad and Maghari, Ashraf Y. A. (2017) "A comparative Study of Sorting Algorithms Comb, Cocktail and Counting Sorting" In *International Research Journal of Engineering and Technology (IRJET)* e-Volume:04 Issue:01 Jan 2017 (PDF Download Available). Available from: https://www.researchgate.net/publication/314753240_A_comparative_Study_of_Sorting_Algorithms_Comb_Cocktail_and_Counting_Sorting [accessed Nov 18 2017].

[12] https://en.wikipedia.org/wiki/Green computing. Retrieved on 2/11/2017

[13]Deepthi, T and Birunda, Antoinette Mary (2018) "Time and Energy Efficiency: A Comparative Study of Sorting Algorithms Implemented in C" In *International Conference on Advancements in Computing Technologies* (IJFRCSCE) -ICACT 2018 ISSN: 2454-4248 Volume: 4 Issue: 225–27 Available @ http://www.ijfrcsce.org. Retrieved 20/5/2018

[14]Aremu, D.R, Adesina, O. O., Makinde, O. E., Ajibola O. and O.O. Agbo-Ajala, O. O. (2013). "A Comparative Study of Sorting Algorithms" In *African Journal. of Comp & ICT*, Vol 6, No. 5, pp 199-206.

[15] Ali, Waqas; Islam, Tahir; Rehman, Habib Ur; Ahmed, Izaz; Khan, Muneeb; Mahmood, Amna (2016) "Comparison of Different Sorting Algorithms" In *International Journal of Advanced Research in Computer Science and Electronics Engineering* (IJARCSEE) volume 5, Issue 7, ISSN: 2277-9043. Pdf [Online] ijarcsee.org/index.php/IJARCSEE/article/view/554/526 . Retrieved on 19/09/2018

[16] Ayodele, Oluwakemi Sade and Oluwade, Bamidele (2019), A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms Using Three Programming Languages I: Execution Time Analysis, *Afr. J. MIS*, Vol 1, Issue 1, pp 1-18.

```
Start_Time      Invoke _System_clock
        Call_sortingAlgorithm_class/method
        End_Time  ←  Invoke_System_Clock
        Execution_Time = End_Time -  Start_
```

Figure 3.1: The algorithm for execution time

1. Start
2. Input: Data
3. Load a Sorting Algorithm
4. Sort data
5. Output
   i. Execution Time of sorted data
   ii. Measure CPU power utilized
6. i. Calculate Energy per unit
   ii. Print value of Energy utilized
7. Stop

Figure 3.2: Program flow of the Sorting

```
Start_Time      Invoke _System_clock
        Call_sortingAlgorithm_class/method
        End_Time ←Invoke_System_Clock
        Execution_Time = End_Time-  Start_
Time
```

Figure 3.3: The algorithm for execution time

APPENDIX

Table 4.1 Energy Comparison of Iterative sorting Algorithms Implementations in C

	Energy Consumption(Joule) of three Iterative sorting Algorithms implemented in C		
Data Size	Insertion	MergeSort	Quick sort
100,000	124.6739091	0.323556	0.143748
200,000	564.4451987	0.726528	0.2375
300,000	1203.689149	1.414728	0.485688
400,000	2173.657483	1.791912	0.678912
500,000	3360.210425	1.926332	1.32668

Table 4.2 Energy Comparison of Iterative sorting Algorithms Implementations in Java

	Energy Consumption(Joule) of three Iterative sorting Algorithms implemented in Java		
Data Size	Insertion	MergeSort	Quick sort
100,000	76.92423296	0.417088	0.38864
200,000	886.2912867	0.899096	0.717664
300,000	3942.79201	1.7496	0.9885
400,000	10900.63841	2.174592	1.365188
500,000	25093.63495	2.6344	1.65624

Table 4.3 Energy Comparison of Recursive sorting Algorithms Implementations in Python

	Energy Consumption(Joule) of three Iterative sorting Algorithms implemented in Python	
Data Size	MergeSort	Quick sort
100,000	124.6739091	58.84715509
200,000	564.4451987	124.9685281
300,000	1203.689149	185.6125664
400,000	2173.657483	263.97059

500,000	3360.210425	281.3010236

Table 4.4 Energy Comparison Of Recursive Quick Algorithms Implementations In C, Java And Python

DATA SIZE ('000)	C	JAVA	PYTHON
100	0.349168	0.41238678	58.84715509
200	1.114224	0.792816928	124.9695281
300	1.909656	1.115287744	185.6125664
400	2.6784	1.527858364	263.97059
500	3.578646	1.600245976	281.3010236

Table 4.5 Energy Comparison Of Recursive Merge Algorithms Implementations In C, Java And Python

DATA SIZE ('000)	C	JAVA	PYTHON
100	0.77364	130.452224	
200	1.590732	531.14712	
300	2.45616	1325.416704	
400	3.616644	2600.851968	
500	4.396648	4107.6272	

Table 4.6 Energy Comparison of Iterative Insertion Algorithms Implementations In C++, Java And Python

DATA SIZE ('000)	C	JAVA	PYTHON
100	124.6739091	76.92423296	76723.02136
200	564.4451987	886.2912867	334925.2484
300	1203.689149	3942.79201	678011.6974
400	2173.657483	10900.63841	1219684.322
500	3360.210425	25093.63495	2097651.857

Table 4.7 Energy Comparison Of Iterative Merge Algorithms Implementations In C++, Java And Python

DATA SIZE ('000)	C	JAVA	PYTHON
100	0.323556	0.417088	75.25877822
200	0.726528	0.899096	147.4104068
300	1.726528	1.7496	229.2594413
400	1.791912	2.174592	320.2936903
500	1.926332	2.6344	448.7777238

Table 4.8 Energy Comparison Of Iterative Quick sort Algorithms Implementations In C++, Java And Python

DATA SIZE ('000)	C++	JAVA	PYTHON
100	0.143748	0.38864	61.55288501
200	0.2375	0.717664	132.4095482
300	0.485688	0.9885	205.3952788
400	0.678912	1.365188	282.8371916
500	1.32668	1.65624	335.5805971

Table 4.9 Power (Watt) of three Iterative and Recursive sorting Algorithms implemented in Java

	Power(Watt) of three Iterative and Recursive sorting Algorithms implemented in Java				
Data Size('000)	Iterative Insertion	Iterative MergeSort	Recursive MergeSort	Iterative Quick sort	Recursive QuickSort
100	30.4	7.84	8.96	6.94	5.98
200	93.8	12.02	13.54	10.94	8.56
300	199.06	14.58	15.24	13.18	11.84
400	333.1	16.18	16.64	15.62	12.26
500	502.4	17.8	16.96	16.08	11.62

Table 4.10 Power(Watt) of three Iterative and Recursive sorting Algorithms implemented in C

	Power(Watt) of three Iterative and Recursive sorting Algorithms implemented in C				
Data Size('000)	Iterative Insertion	Iterative MergeSort	Recursive MergeSort	Iterative Quick sort	Recursive QuickSort
100	8.1782342	3.54	18.42	2.42	6.28
200	9.5526198	3.84	18.54	1.9	13.36
300	9.567683	5.88	18.06	3.54	16.84
400	9.568679	5.91	18.49	2.56	16.74
500	9.7038726	4.39	18.52	3.4	17.09

Table 4.11 Power (Watt) of three Iterative and Recursive sorting Algorithms implemented in Python

	Power(Watt) of three Iterative and Recursive sorting Algorithms implemented in Python				
Data Size('000)	Iterative Insertion	Iterative MergeSort	Recursive MergeSort	Iterative Quick sort	Recursive QuickSort
100	35.94	27.78	8.1782	35.78	35.66
200	38.06	30.26	9.5526	35.9685713	36.22
300	37.18	31.8	9.5677	36.14	36.9
400	37.04	32.76	9.5687	35.98645963	36.92
500	36.7	36.08	9.7039	35.96	36.98

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
POWER USAGE ANALYSIS OF THREE SORTING ALGORITHMS IMPLEMENTED IN THREE PROGRAMMING LANGUAGES AND TWO IMPLEMENTATION STYLES.																	
DATA S	QUICK SORT JAVA		QUICKSORT C		ICKSORT PHYT		ERGE SORT JAV		MERGESORT C		ERGESORT PHYT		INSERTION	INSERTION	INSERTION PYTHON		
	ITERATIVI	RECUR	ITERATI	REC	ITERAT	RECUF	ITERAT	RECUF	ITERAT	RECUF	ITERAT	RECURSI	ITERATIVE	ITERATIVE	ITERATIVE		
100	6.94	5.98	2.42	6.28	35.78	35.66	7.84	8.96	3.54	18.42	27.78	8.1782	8.1782342	30.4	35.94		
200	10.94	8.56	1.9	13.4	35.969	36.22	12.02	13.54	3.84	18.54	30.26	9.5526	9.5526198	93.8	38.06		
300	13.18	11.84	3.54	16.8	36.14	36.9	14.58	15.24	5.88	18.06	31.8	9.5677	9.567683	199.06	37.18		
400	15.62	12.26	2.56	16.7	35.986	36.92	16.18	16.64	5.91	18.49	32.76	9.5687	9.568679	333.1	37.04		
500	16.08	11.62	3.4	17.1	35.96	36.98	17.8	16.96	4.39	18.52	36.08	9.7039	9.7038726	502.4	36.7		

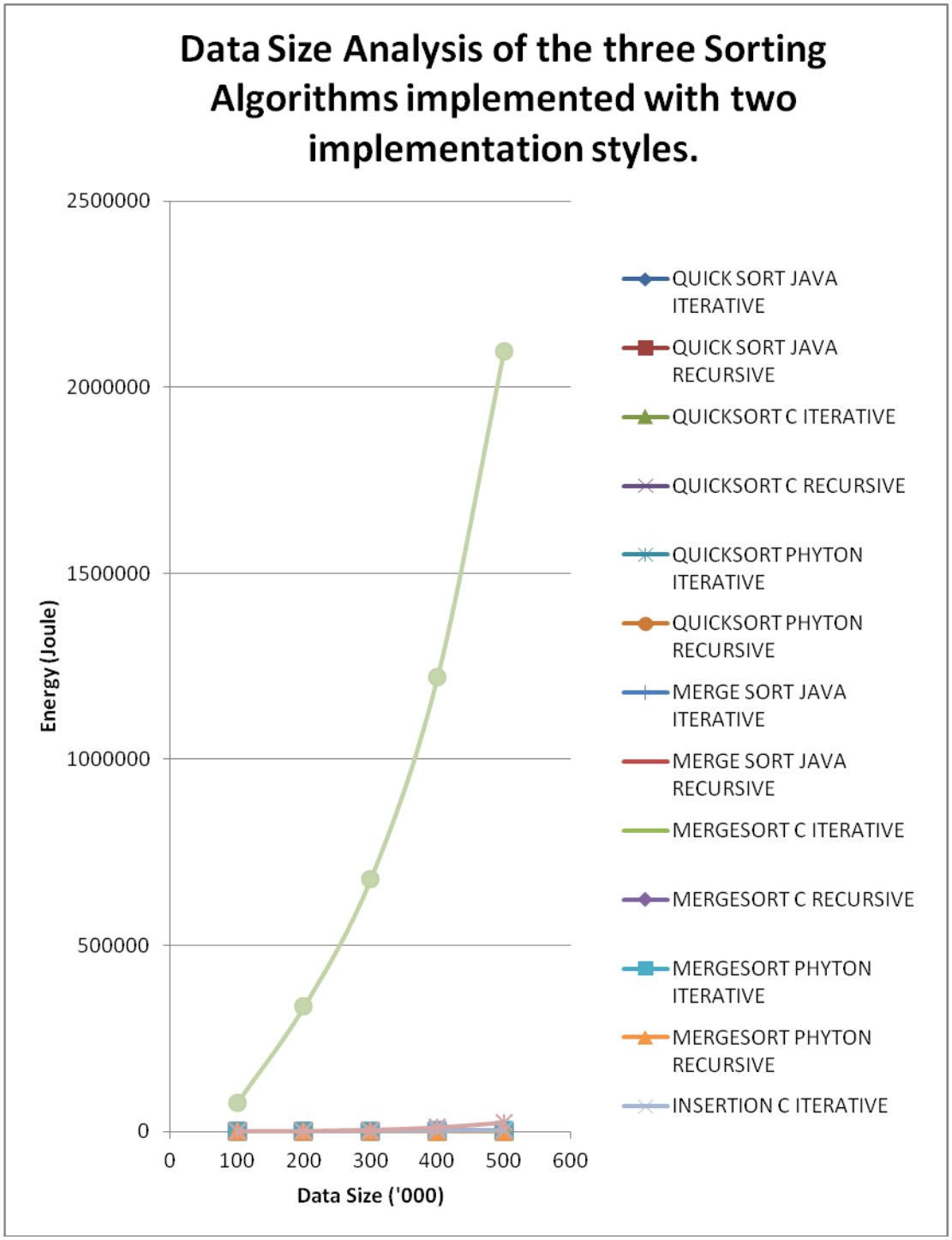


Figure 1: **Data Size Analysis of the three Sorting Algorithms implemented with two algorithm implementation styles and three programming languages**

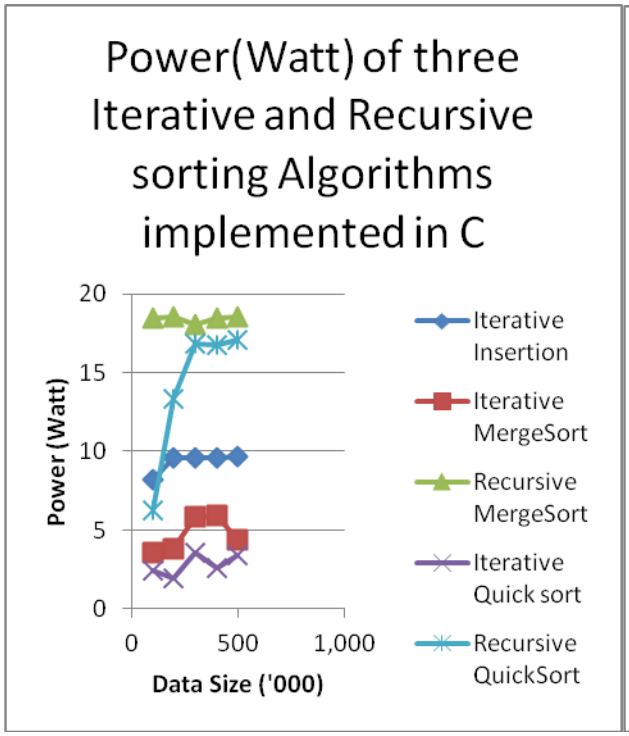


Figure 3: Power (Watt) Analysis of three Iterative and Recursive Sorting Algorithms in C

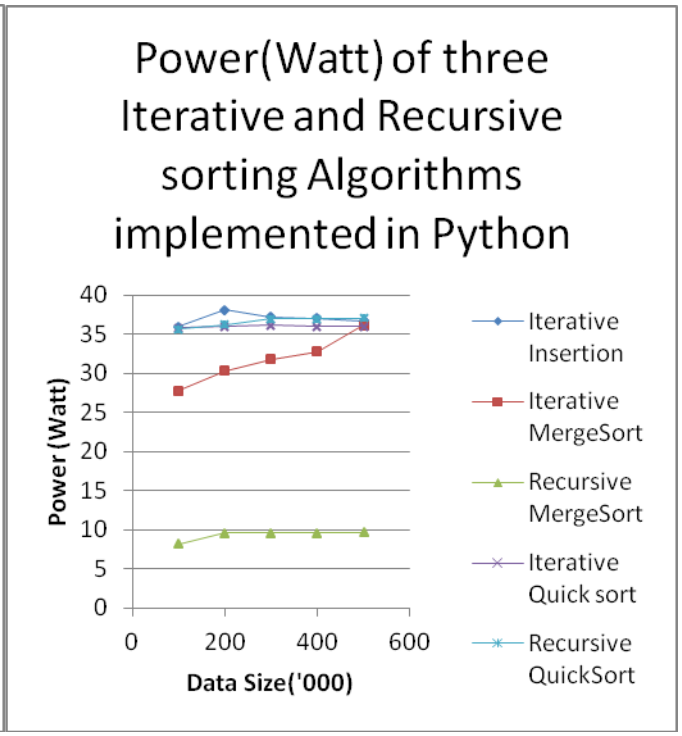


Figure 4: Power (Watt) Analysis of three Iterative and Recursive sorting Algorithms implemented in Python

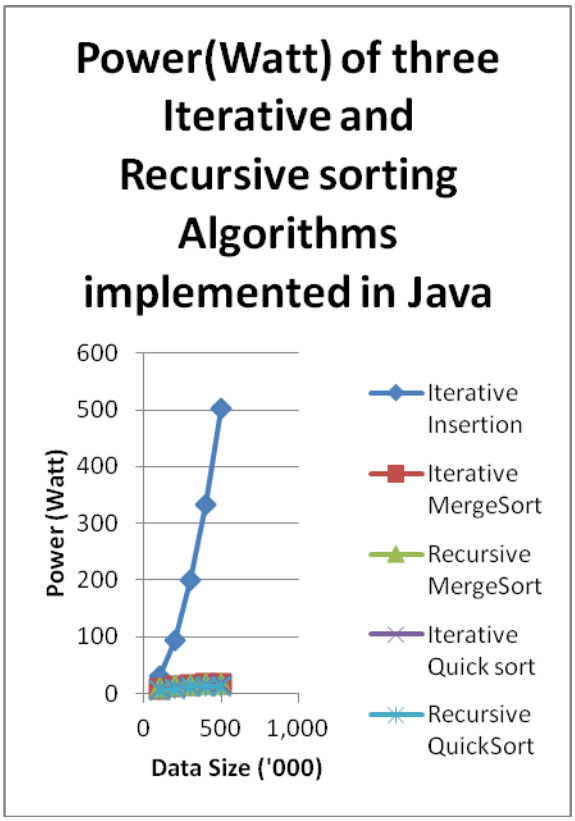


Figure 5: Power (Watt) Analysis of Three Iterative and Recursive Sorting Algorithms in Java

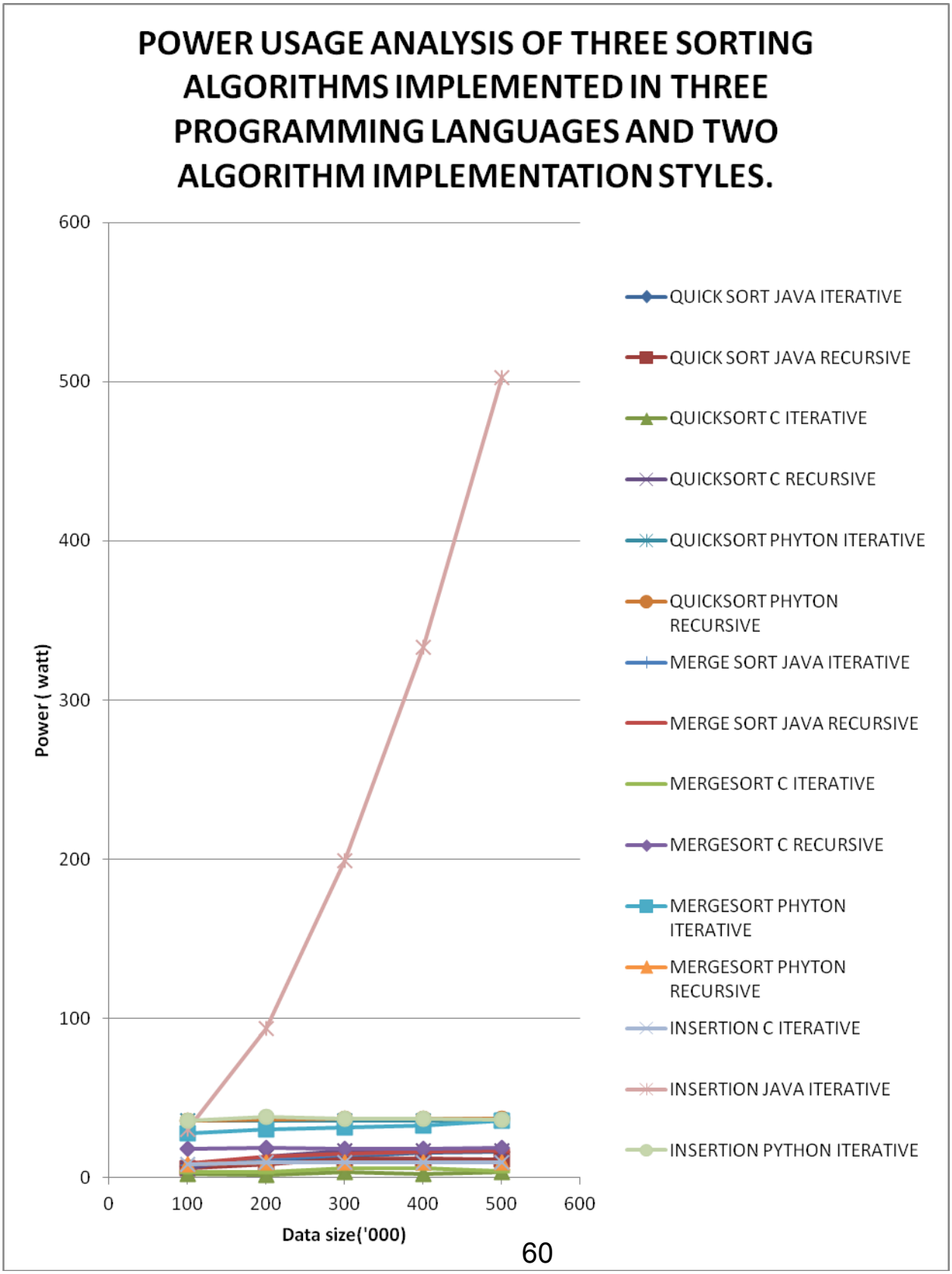


Figure 6 Power Usage Analysis Of Three Sorting Algorithms Implemented In Three Programming Languages And Two Algorithm implementation styles.

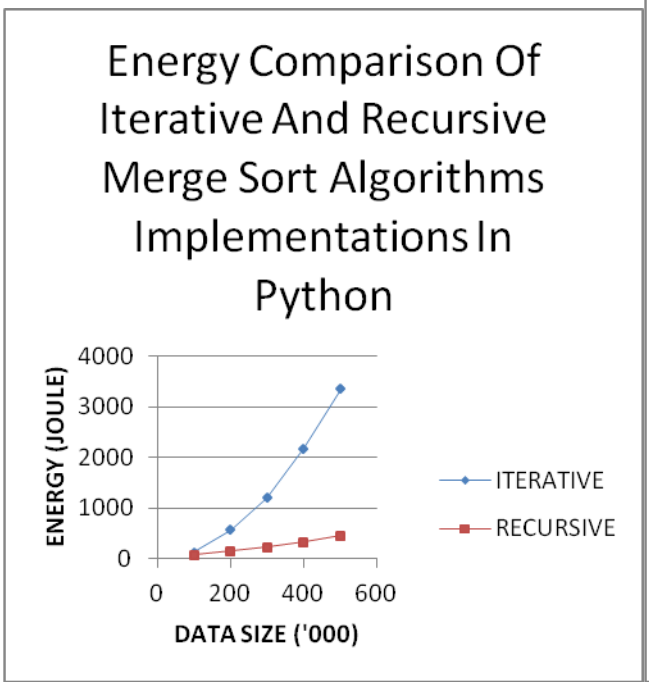


Figure7: Energy Comparison of Iterative and implemented in Python

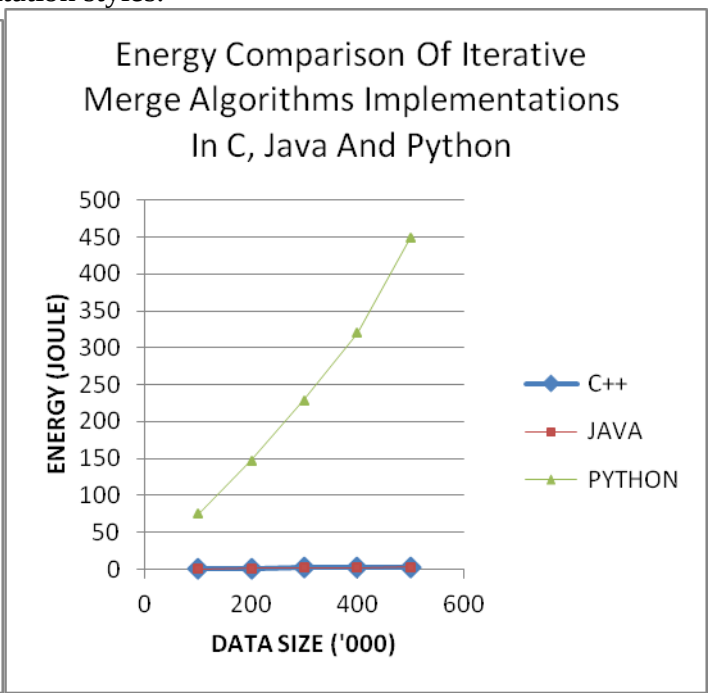


Figure 8: Energy Comparison Of Iterative Merge Sort Algorithms Implementations in C, Java and Python

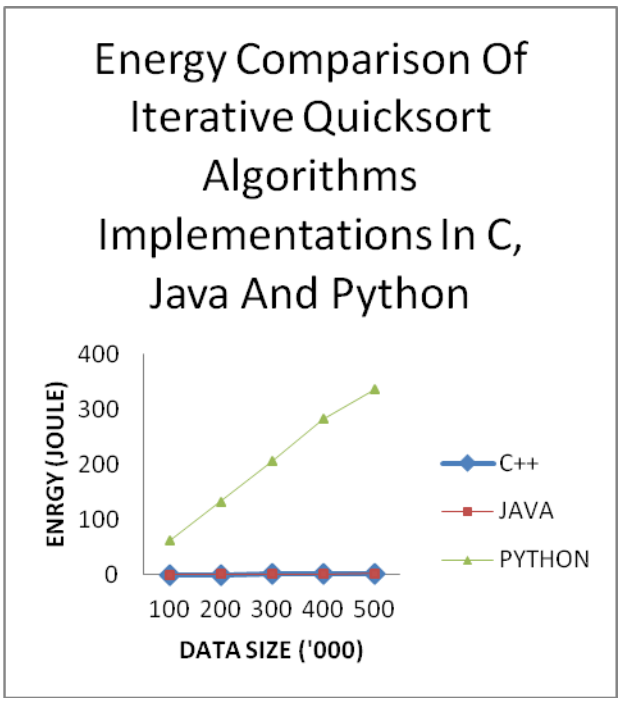


Figure 9: Energy Comparison of Iterative Quick sort Algorithm Implementations in C, Java nd Python

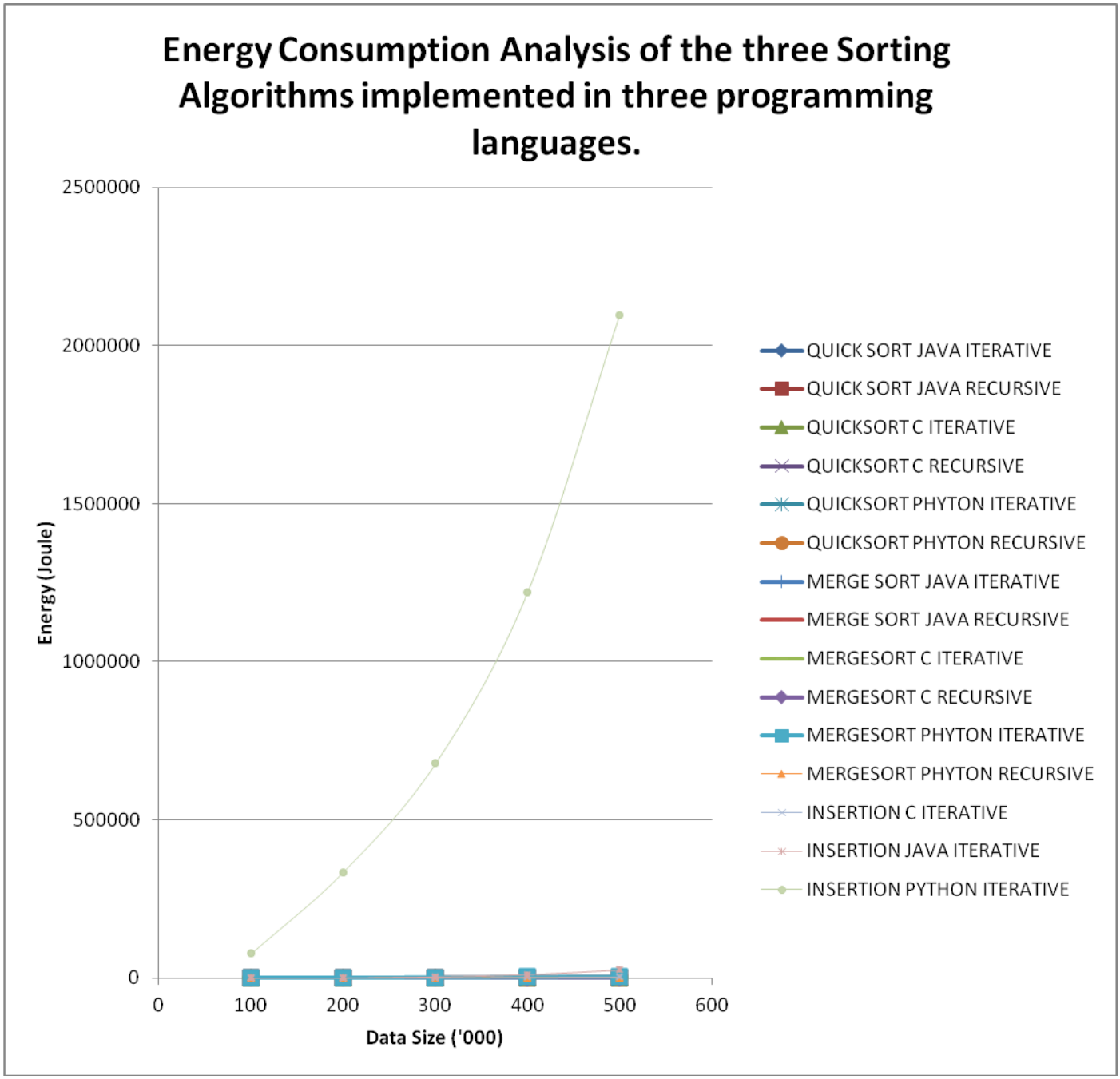
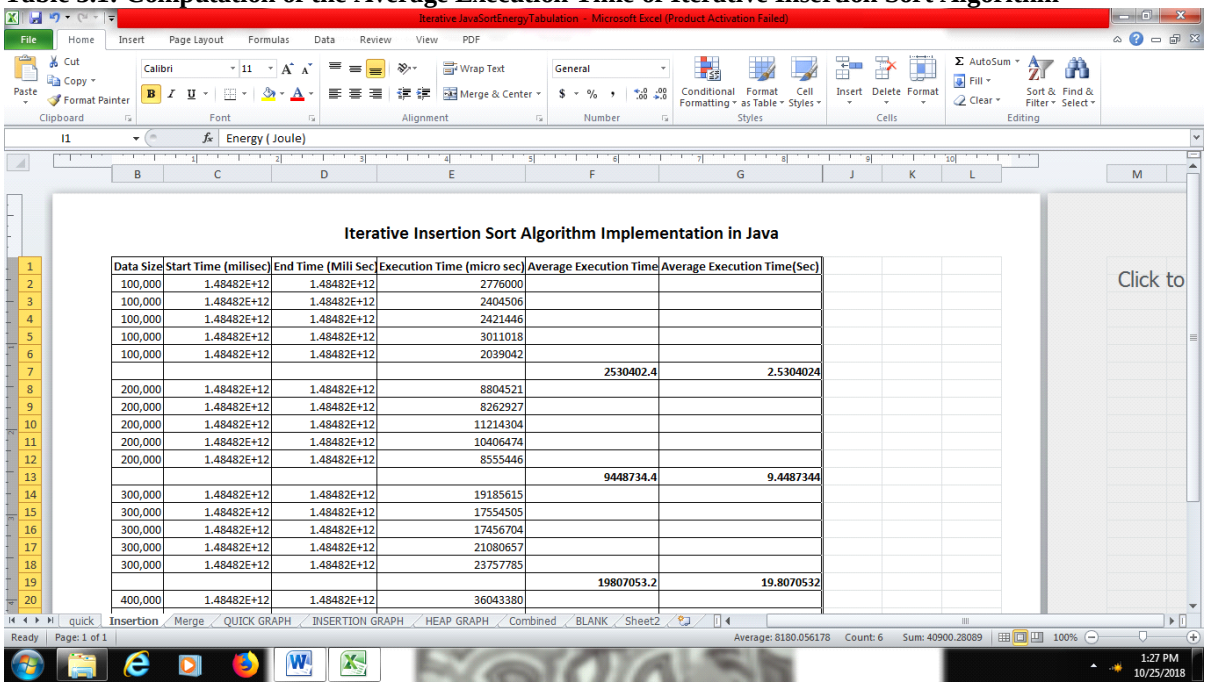


Figure 10: Energy Consumption Analysis of the three Sorting Algorithms implemented in three programming languages.

Table 3.1: Computation of the Average Execution Time of Iterative Insertion Sort Algorithm



Iterative JavaSortEnergyTabulation - Microsoft Excel (Product Activation Failed)

Energy (Joule)

Iterative Insertion Sort Algorithm Implementation in Java

Data Size	Start Time (milisec)	End Time (Mili Sec)	Execution Time (micro sec)	Average Execution Time	Average Execution Time(Sec)
100,000	1.48482E+12	1.48482E+12	2776000		
100,000	1.48482E+12	1.48482E+12	2404506		
100,000	1.48482E+12	1.48482E+12	2421446		
100,000	1.48482E+12	1.48482E+12	3011018		
100,000	1.48482E+12	1.48482E+12	2039042		
				2530402.4	**2.5304024**
200,000	1.48482E+12	1.48482E+12	8804521		
200,000	1.48482E+12	1.48482E+12	8262927		
200,000	1.48482E+12	1.48482E+12	11214304		
200,000	1.48482E+12	1.48482E+12	10406474		
200,000	1.48482E+12	1.48482E+12	8555446		
				9448734.4	**9.4487344**
300,000	1.48482E+12	1.48482E+12	19185615		
300,000	1.48482E+12	1.48482E+12	17554505		
300,000	1.48482E+12	1.48482E+12	17456704		
300,000	1.48482E+12	1.48482E+12	21080657		
300,000	1.48482E+12	1.48482E+12	23757785		
				19807053.2	**19.8070532**
400,000	1.48482E+12	1.48482E+12	36043380		