

Efficiency Comparison of Programming Languages in Algorithm Development

Rafeed Mohammad Sultan Shajreen Tabassum Diya
ECE (NSU) *ECE (NSU)*
North South University *North South University*
2022373642 2014061042

Md Shahidul Islam
ECE (NSU)
North South University
2011703642

Mir Faiyaz Hossain
ECE (NSU)
North South University
2011385042

rafeed.sultan@northsouth.edu shajreen.diya@northsouth.edu shahidul.islam1@northsouth.edu mir.hossain01@northsouth.edu

Abstract—In this project, we compared four different programming languages (Java, C, C++ and Python) using four different algorithms (Floyd-Warshall, Prim's Algorithm, Coin Change Problem, Convex Hull using Graham scan) to find which language performs better in terms of execution speed, memory consumption and CPU usage. We evaluated each metric based on varying input size and our results revealed that C and C++ generally outperforms Java and Python in all Categories.

Index Terms—execution-speed, memory, CPU, C, C++, Java, Pythom, Floyd-Warshall, Prim's, Coin-Change, Convex Hull.

I. INTRODUCTION

The performance of an algorithm shows variation for different programming languages, so there is a need to compare the algorithms in different programming languages. Different programming languages have different strengths and weaknesses. As a result, certain types of algorithms perform better in certain programming languages. Thus, comparing the efficiency of a certain algorithm in different languages will help us determine the most efficient language for that algorithm.

We have selected four different programming languages to implement four different algorithms to compare the efficiency. The languages we aimed to test are C, C++, Java, and Python, and the algorithms we have chosen are Prim's algorithm, Floyd-Warshall Algorithm, Convex Hull using Graham Scan, and Coin-Change Problem. The algorithms have been chosen based on different time complexity.

Our choice of programming languages were based on their different execution strategy (compiled or interpreted) and the language paradigm.

So far, we have successfully implemented all four aforementioned algorithms in Java, Python, C, and C++. After implementing the different algorithms, we measured the runtime, CPU utilization, and memory usage.

II. RELATED WORKS

When programming languages are compared in the context of efficiency of algorithms, the choice of programming languages and evaluation metrics follows a general trend. [1] In one research, 7 programming languages were compared

and several aspects of each programming language were investigated, such as: program length, programming effort, runtime efficiency, memory consumption and reliability. To evaluate the aforementioned aspects, the "Phonecode programming problem" was used. The comparison results are summarized below.

Compared to C/C++ programs, the script programs consumed twice the amount of memory.

Java programs consume three or four times as much memory as C or C++ programs.

In another research [2] 8 widely used programming languages were benchmarked by the Rosetta code repository, which contains 7'087 solution programs corresponding to 745 tasks, and the languages were categorized into three different programming language paradigms for comparison. The evaluation metrics involved comparison of lines of code, runtime performance, memory consumption and failure rate. Although lines of code (LOC) have been a common evaluation tool, we have found it to be less useful for our project. This research concluded that C is hard to beat when it comes to raw speed on large inputs.

Another research on 5 programming languages [3] using graph clustering algorithms also used the same evaluation metrics: runtime, memory and code size results. The paper associated with this research concluded that C++ is the fastest language for the challenge at hand, but they also show that Java, C# and F# come close under some circumstances.

To summarize, our goal is to use the same evaluation metrics on different algorithms implemented in different programming languages to find out whether our benchmarks remain consistent to the related works.

III. METHODS

We measured four algorithms' runtime, CPU utilization, and memory usage across Java, Python, C, and C++. We tried to keep the flow of the program consistent in all four languages.

In four programming languages, we wrote four programs: one for Prim's algorithm, one for Floyd-Warshall, one for Convex Hull, and one for the Coin change problem. We

followed pseudo-codes and took help from different resources to build our algorithms. To make our process easier, we defined a function for each algorithm.

A. Measuring the Runtime

We measured runtime by starting a high-precision clock before the execution of the algorithm and stopping the clock after the execution. The difference between the end and start times calculates the total execution time.

$$execution\ time = start\ time - end\ time \quad (1)$$

Note: For the Convex Hull problem, we measured execution time in nano-seconds in all programming languages. This is because the Convex hull executes much faster than other algorithms for the same set of inputs.

B. Measuring CPU utilization

CPU utilization provides insights into how effectively the running processes or tasks are utilizing the CPU. CPU utilization measures the percentage of time the CPU is actively processing tasks relative to the total time observed. CPU utilization can vary from 0% (idle) to 100% (fully utilized)

Table-I summarizes the libraries used in different languages.

C. Measuring Memory Consumption

Memory consumption refers to the amount of memory required by an algorithm to execute a given task. We measured memory consumption using external libraries for all four programming languages. Memory consumed is measured in MB in all languages.

TABLE I
RUNTIME AND RESOURCE UTILIZATION

Language	Runtime in ms	Memory used MB	CPU utilization %
Python	<code>time.time()</code>	<code>psutil</code>	<code>psutil</code>
Java	<code>System. current TimeMillis()</code>	<code>Management Factory</code>	<code>Management Factory</code>
C	<code>#include <time.h></code>	<code>#include <psapi.h></code>	<code>#include <windows.h></code>
C++	<code>#include <chrono></code>	<code>#include <psapi.h></code>	<code>#include <psapi.h></code>

IV. MODELS

A. Hardware Specification

All the programs are going to run on the following machine:

1. Windows 10 Pro
2. Processor: AMD Ryzen 5 3600 @ 3.60 GHz
3. RAM: 16 GB DDR4 memory (3200 MHz Bus speed).

B. Detailed Algorithm Description

The algorithms selected for the project have been listed below:

1. Algorithm Name: Prim's Algorithm

Type: *Graph-based*

Description: A greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.

Time Complexity: $O(E \log V)$

2. Algorithm Name: **Floyd-Warshall Algorithm**

Type: *Graph-based*

Description: Finds the shortest paths in a directed weighted graph with positive or negative edge weights (but with no negative cycles).

Time Complexity: $O(V^3)$

3. Algorithm Name: **Convex Hull using Graham Scan**

Type: *Geometry*

Description: Graham's scan is a method of finding the convex hull. Given n line points on plane P_1, P_2, \dots, P_n , this algorithm finds the smallest convex polygon that contains all points P_1, P_2, \dots, P_n .

Time Complexity: $O(n \log n)$

4. Algorithm Name: **Coin-Change Problem**

Type: *Dynamic Programming*

Description: The task is to find the minimum number of coins that add up to a given denomination amount. We are given a set (via an array) of coins of different denominations and assume that each one of them has an infinite supply.

Time Complexity: $O(n * c)$

V. INPUTS FOR ALGORITHMS AND STORING THE DATA

In all languages, our input value ranges from 0-500 for all algorithms except Floyd-Warshall. Due to its large time complexity, the input range has been restricted to 200. The algorithm functions are iteratively called with increasing input size and the evaluation metrics are stored in separate arrays. The results stored are then pulled into a .CSV file and using python's Matplotlib library we represented the data in graphs.

VI. RESULTS

Prim's Algorithm

1) CPU Utilization

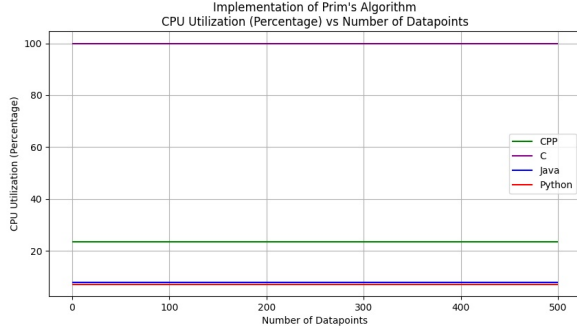


Fig. 1. CPU utilization in Java Python C C++

In the figure above, we can see that the CPU utilization for C is 100%, for C++ it is about 22%, for Java and Python it is about 8%. As the cpu utilization values were fluctuating, the average of the results were taken to plot the graph. From the graph, it can be concluded that C utilizes the CPU most efficiently whereas Python uses it least efficiently.

2) Memory Usage

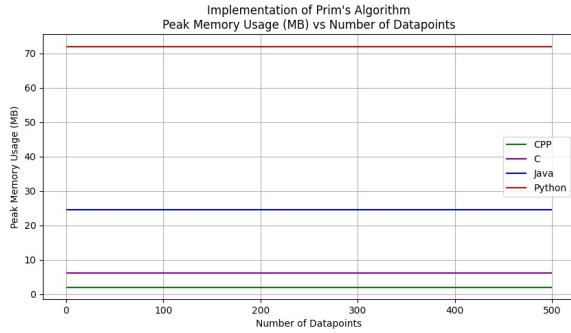


Fig. 2. Peak Memory Usage in Java Python C C++

From the figure above we can see that Python consumed about 72 MB, Java about 25 MB, C about 8MB and C++ about 2MB of memory. As the memory consumption values were fluctuating, the average of the results were taken to plot the graph. It can be concluded that as the memory consumption was the highest for Python it was the least efficient in terms of memory usage and C++ was the most efficient.

1

¹In our memory consumption analysis for all the algorithms tested on different programming languages, we observed a straight line in the graph indicating memory consumption remained constant regardless of the size of data points or inputs. To achieve a more accurate representation, we averaged the fluctuating results. However, it is noteworthy to mention that for larger data input sizes, our platform encountered difficulties in handling the execution. This highlights a limitation in scalability where our tools were a constraint for processing larger datasets.

3) Runtime

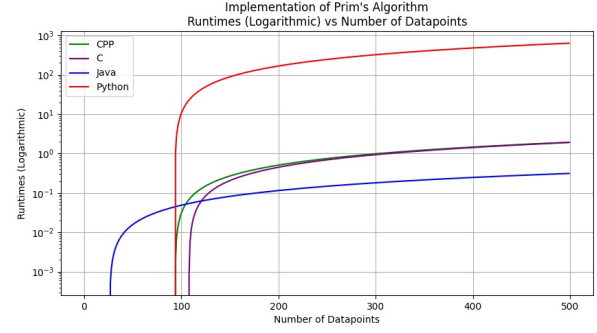


Fig. 3. Run-time in Java Python C C++

In the graph above, the y-axis uses logarithmic scales to compare all four languages in the same axes. As the input size increases, the runtime increases for all algorithms. From the graph, we can see that Python has the slowest execution time, Java has the fastest, and C/C++ executes slower than Java but faster than Python.

Convex Hull

1) CPU Utilization

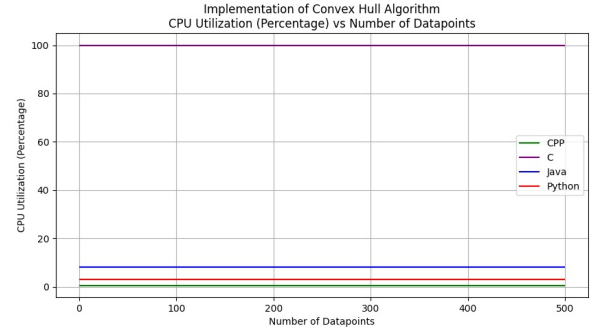


Fig. 4. CPU utilization in Java Python C C++

In the figure above, we can see that the CPU utilization for C is 100%, for C++ it is about 1%, for Java about 10% and for Python it is about 5%. As the cpu utilization values were fluctuating, the average of the results were taken to plot the graph. The number of data points has no correlation with CPU utilization. From the graph it can be concluded that C utilizes the CPU most efficiently whereas C++ uses it least efficiently.

2) Memory Usage

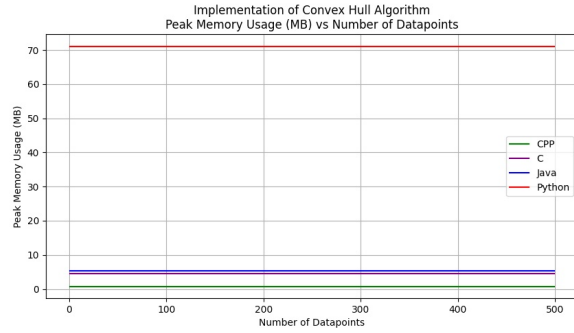


Fig. 5. Peak Memory Usage in Java Python C C++

From the figure above we can see that Python consumed about 72MB, Java about 8MB, C about 7MB and C++ about 1MB of memory. As the memory consumption values were fluctuating, the average of the results were taken to plot the graph. It can be concluded that as the memory consumption was the highest for Python it was the least efficient in terms of memory usage whereas C++ was the most efficient.

3) Runtime

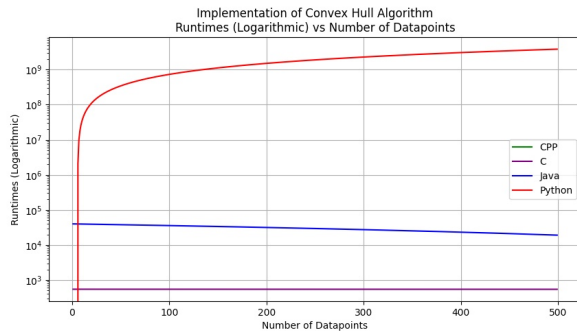


Fig. 6. Run-time in Java Python C C++

In the graph above, the y-axis uses logarithmic scales to compare all four languages in the same axes. As the input size increases, the runtime increases for all algorithms. From the graph, we can see that Python has the slowest execution time, C/C++ has the fastest, and Java executes slower than C/C++ but faster than Python. The runtime for this algorithm is calculated in nanoseconds due to its small time complexity.

Floyd-Warshall

1) CPU Utilization

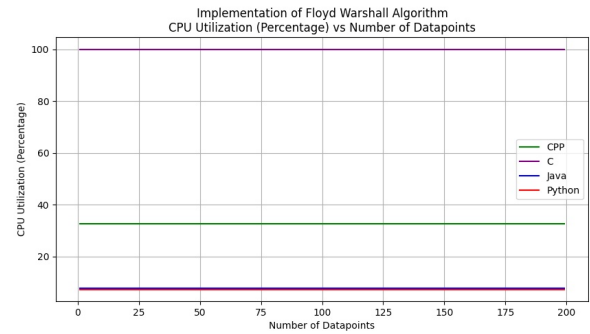


Fig. 7. CPU utilization in Java Python C C++

In the figure above, we can see that the CPU utilization for C is 100%, for C++ it is about 33%, for Java and Python it is about 5%. As the cpu utilization values were fluctuating, the average of the results were taken to plot the graph. From the graph, it can be concluded that C utilizes the CPU most efficiently whereas Python uses it least efficiently.

2) Memory Usage

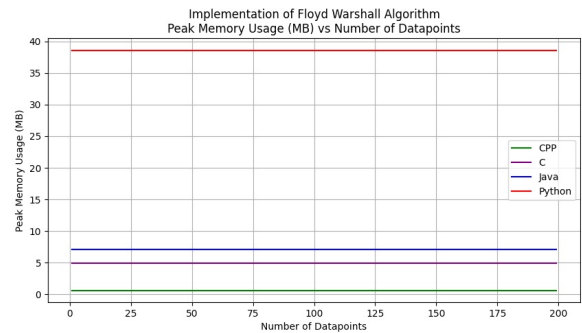


Fig. 8. Peak Memory Usage in Java Python C C++

From the figure above we can see that Python consumed about 37 MB, Java about 7 MB, C about 5MB and C++ about 1MB of memory. As the memory consumption values were fluctuating, the average of the results were taken to plot the graph. It can be concluded that as the memory consumption was the highest for Python it was the least efficient in terms of memory usage and C++ was the most efficient.

3) Runtime

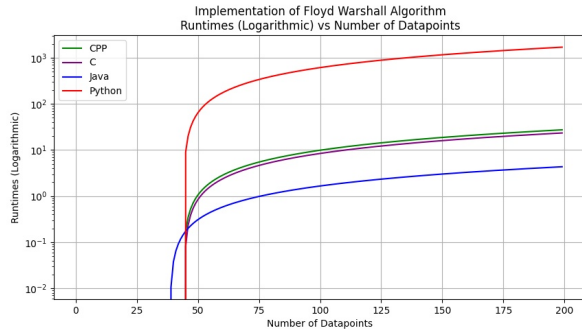


Fig. 9. Run-time in Java Python C C++

In the graph above, the y-axis uses logarithmic scales to compare all four languages in the same axes. As the input size increases, the runtime increases for all algorithms. From the graph, we can see that Python has the slowest execution time, Java has the fastest, and C/C++ executes slower than Java but faster than Python.

Coin-Change Problem

1) CPU Utilization

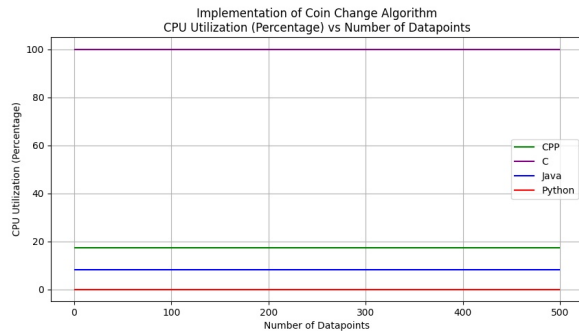


Fig. 10. CPU utilization in Java Python C C++

In the figure above, for Coin-Change problem, C utilizes 100% of the CPU, C++ uses 19%, Java uses 10% and Python 2%. As the CPU utilization values were fluctuating, the average of the results were taken to plot the graph. From the graph, it can be concluded that C utilizes the CPU most efficiently whereas Python uses it least efficiently.

2) Memory Usage

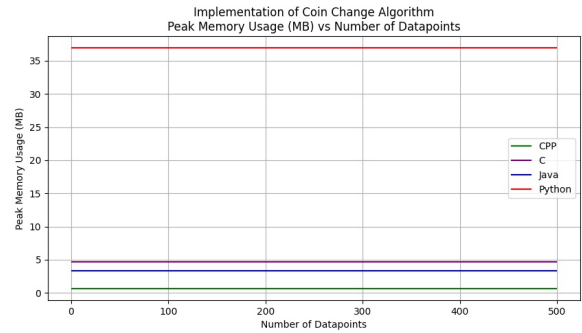


Fig. 11. Peak Memory Usage in Java Python C C++

In the figure above we can see that Python consumed about 38 MB, C about 5 MB, Java about 3MB and C++ about 2MB of memory. As the memory consumption values were fluctuating, the average of the results were taken to plot the graph. It can be concluded that as the memory consumption was the highest for Python it was the least efficient in terms of memory usage and C++ was the most efficient.

3) Runtime

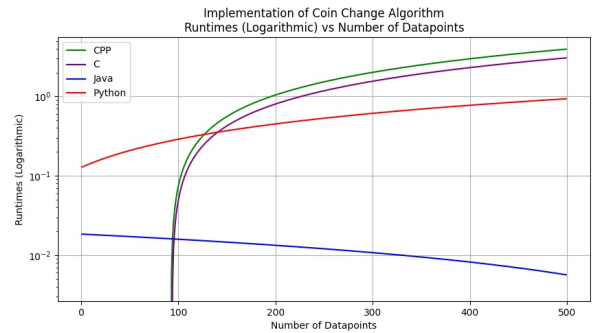


Fig. 12. Run-time in Java Python C C++

In the graph above, the y-axis uses logarithmic scales to compare all four languages in the same axes. Generally as input size increases, runtime also increases, but for Java the runtime decreased, which could be due to JVM optimization. The other languages follows the general trend. In the graph above we can see that Java executes the fastest, C++ and C the slowest, and Python lies between C/C++ and Java.

VII. DISCUSSION

The entire theme of the project revolved around **programmer efficiency**. The evaluation metrics used and the choice of algorithms are optimal for those who have programming experience.

In terms of **user efficiency**, not much has been done till now. Our results revealed that high-level language like Python fall behind other languages in the context of our evaluation, but users still prefer the simple English-like construct of Python over other languages, and despite its lacking, Python is used more frequently when compared to other languages.

In future works, we aim to tend towards user efficiency as well, where we can try to implement a wrapper that will allow users to write programs in Python, and a compile based approach will be implemented to make the language level with other ones.

Our results are based on varying input size only. The **correlation** between different metrics were not recorded during our run, however we aim to do so in the next phase of implementation. We also aim to add **Energy efficiency** as another evaluation metric, where we would see how execution speed, memory usage and CPU utilization effect memory usage. A similar work has been done in the paper [4] Energy efficiency across programming languages: how do energy, time, and memory relate?. This would tend to the rising question of which programming language is more energy efficient.

Lastly, we aim to diversify our work with algorithms from different fields such as Bioinformatics, Economics, etc.

VIII. CONCLUSION

Based on our evaluation metrics, the following conclusions can be drawn.

1) CPU Utilization

C is the most efficient language in terms of CPU utilization and python is generally the least efficient language in terms of CPU utilization.

2) Memory Usage

C++ is the most efficient language in terms of Memory usage and python is the least efficient.

3) Execution Time

Java is the fastest among all the languages and Python is the slowest.

The figure below summarizes the results.

Algorithm	CPU Utilization		Memory Consumption		Execution Time	
	Most efficient	Least efficient	Most efficient	Least efficient	Most efficient	Least efficient
Prim's	C	Python	C++	Python	Java	Python
Floyd Warshall	C	Python	C++	Python	Java	Python
Convex Hull	C	C++	C++	Python	C,C++	Python
Coin Change	C	Python	C++	Python	Java	C++

Fig. 13. Evaluation results in four programming languages.

Overall, C and C++ are the most efficient in terms of CPU utilization and memory usage, Java is the most efficient in terms of execution time and Python is the least efficient in all three criterias.

IX. CODE APPENDIX

You can find the project on GitHub at <https://tinyurl.com/CSE425Project>

X. CONTRIBUTION LIST

The table summarizes the contribution of individual group members in this project:





Group Member	Contribution towards project	Signature
Rafeed Mohammad Sultan 2022373642	<ul style="list-style-type: none">• Wrote the "Prim's" algorithm in C, C++, Java and Python.• Contributed in report writing "Methods and Results section"• Contributed in writing code for graph generation in Python using matplotlib and finding libraries for Memory consumption in all four programming language• Contributed creating slides for group presentation	
Shajreen Tabassum Diya 2014061042	<ul style="list-style-type: none">• Wrote the "Flyod-Warshall" algorithm in C, C++, Java and Python.• Contributed in report writing "Introduction, Results, and conclusion section"• Contributed in finding libraries for Memory consumption in all four programming languages.• Contributed creating slides for group presentation	
Md Shahidul Islam 2011703642	<ul style="list-style-type: none">• Wrote the "Convex Hull" algorithm in C, C++, Java and Python.• Contributed in report writing "Result section"• Contributed in writing code for graph generation in Python using matplotlib• Contributed in writing code to extract .CSV files across all languages.	
Mir Faiyaz Hossain 2011385042	<ul style="list-style-type: none">• Wrote the "Coin change problem" algorithm in C, C++, Java and Python.• Contributed in report writing "Results, Methods and discussion section"• Contributed in finding libraries for CPU utilization in all four programming languages.• Contributed in finding the strategy to calculate execution speed.	

Fig. 14. Contribution list.

REFERENCES

- [1] L. Prechelt, "An empirical comparison of seven programming languages," *Computer*, vol. 33, no. 10, pp. 23–29, 2000.
- [2] S. Nanz and C. A. Furia, "A comparative study of programming languages in rosetta code," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 778–788.
- [3] M. Stein and A. Geyer-Schulz, "A comparison of five programming languages in a graph clustering scenario." *J. Univers. Comput. Sci.*, vol. 19, no. 3, pp. 428–456, 2013.
- [4] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva, "Energy efficiency across programming languages: how do energy, time, and memory relate?" in *Proceedings of the 10th ACM SIGPLAN international conference on software language engineering*, 2017, pp. 256–267.