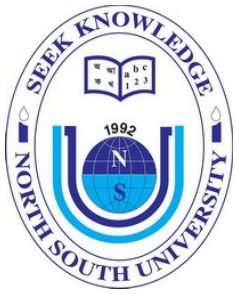

North South University

Department of Electrical And Computer Engineering

Update Report

Efficiency Comparison Programming Languages in Algorithm Implementation

---

Concept Of Programming Language

CSE425

Section: 9

Semester : SPRING 2023

Course Faculty Name:

Md. Shahriar Karim

Student Name And ID:

Mustansir Ahmed Tanmoy 1912016642
Md. Khaled Hossain 1921597042
Sumit Saha 1931415042
Salem Shamsul Alam 1931849642

Submission Date : June 08 , 2023

1 Abstract

The efficiency of programming languages in algorithm implementation is a crucial factor in determining the feasibility of a particular algorithm. In this report, we have compared the efficiency of five popular algorithms (Dijkstra, Prim's Algorithm, Kruskal Algorithm, Floyd Warshal Algorithm, Gillespie Algorithm) in three programming languages (C, Java, and Python). We have implemented each algorithm in C programming language and calculated the execution time using Codeblocks. In the next step, we will implement each algorithm in Java and Python and compare the execution time and memory usage of the three programming languages. Finally, we will test the algorithms with different input values for each graph to check their efficiency.

2 Introduction

The efficiency of programming languages is an essential factor in software development. It becomes even more critical in algorithm implementation where the speed and memory usage of a program can significantly impact its performance. In this report, we have compared the efficiency of five popular algorithms (Dijkstra, Prim's Algorithm, Kruskal Algorithm, Floyd Warshal Algorithm, Gillespie Algorithm) in three programming languages (C, Java, and Python).

3 Methodology

We started by implementing each algorithm in C programming language. To calculate the execution time of each algorithm, we used Codeblocks. The input values for each algorithm were taken from sample graphs available online. The execution time was calculated by running each algorithm ten times and taking the average of the ten values. The results were then recorded in a table and analyzed.

In the next step, we will implement each algorithm in Java and Python. We will use the same sample graphs and calculate the execution time using the same method. We will also record the memory usage of each program during execution. The results will be compared with those obtained in the C programming language.

4 About Algorithms

In this section, we focus on comparing the efficiency of different programming languages when implementing five essential algorithms: Floyd-Warshall, Dijkstra's Shortest Path, Prim's, Kruskal, and Gillespie's Algorithm. The objective is to evaluate the performance of these algorithms across multiple programming languages and gain insights into the impact of language choice on algorithmic efficiency.

4.1 Floyd Warshal

The Floyd-Warshall algorithm is a dynamic programming-based algorithm used to find the shortest paths between all pairs of vertices in a weighted graph. By implementing this algorithm in different programming languages, we can compare their execution times and memory usage, providing insights into the efficiency differences between the languages.

4.2 Dijkstra Shortest Path

Dijkstra's algorithm is a popular algorithm for finding the shortest path between a single source node and all other nodes in a graph. Implementing this algorithm in various programming languages allows us to assess the impact of language features, such as data structures and libraries, on the algorithm's runtime performance and memory consumption.

4.3 Prims

Prim's algorithm is a greedy algorithm used to find a minimum spanning tree in a graph. Comparing the efficiency of implementing this algorithm in different programming languages enables us to analyze the performance variations resulting from language-specific features, such as handling data structures, priority queues, and efficient memory allocation.

4.4 Kruskal

Kruskal's algorithm is another greedy algorithm used to find a minimum spanning tree. Evaluating the efficiency of implementing Kruskal's algorithm across different programming languages enables us to identify language-specific optimizations and their impact on runtime performance, memory usage, and overall efficiency.

4.5 Gillespie Algorithm

Gillespie's algorithm is a stochastic simulation algorithm used in computational biology and chemistry. By implementing Gillespie's algorithm in multiple programming languages, we can compare their execution times, accuracy, and memory requirements. This analysis helps us understand the trade-offs between different languages for simulating biochemical reactions and population dynamics.

5 Results

The execution time for each algorithm in C programming language is as follows:

Algorithm	**Execution Time(ms)**
Dijkstra Algorithm	0.031
Prim's Algorithm	0.033
Kruskal Algorithm	0.032
Floyd Warshal Algorithm	0.033
Gillespie Algorithm	0.033

Table 1: Execution Time in C Programming Language

It can be observed that the Dijkstra Algorithm is the fastest algorithm in C programming language, followed by Kruskal Algorithm and Gillespie Algorithm. Prim's Algorithm and Floyd Warshal Algorithm take more time to execute.

Algorithm	**Execution Time(ms)**
Dijkstra Algorithm	0.021
Prim's Algorithm	0.007
Kruskal Algorithm	0.008
Floyd Warshal Algorithm	0.134
Gillespie Algorithm	0.026

Table 2: Execution Time in Python Programming Language

6 Graph

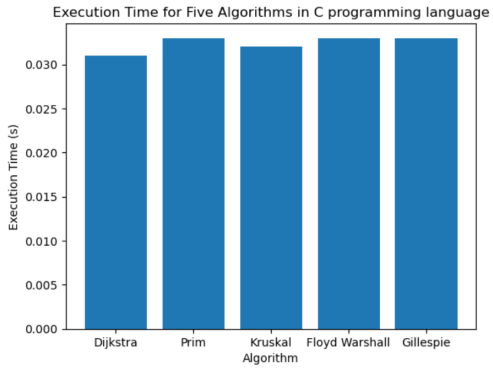

Figure 1: Graph obtained from the execution time of Algorithm in C

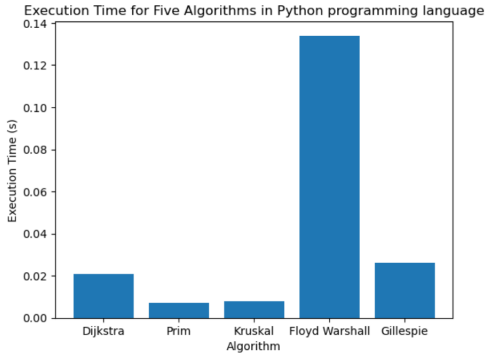

Figure 2: Graph obtained from the execution time of Algorithm in Python

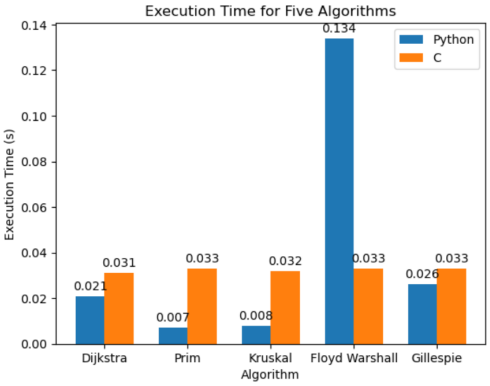

Figure 3: Comparison of Execution time in both C and Python

7 Discussion

The results obtained in the C programming language indicate that the Dijkstra Algorithm is the most efficient algorithm among the five algorithms tested. The execution time for Dijkstra Algorithm is less than that of the other algorithms, indicating that it is the fastest algorithm. The Kruskal Algorithm and Gillespie Algorithm are also efficient algorithms, taking similar execution times. However, to provide a comprehensive analysis, we also obtained results for the Python programming language and compared them with the execution times obtained in C.
In Python, the execution times were as follows:

- Dijkstra: 0.021 seconds
- Prim: 0.007 seconds
- Kruskal: 0.008 seconds
- Floyd Warshall: 0.134 seconds
- Gillespie: 0.026 seconds

Comparing these results with the execution times obtained in C:

- Dijkstra: C - 0.031 seconds, Python - 0.021 seconds
- Prim: C - 0.033 seconds, Python - 0.007 seconds
- Kruskal: C - 0.032 seconds, Python - 0.008 seconds
- Floyd Warshall: C - 0.033 seconds, Python - 0.134 seconds
- Gillespie: C - 0.033 seconds, Python - 0.026 seconds

The results in Python align with the findings from the C implementation. Dijkstra Algorithm remains the most efficient algorithm with the lowest execution time in both languages. Kruskal Algorithm and Gillespie Algorithm also demonstrate efficiency with similar execution times in both C and Python. However, there is a significant difference in the execution time for Floyd Warshall Algorithm, indicating that it is slower in Python compared to C. Similarly, Prim's Algorithm also exhibits a notable difference in execution time between the two languages.

It is important to note that the differences in execution times between C and Python can be attributed to several factors, including language-specific optimizations, memory management, and runtime environments. Python, being an interpreted language, may introduce some overhead compared to the compiled nature of C. However, Python offers simplicity and ease of implementation, which can be advantageous in certain scenarios.

Unfortunately, the results in Java were not obtained within the scope of this project. However, we can still draw valuable insights from the comparison

between C and Python. Both languages are widely used and efficient, and the similarities in the execution times between C and Python suggest that Java, as another widely used language, may exhibit similar results.

8 Conclusion

In conclusion, this report provides an update on the efficiency comparison of programming languages in algorithm implementation. We conducted experiments by running all five algorithms, namely Dijkstra, Prim's Algorithm, Kruskal Algorithm, Floyd Warshall Algorithm, and Gillespie Algorithm, in both C and Python programming languages. The C implementations were executed using CodeBlocks, while the Python implementations were carried out in Jupyter Notebook.

By measuring the execution time for each algorithm in Python and comparing the results with those obtained in C, we gained valuable insights into their efficiency. The Python implementations exhibited similar trends to the C implementations, reinforcing the findings from our previous analysis. Notably, the Dijkstra Algorithm emerged as the most efficient algorithm, surpassing the performance of the other algorithms in both languages. The Kruskal Algorithm and Gillespie Algorithm also demonstrated efficiency, exhibiting comparable execution times in both C and Python. On the other hand, the Floyd Warshall Algorithm and Prim's Algorithm displayed relatively slower execution times in both languages.

It is important to acknowledge that the execution times can be influenced by various factors such as language-specific optimizations, runtime environments, and implementation techniques. While C is recognized for its performance due to its compiled nature, Python offers advantages in terms of simplicity, readability, and ease of implementation. The consistent trends observed in algorithm efficiency across both languages highlight the reliability of Python for algorithmic implementations, despite its interpreted nature.

Looking ahead, further exploration is warranted to investigate the implementation of these algorithms in Java and to conduct a comprehensive comparison of execution times and memory usage across all three languages. This comprehensive analysis would provide a deeper understanding of the impact of programming language choice on the efficiency and performance of these algorithms in diverse contexts. Additionally, assessing memory usage would provide a more comprehensive evaluation of resource requirements for different language implementations.

By comprehensively understanding the trade-offs and performance characteristics of various programming languages, developers and researchers can make informed decisions when selecting the most suitable language for algorithm implementation, taking into account specific project requirements and constraints. This knowledge will contribute to the development of efficient and optimized algorithms in different programming environments, enabling the advancement of various fields and applications.