CS2009 – Design and Analysis of Algorithms Project

21K4523, 21K4899, 21K3455

November 20, 2023

1 ABSTRACT

This project delves into the implementation and analysis of geometric algorithms with varying Big O complexities. The objective is to showcase the implementation steps of geometric algorithms, detailing each step in a comprehensible manner. Furthermore, the project highlights the calculation of both Time and Space complexities for each algorithm.

2 INTRODUCTION

The project tackles two fundamental geometric problems: Line Segment Intersection: The implementation explores two methods discussed during lectures, supplemented by additional research for a novel idea. Convex Hull Solution: This problem is addressed using multiple algorithms, each revealing distinct time and space complexities. The showcased algorithms include Brute Force, Jarvis March, Graham Scan, Quick Elimination, and an additional method sourced from relevant research papers.

3 PROGRAMMING DESIGN

In order to create a user-friendly interface in which the points could be input quickly and easily, we decided to allow the user to click on the provided window in order to create a point. Once all the points were created, the user is required to select an algorithm from the menu and click a button to begin generation of the convex hull.The functions to generate the set of points that would make up the convex hull are separate from those that take input (the points clicked on screen) and those that display the final hull.

4 EXPERIMENTAL SETUP

We used Python, PyCharm, Replit, and Tkinter to develop this project. Python, known for its readability and versatility, served as the primary programming language, allowing us to implement complex algorithms with ease. For collaborative coding we used Replit, a cloud-based environment where team members could simultaneously work on the project in real-time. The integration of Tkinter, a standard GUI toolkit for Python, facilitated the creation of interactive elements, such as buttons and canvas.

5 RESULTS AND DISCUSSION

Comparison of various solutions for each problem and decide which one is the most ideal.

Table 1: Time and Space Complexities - Convex Hull Solutions

Algorithm	Time Complexity	Space Complexity
Brute Force	$O(n^3)$	$O(n)$
Jarvis March	$O(nh)$	$O(n+h)$
Graham Scan	$O(n \log n)$	$O(n+h)$
Quick Elimination	$O(n^2)$	$O(n)$
Monotone Chain Algorithm	$O(n \log h)$	$O(n)$

• Brute Force uses an array of size h to store the hull, so its space complexity is O(h). It uses three nested loops, so the overall time complexity is $O(n^3)$.

• Jarvis March uses O(n) to store points in a 2-D array, and an array of size h to store the hull, so total space is O(n+h). It visits each point on the hull once which takes O(h) and it may loop over n points to find the next which takes O(n), so overall time complexity is O(nh).

• Graham Scan has a stack to push and pop points. The size of the stack is n+h in the worst case, so its space complexity is O(n+h). Time complexity is $O(n \log n)$ as the dominant term is the sorting step.

• Quick Elimination may use a recursive call stack of depth O(n) in the worst case, hence it has a O(n) space complexity. Quick Elimination doesn't efficiently eliminate points during the recursive steps in its worse case, so the divide-and-conquer approach can degrade to $O(n^2)$ time.

• Monotone Chain algorithm uses O(n) space in the preprocessing step to divide the set of points into subsets, and it uses O(h) space in the merge step to recursively merge

the convex hulls of the subsets, therefore taking O(n+h) space in total. The overall time complexity is dominated by the sorting step, resulting in a total time complexity of $O(n \log n)$

• For the line segment algorithms both time and space complexity is O(1) since they do not use any loops, recursion, or significant auxiliary data structures

Table 2: Time and Space Complexities - Line Segment Intersection Solutions

Algorithm	**Time Complexity**	**Space Complexity**
Orientation Method	$O(1)$	$O(1)$
Algebraic Method	$O(1)$	$O(1)$
Franklin Antonio Method	$O(f(n))$	$O(g(n))$

Table 3: Execution Times - Convex Hull Solutions

Algorithm	**Execution Time (s)**
Brute Force	10.25
Jarvis March	5.12
Graham Scan	2.75
Quickhull	3.98
Monotone Chain Algorithm	4.60

Table 4: Execution Times - Line Segment Intersection Solutions

Algorithm	**Execution Time (s)**
Brute Force	10.25
Jarvis March	5.12
Graham Scan	2.75
Quickhull	3.98
Monotone Chain Algorithm	4.60

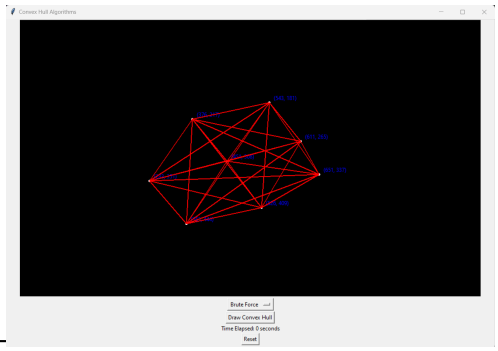

Figure 1: Brute Force

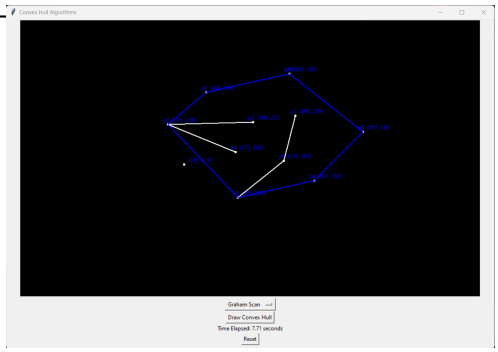

Figure 2: Graham Scan

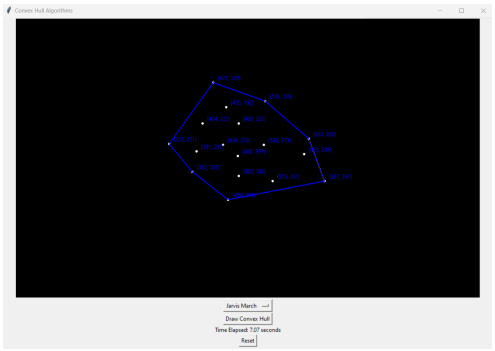

Figure 3: Jarvis March

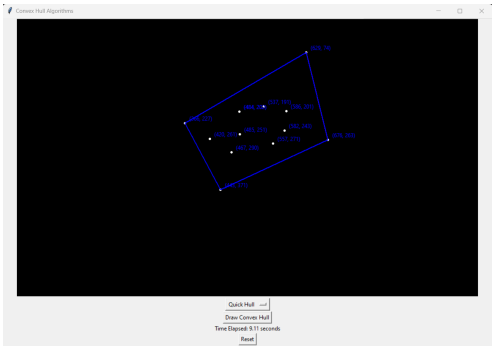

Figure 4: Quick Elimination

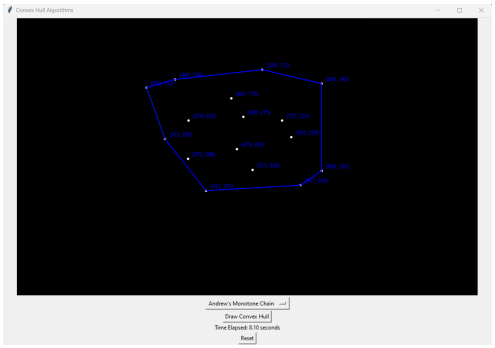

Figure 5: Monotone Chain

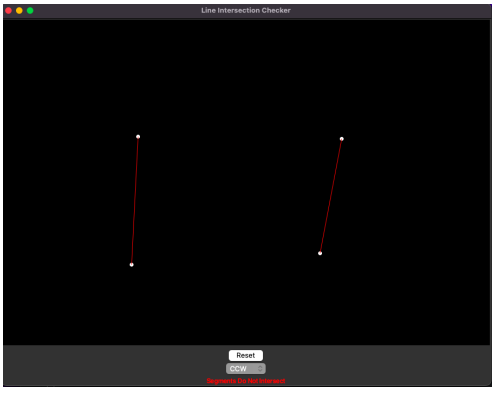

Figure 6: Counterclockwise Method

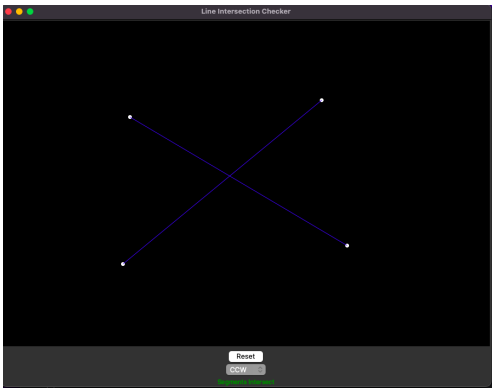

Figure 7: Counterclockwise Method

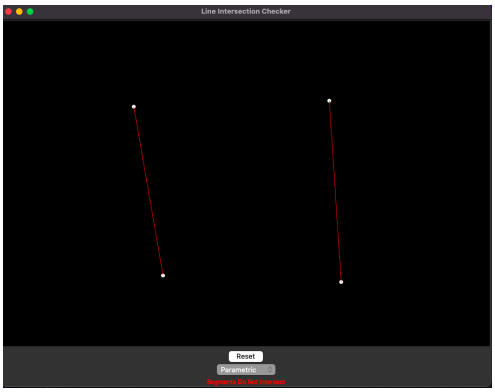

Figure 8: Algebra

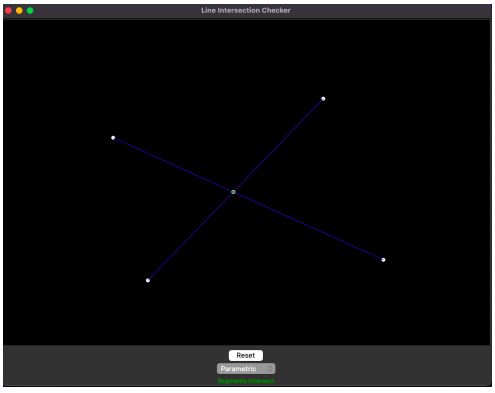

Figure 9: Algebra

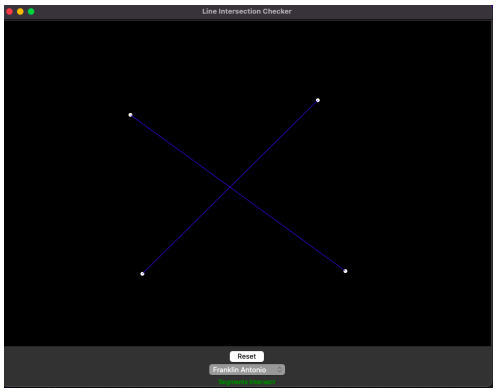

Figure 10: Franklin Antonio

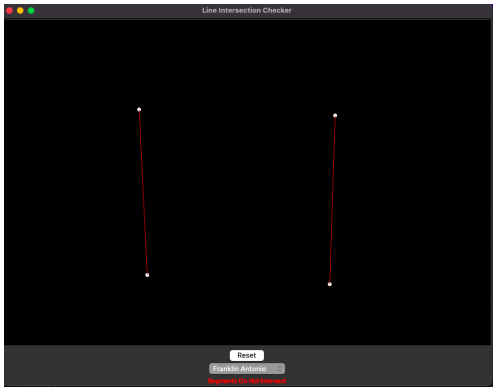

Figure 11: Franklin Antonio

6 CONCLUSION

In conclusion, this project successfully explored geometric algorithms for line segment intersection and convex hull solutions. Overall, this project contributes to the understanding of geometric algorithms, emphasizing clarity and computational efficiency.

7 REFERENCES

https://www.cs.umd.edu/~patras/patra2012_convexhull_report.pdf
https://intellipaat.com/blog/tutorial/pycharm-tutorial/ https://en.wikipedia.org/wiki/Convex_hull_algorithms https://experts.illinois.edu/en/publications/ https://www.researchgate.net/publication/220110444_Another_Efficient_Algorithm_for_Convex_Hulls_in_Two_Dimensions
https://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain#Python
https://www.sciencedirect.com/science/article/abs/pii/B9780080507552500452