Design And Analysis of Algorithm Project

Zeeshan Mustafa (21k3919)
Suleman Mohiuddin (21k3887)

20/Nov/2023

Abstract

This project offers a dynamic exploration of Geometric Algorithms, revealing their intricate details and diverse Big O complexities. Users actively participate by inputting varying numbers of points in the 2D plane, influencing the outcomes. The program features an interactive simulation of each algorithm's step-by-step execution, implemented and tested in Java, and presented through a user-friendly Swing GUI. The project's focus centers on Convex Hull and Line Segment Intersection algorithms, providing an immersive learning experience within a limited scope.

1 Introduction

Our project delves into the realm of Line Segment Intersection and Convex Hull algorithms, employing robust methodologies to address fundamental geometric problems. For line segment intersections, we incorporate the *Orientation* method, *HertelMehlHorn*, and the *SweepLineIntersectionChecker* (as per relevant research). These algorithms meticulously ascertain whether two given line segments intersect. Simultaneously, our Convex Hull algorithms, including *Brute Force*, *Jarvis March*, *Graham Scan*, *Quick Elimination*, and *Divide And Conquer* (referencing research), play a crucial role in determining the smallest convex polygon enveloping a set of points.

These geometric algorithms hold pivotal significance, finding applications across diverse fields. Efficient computation becomes paramount, prompting a meticulous analysis and comparison of their time and space complexities. The project's approach involves leveraging the outcomes of these algorithms to gain valuable insights, facilitating a concise yet comprehensive exploration of their computational intricacies.

2 Programming Design

Java, with its versatility, readability, and extensive libraries, serves as the cornerstone of our project, particularly Java Swing for the graphical user interface (GUI). The decision to opt for Java stems from its robust capabilities. In the implementation, a *Point* class captures 2D coordinates (x and y). The *ConvexHull* class encapsulates the Convex Hull algorithms, featuring simulations like *Jarvis March*, *Graham Scan*, *Brute Force*, *Quick Elimination*, and *Divide And Conquer*. Line Intersection simulations are housed in the *LineIntersection* class, incorporating three methods: one leveraging orientations and the other employing *SweepLine Intersection* and *Hertel-Mehlhorn* Line Segment Intersection algorithm.

To enhance user interaction, we've crafted the *MainMenu* class, offering a seamless interface for users to navigate between simulations and explore the functionalities of the program.

3 Experimental Setup

We facilitate user input for Convex Hull simulations, welcoming point entries directly onto the graph panel frame. For Line Segment Intersection simulations, users input two points for each line segment. This deliberate choice adds flexibility, enabling users to explore various point sets and introducing complexity for Hull determination.

Within the Swing GUI, interactive options abound. Users effortlessly navigate through geometric algorithms using menu buttons, visualizing each step by observing real-time updates on the Swing canvas. Simplicity prevails with convenient close buttons to exit simulations.

Our code structure is intentionally designed for seamless extension, allowing for the effortless incorporation of additional algorithms and future enhancements.

4 Results and Discussion

All the points are visually represented on the Swing Panel canvas during the algorithm simulations, and the results of Convex Hull and Line Segment Intersection are displayed in different ways.

For Convex Hull simulations, the program displays each step of the algorithm at a constant speed, and the final convex hull made by the points is represented by black lines between the hull points. For Line Segment Intersection, the program displays whether the line segments intersect and, if so, the point of intersection.

For all methods, the time in (ms) and (s) will depend on the points given by the user at runtime. Following are the time complexities mentioned in the table of the 15 random points we have given to the canvas.

Algorithm	**Time (ms)**	**Time (s)**
Brute Force	3	0.003
Jarvis March	4	0.004
Graham Scan	103	0.103
Quick Elimination	18	0.018
Divide and Conquer	33	0.033

Table 1: Time taken by different convex hull algorithms, first excluding the simulation time, then including time taken for simulation.

Algorithm	**Time (ms)**	**Time (s)**
Orientation Method	8.52942	-
Sweep Line Intersection	10.07366	-
Hertel-Mehlhorn Algorithm	10.07366	-

Table 2: Time taken by line segment intersection algorithms.

4.1 Space Complexities

Here is a table comparing the space complexities of various algorithms:

Algorithm	**Space Complexity**
Brute Force	$O(h)$
Jarvis March	$O(n+h)$
Graham Scan	$O(n+h)$
Quick Elimination	$O(n)$
Divide And Conquer	$O(n+h)$

Table 3: Comparison of Space Complexities for Convex Hull Algorithms

Algorithm	**Space Complexity**
Orientation Method	$O(1)$
Sweep Line Intersection	$O(n)$
Hertel-MehlHorn	$O(n)$

Table 4: Comparison of Space Complexities for Line Segment Intersection Algorithms

4.2 Time Complexities

Discuss the potential impact of your findings on future research in the field. Identify areas that may benefit from further investigation.

Algorithm	**Time Complexity**
Brute Force	$O(n^3)$
Jarvis March	$O(nh)$
Graham Scan	$O(n \log n)$
Quick Elimination	$O(n^2)$
Divide And Conquer	$O(n \log h)$
Orientation Method	$O(1)$
SweepLine Intersection	$O((n+k) \log n)$
Hertel-Mehlhorn	$O(n^2)$

Table 5: Comparison of Time Complexities

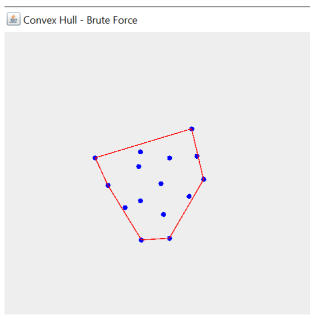
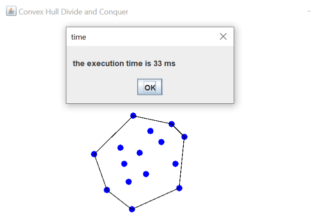
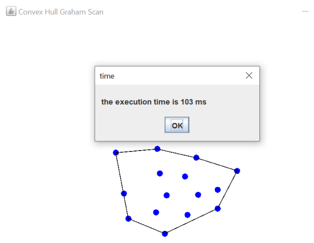


Figure 1: Brute Force Algorthim



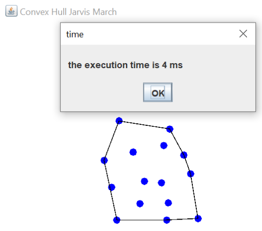
the execution time is 33 ms

Figure 2: Divide And Conquer.



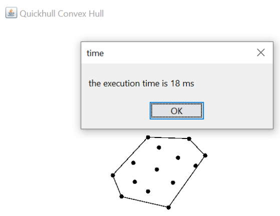
the execution time is 103 ms

Figure 3: Graham Scan.



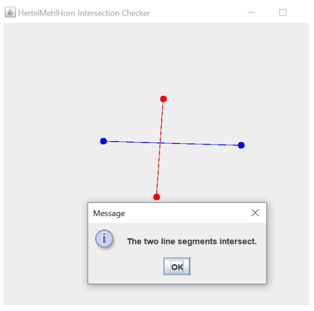
the execution time is 4 ms

Figure 4: Jarvis March.



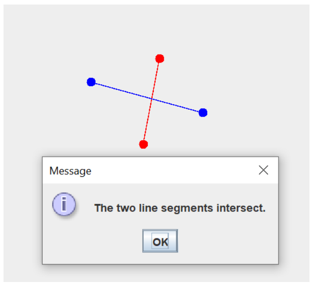
the execution time is 18 ms

Figure 5: Quick Elimination



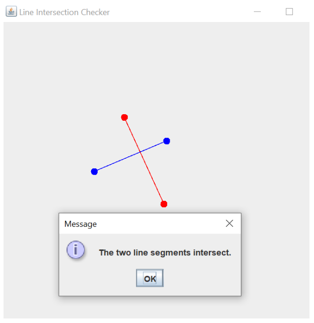
The two line segments intersect.

Figure 6: Hertel-MehlHorn



The two line segments intersect.

Figure 7: Sweep Line Intersection



The two line segments intersect.

Figure 8: Orientation Methofctd

5 Conclusion

In conclusion, our project offers a dynamic exploration of Convex Hull and Line Segment Intersection algorithms through an interactive Java-based Swing GUI. Users input points in the 2D plane, influencing algorithm outcomes displayed in real-time on the Swing canvas.

Implemented algorithms include Orientation, HertelMehlHorn, SweepLineIntersectionChecker, Brute Force, Jarvis March, Graham Scan, Quick Elimination, and Divide And Conquer. Java and Java Swing provide a versatile and user-friendly environment.

Time and space complexities are analyzed, revealing varying performance across algorithms. Convex Hull space complexities range from linear to logarithmic, while Line Segment Intersection complexities vary from constant to linear. Notably, the Hertel-Mehlhorn algorithm exhibits $O(n^2)$ time complexity.

Despite successful exploration, the project has limitations, focusing solely on Convex Hull and Line Segment Intersection. Future work could expand the scope, refine the interface, and conduct empirical studies. Our findings contribute to computational geometry understanding and lay a foundation for further research

6 References

1. Jarvis March Convex Hull Algorithm. GeeksforGeeks. https://www.geeksforgeeks.org/convex-hull-using-jarvis-algorithm-or-wrapping/

2. Graham Scan Convex Hull Algorithm. GeeksforGeeks. https://www.geeksforgeeks.org/convex-hull-using-graham-scan/

3. Quickhull Algorithm for Convex Hull. GeeksforGeeks. https://www.geeksforgeeks.org/quickhull-algorithm-convex-hull/

4. Orientation Method for Line Segment Intersection. https://iq.opengenus.org/line-segment-intersection-orientation/

5. Franklin Antonio's Algorithm for Line Segment Intersection. https://www.researchgate.net/publication/220523358_Faster_Line_Segment_Intersection

6. Hertel-Mehlhorn Algorithm for Line Segment Intersection. https://courses.csail.mit.edu/6.851/spring10/scribe/lec04b.tex

7. Computational Geometry Algorithms. Phyed Learn. https://phyed.in/algorithm/computational-geo

8. https://github.com/ermel272/convex-hull-animations. https://github.com/ermel272/convex-hull-animations