Design And Analysis Of Algorithm Project Report

Shayan Haider 21k-3211, Ahmed Ali 21k-3212, Taha Hassan 21k-4680

November 27, 2023

Abstract

This project explores geometric problem-solving methods with different complexities. Users interact with the system using drawings. It also includes a detailed analysis of how much time each algorithm needs. The project covers two main problems:

1 Introduction

This project focuses on implementing geometric algorithms, with their time complexities. Users can interact with the system by either visually drawing objects on the screen. The primary objective is to present visually appealing user interface, effectively illustrating the step-by-step processes of the algorithms employed. The project revolves around two core problem-solving tasks:

- Intersection of Two-Line Segments: Exploring multiple methods to determine if two line segments intersect, including approaches discussed in lectures and innovative idea derived from research.
- Solving Convex Hulls: Demonstrating and analyzing algorithms such as Brute Force, Jarvis March, Graham scan, Quick Elimination, and an additional method from research paper for solving the Convex Hull problem. The project aims to provide a visually comprehensive understanding of these geometric algorithms.

2 Programming Design

For this project, the programming language chosen is C# utilizing the WinForms framework to develop the user interface. C# was selected due to its robustness in handling graphical user interfaces. User interface is given below:

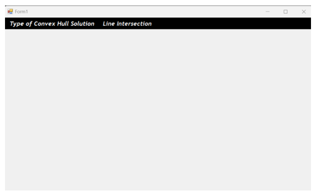


Figure 1: User Layout of Project

The above diagram features two sections: one for line segment intersection algorithms and another for Convex Hull algorithms. Each section offers various algorithm choices. Upon selection, a button appears to execute the chosen algorithm, displaying its time complexity upon clicking.

3 Experimental Setup

3.1 Input Data Preparation:

Design a method to accept input data. This involves allowing users to draw line segments.

3.2 Algorithm Implementation:

Implement each algorithm separately (for line segment intersection and Convex Hull) using C#. Ensure that each algorithm is encapsulated as a separate module for ease of testing.

3.3 Algorithm Execution:

Execute algorithms on various sets of line segments to determine intersections or compute Convex Hulls.

3.4 Performance Analysis:

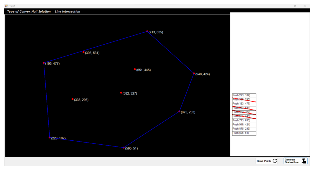
Calculate the time complexity of each algorithm by analyzing its execution time with varying input sizes.

3.5 Visualization:

Develop a graphical representation to display:

- Input line segments or datasets used for experiments.
- Executed steps and decisions made by algorithms (intersections found, Convex Hulls formed).

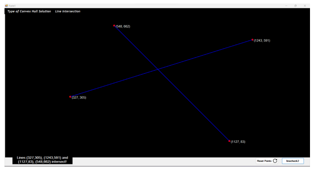
4 Results and Discussion



```
Push(223, 102)
~~Push(338, 295)~~
Push(193, 477)
~~Push(393, 531)~~
~~Push(582, 327)~~
~~Push(651, 445)~~
Push(713, 635)
Push(948, 424)
Push(875, 233)
Push(595, 51)
```

Figure 2: Graham Scan algorithm with Stack Visualization

The first figure (Figure 2) illustrates the Graham Scan algorithm with a visualization of the stack used during its execution. This visualization showcases the step-by-step process of the algorithm for computing the convex hull.



Lines (327,305), (1243,591) and (1127,83), (548,662) intersect!

Figure 3: Sweep Line Algorithm

The second figure (Figure 3) represents the Sweep Line Algorithm. It allows user to draw lines and shows user whether the given lines intersect or not.

5 Conclusion

In conclusion, The project explored geometric problem-solving with various algorithms, focusing on line segment intersection and Convex Hull. It showcased classic methods and introduced a new idea from research for line intersections. For Convex Hull, it demonstrated Brute Force, Jarvis March, Graham Scan, and Quick Elimination. The user-friendly interface visualized their operations, aiding understanding of complexities. Analyzing time complexities highlighted their efficiency and variations. Overall, the project aimed to illustrate, analyze, and encourage exploration of geometric algorithms.

6 References

- **A New Approach to Compute Convex Hull**
- **Introduction to Line Segments in Computational Geometry**

A Appendix

A.1 Algorithm Implementations

A.1.1 Intersection of Two-Line Segments

Two-line segment intersection algorithms were implemented:

- **Counter Clockwise (Lecture) :** This algorithm utilizes This algorithm assesses the orientation of three points to determine if they form a counterclockwise or clockwise arrangement. If the orientation is counter-clockwise for both endpoints of one line concerning the other line's endpoints, they might intersect
- **Sweep Line (Lecture):** Another approach discussed in the lecture. Please click here for detailed algorithm implementations.
- **Research-derived Method:** Research led to the implementation of an innovative approach. link . This method presents [describe the new idea and how it's implemented].

A.1.2 Convex Hull Solutions

Various algorithms for solving the Convex Hull problem were implemented:

- **Brute Force:** Check all possible combinations of points and identify those that contribute to the Convex Hull by examining their orientation relative to other points.
- **Jarvis March:** Start from the point with the lowest y-coordinate, proceed by selecting the point that forms the rightmost angle until returning to the starting point.
- **Graham Scan:** Graham Scan: Sort points by polar angle around the lowest point, construct the Convex Hull by considering each point's position relative to the top two points on the stack.
- **Quick Elimination:** Divide points into sets and eliminate those not contributing to the Convex Hull, based on their relative positioning to certain lines formed by extreme points.
- **Research-derived Method:** Additionally, an algorithm sourced from contemporary research papers was implemented. This algorithm, based on [reference/source], presents [explain the approach taken and its unique characteristics].

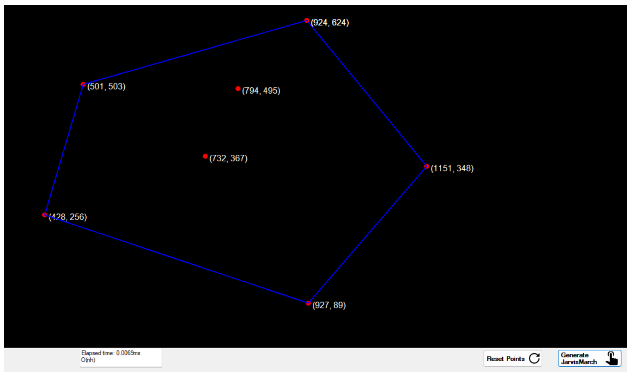
A.2 User Interface and Algorithm Steps

Click here to see User Interface and its interaction with system.

A.3 Time Complexity Calculation

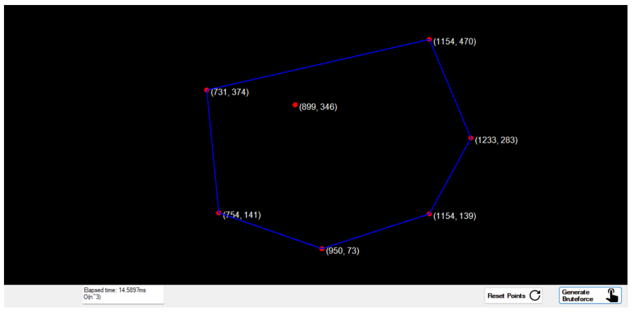
A.3.1 Convex Hull Solutions

The time for some of the Convex Hull solutions are listed below:



```
Elapsed time: 0.0069ms
O(nh)
```

Figure 4: Graham Scan Algorithm



```
Elapsed time: 14.5897ms
O(n^3)
```

Figure 5: Bruteforce Algorithm