Design and Analysis of Algorithms Project Report

K214945 Asad Ullah Khan - K213200 Abdur Razzaq - K213201 Arham Mehmood

November 20, 2023

Abstract

In this project we looked upon 5 Geometric Algorithms, and 3 Line segment intersection algorithms. We analyzed all of these algorithms on basis of their complexities and found out which algorithm would be beneficial for future work regarding convex hulls and segments in real-world domains.

1 Introduction

We worked on Brute Force, Graham Scan, Jarvis March, Quick Elimination, and a research based algorithm ”Andrew’s Monotone Chain” for computing convex hulls. Whereas for line intersection we used CCW tests, Algebraic Equations, and Line Sweep algorithm. Our project’s objective was to compute and compare convex hulls using different algorithms and understand line segments. This subjects to computer graphics, image processing, pattern recognition etc.

2 Your Programming Design

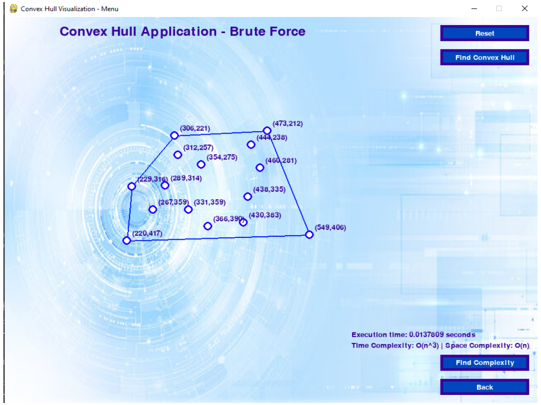
This project is implemented in Python version 3.19.1. Here we are basically using a list of 2D points Explain the programming language, show a simple diagram, and provide a quick overview of implemented algorithms.

3 Experimental Setup

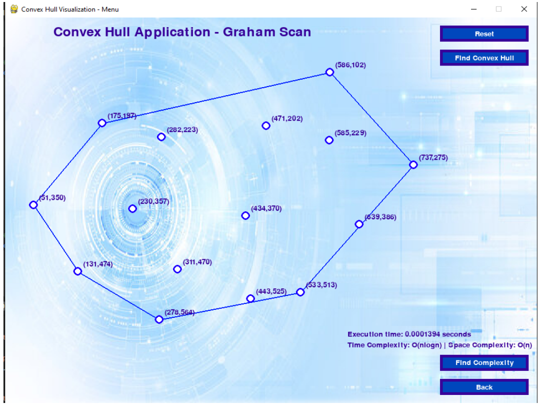
User uses his mouse to create points on the canvas dynamically and that is how the data is inputted to different functions. In line intersection the two lines are automatically drawn as points are created by user. We implemented this entire setup using Python’s library ”PyGame” which enabled us to create GUI and with appropriate header files we imported our self-built functions. For appropriate testing we used Visual Studio’s coding environment.

4 Results and Discussion

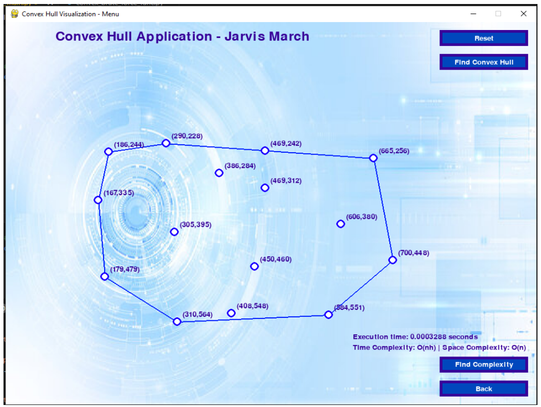
Below are Geometric Algorithms and Line Intersection Screenshots



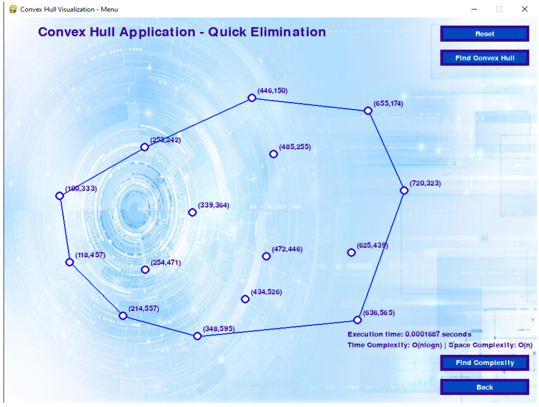
Execution time: 0.0137809 seconds
Time Complexity: O(n^3) | Space Complexity: O(n)



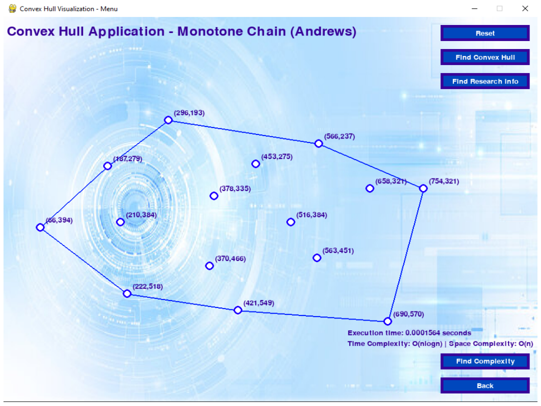
Execution time: 0.0001394 seconds
Time Complexity: O(nlogn) | Space Complexity: O(n)



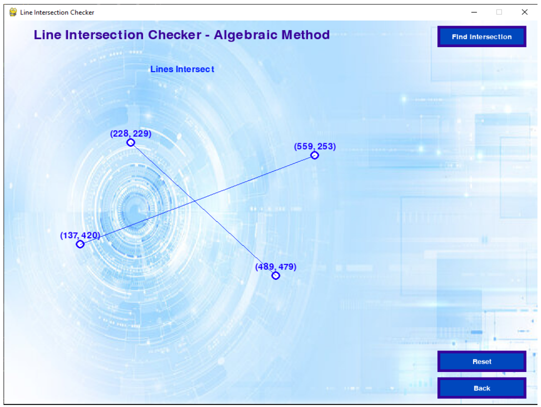
```
Execution time: 0.0003288 seconds
Time Complexity: O(nh) | Space Complexity: O(n)
```



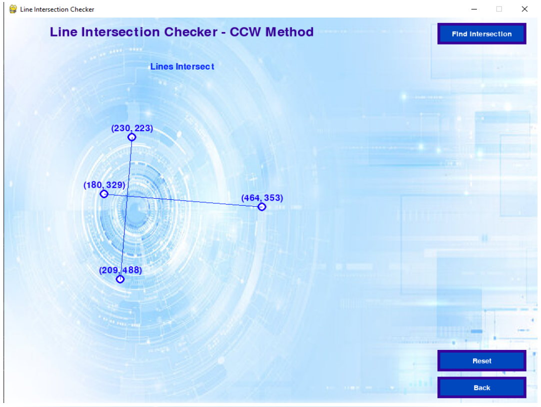
```
Execution time: 0.0001687 seconds
Time Complexity: O(nlogn) | Space Complexity: O(n)
```



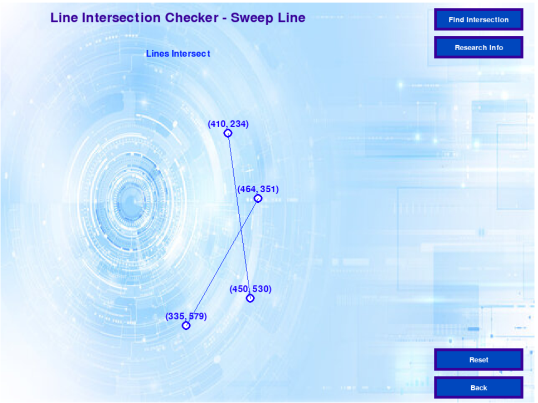
```
Execution time: 0.0001564 seconds
Time Complexity: O(nlogn) | Space Complexity: O(n)
```


Line Intersection Checker
Line Intersection Checker - Algebraic Method
Find Intersection
(228, 229)
(559, 253)
(137, 420)
(489, 479)
Reset
Back

Lines Intersect

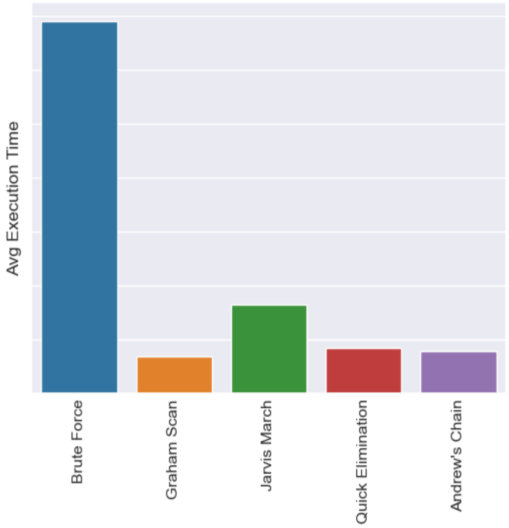


Lines Intersect



Lines Intersect

We recorded the execution time that these 5 Geometric algorithms take to compute convex hull and we come to the results below. Now here is our finding of Geometric algorithms on average execution time.



5 Conclusion

Over here we got to know that Graham Scan, Quick Elimination and Andrew's Monotone performs the best with O(nlogn) complexities. Whereas for line intersection all algorithms perform fine. In the end this project enabled us to understand the working of convex hulls and line segments which would be really handy in future studies related to AI and different tech domains.

References

Kris Min, R. (2020). A Novel Method for a Convex Hull Algorithm on Reconfigurable Hardware,CALSTATE University.

scholarworks.calstate.edu/downloads/2z10ww05b

Diane Souvaine, R. (2005). Line Intersection using Sweep Line Algorithm, TUFTS University.

www.cs.tufts.edu/comp/163/notes05/seg$_{i}ntersection_{h}andout.pdf$

$CS373Faculty, R.(2002).LineSegmentIntersection, UniversityofIll$

$jeffe.cs.illinois.edu/teaching/373/notes/x06 - sweepline.pdf$

Appendix

pip install pygame
pygame.display: Manages the display window and screen settings.
pygame.event: Handles events such as keyboard and mouse input.
pygame.image: Deals with image loading and manipulation.
pygame.mixer: Manages sound and music.
pygame.sprite: Provides a basic sprite class for game objects.
pygame.font: Allows rendering text on the screen.
pygame.time: Handles time-related functionality.
Pygame Documentation: https://www.pygame.org/docs/
Pygame Community: https://www.pygame.org/community

The brute force convex hull algorithm computes the convex hull by checking every possible combination of points. While simple, it is computationally expensive and inefficient for larger datasets. Jarvis March, also known as the Gift Wrapping algorithm, iteratively selects the point with the smallest polar angle with respect to the current point. It has a time complexity of O(nh), where n is the number of points and h is the number of points on the convex hull. The Quick Elimination algorithm is a divide-and-conquer approach. It recursively divides the set of points into two halves, computes the convex hulls of the halves, and merges them to find the overall convex hull. The Graham Scan algorithm sorts the points based on their polar angles with respect to a reference point. It then traverses the sorted points to build the convex hull efficiently. It has a time complexity of O(n log n). Andrew's Monotone Chain algorithm is an efficient convex hull algorithm that first sorts the points lexicographically and then builds the upper and lower hulls separately. It achieves a time complexity of O(n log n).