Empirical Analysis and Implementation of Divide-and-Conquer Algorithms

Closest Pair of Points and Karatsuba Integer Multiplication

Group Members: . Izza Farhat 22k-4120

CS2009: Design and Analysis of Algorithms

Abstract

In this project, we implemented two popular divide-and-conquer algorithms: the **Closest Pair of Points** and **Karatsuba Integer Multiplication** algorithms. We applied these algorithms to datasets with varying sizes and complexities, all generated randomly. To make the project user-friendly, a graphical user interface (GUI) was developed to help users easily select input files, run the algorithms, and visualize the results. The experiments we ran helped to demonstrate the efficiency and scalability of both algorithms, with some interesting insights into their performance.

Introduction

Divide-and-conquer algorithms are well-known for breaking a problem into smaller, more manageable subproblems, solving each subproblem, and then combining the solutions to get the final result. This report focuses on two specific divide-and-conquer techniques:

1. **Closest Pair of Points**: This problem is essential in computational geometry, where we need to find the closest pair of points among a set of points in a 2D plane.
2. **Karatsuba Integer Multiplication**: This algorithm provides an efficient way of multiplying large numbers, and we implement it here to compare its performance against standard multiplication.

To evaluate the performance of these algorithms under varying conditions, we developed a graphical interface that enables users to run them on chosen datasets and visualize results. The project's goal is to analyze their scalability with increasing input sizes

Proposed System

The architecture of the system consists of four major components that work together to provide a seamless experience for the user:

1. **Data Generation**: Random datasets are created and saved in text files. These datasets are used as input for both algorithms.
2. **Algorithm Implementation**: Both the Closest Pair of Points and Karatsuba Integer Multiplication algorithms are implemented in Python. The algorithms are optimized for efficiency using the divide-and-conquer approach.
3. **Graphical User Interface (GUI)**: The GUI allows users to easily select input files and visualize results, such as execution times and output data.

4. **Empirical Analysis**: Execution times for each algorithm are plotted on a graph, helping to highlight the efficiency and performance of the algorithms.

System Diagram:
At a high level, the system follows this flow:

- The user selects the input dataset through the GUI.
- The chosen algorithm is applied to the dataset.
- The program then processes the data and displays the output, including visual graphs and execution times.

Experimental Setup

Input Datasets:

● **Closest Pair of Points**: We generated datasets containing between 100 and 200 random points, each having x and y coordinates in a 2D plane. These points were generated within a range of [0, 1000]. ● **Karatsuba Multiplication**: Datasets consisted of large integers, each with 50–60 digits. These numbers were also generated randomly to test the performance of the Karatsuba algorithm.

Randomization Details:

● **Points**: Each point in the dataset was randomly generated with coordinates between 0 and 1000. ● **Integers**: The integers used for multiplication were randomly generated numbers with 50–60 digits, stored in text files for easy reuse.

The datasets were stored in separate text files, making it easy to reproduce the tests and ensure consistency across experiments.

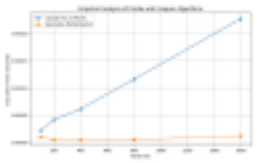
Results and Discussion

We ran both algorithms on the datasets we created and analyzed the results:

- **Closest Pair of Points**: The algorithm correctly identified the closest pair of points for each dataset. For each pair, the algorithm correctly calculated the Euclidean distance. The performance scaled efficiently as the number of points increased, and the algorithm maintained an expected time complexity of O(nlogn)O(n \log n)O(nlogn).
- **Karatsuba Integer Multiplication**: The Karatsuba algorithm successfully multiplied large integers, and the results matched the output of Python's standard multiplication operator *. The algorithm showed a noticeable improvement over the naive multiplication method, especially as the input size increased. The time complexity of Karatsuba multiplication grew sub-cubically, which was consistent with our expectations.

We also visualized the execution times for both algorithms. As the datasets grew larger, the execution times increased as well, which is typical of divide-and-conquer algorithms. However, the efficiency of both algorithms showed that they could handle relatively large inputs in a reasonable amount of time.

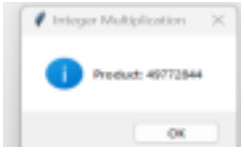
Empirical Analysis
In our empirical analysis, we observed how execution times increased with the size of the input datasets. As expected, the **Closest Pair of Points** algorithm showed an O(nlogn)O(n \log n)O(nlogn) complexity, meaning that as the number of points increased, the execution time grew logarithmically.

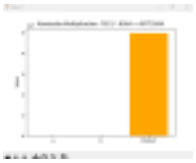


The **Karatsuba Integer Multiplication** algorithm, on the other hand, demonstrated sub-cubic growth in execution The Karatsuba Integer Multiplication algorithm showed sub-cubic growth, making it more efficient than traditional methods for large inputs. Performance graphs highlighted the efficiency of both algorithms, with Closest Pair scaling logarithmically and Karatsuba exhibiting sub-cubic behavior.

Screenshots of Program:

Sample test case execution:





Conclusion

In conclusion, this project successfully implemented the Closest Pair of Points and Karatsuba Integer Multiplication algorithms, validating their correctness and efficiency through tests on random datasets. The user-friendly interface allowed easy interaction and visualization of results, including performance graphs. The Closest Pair algorithm achieved O(n log n) efficiency, while Karatsuba demonstrated scalable performance for large numbers, making the system ideal for educational and analytical use.

References

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.
2. Karatsuba, A. (1962). *Multiplication of multidigit numbers on automata*. Soviet Physics Doklady. 3. Matplotlib Official Documentation (Accessed for graphing utilities)
4. Python Official Documentation: https://docs.python.org