Project 1

Texas Tech University

CS 3364: Design and Analysis of Algorithms

Professor: Dr. Victor Sheng

October 9, 2023

Dhruv Maniar R11713343

Atharva Dalvi R11765481

Apoorv Rana R1172307

Alex Murangira R11648884

Evaluating Search Engine Reliability through Sorting Algorithms

Problem Statement

In the realm of web search engines, assessing the reliability of different sources and ranking functions is crucial. The challenge lies in selecting the most effective search engine among three options based on their ranking of 10,000 web pages from five source files. This project aims to provide a data-driven solution to this challenge by evaluating and comparing the performance of these search engines.

The primary objective is to determine which search engine produces the most accurate and reliable rankings. To achieve this, we will rank the web pages using each search engine and then apply three sorting algorithms: Merge Sort, Quick Sort, and Insertion Sort. These algorithms will be adapted to calculate the number of inversions between each search engine's ranking and the combined ranking. Fewer inversions will indicate greater reliability, allowing us to select the optimal search engine based on empirical data and algorithmic evaluations.

This project bridges the gap between real-world search engine reliability assessment and algorithmic solutions, providing a systematic methodology for making an informed choice among the available search engines.

Methodology and Solution Overview

Our solution involves the following steps:

1. Calculating a combined rank for each web page by summing the ranks assigned by all five search engines.
2. Sorting the combined ranks in ascending order.
3. Rearranging the data in each of the five source files based on the sorted order.
4. Applying three sorting algorithms—Merge Sort, Quick Sort, and Insertion Sort—to the rearranged data to calculate the number of inversions in each file.
5. The search engine with the least number of inversions is deemed the most reliable.

Framework and Algorithms

To accomplish this, we employ three distinct sorting algorithms:

1. ***Merge Sort***: We extend the traditional Merge Sort algorithm to calculate inversions during the merging stage. Inversions are computed by subtracting the insertion position of data from the actual length of the left array (containing smaller values). If a value is smaller, it needs to be placed on the left side to ensure ascending order. Inversions at each step are recorded in an inversions list, facilitating in-depth analysis. Total inversions are calculated by summing all values in the list.

2. ***Quick Sort***: We modify the classic Quick Sort algorithm to calculate inversions. The first value serves as the pivot, and the array is divided into three subproblems: elements less than, equal to, and greater than the pivot. When an element is less than the pivot, it must cross both the 'greater than' and 'equal to' arrays, resulting in an inversion equal to the sum of both previous arrays. Similarly, when an element shares the same value as the pivot, the inversion is calculated

as the length of the 'greater than' array. Recursive calls are made to calculate inversions for all steps.

3. ***Insertion Sort***: In this method, we calculate inversions by measuring the distance an element must travel to reach its appropriate destination. If an element must be placed on the left side, we compute the inversion by considering its current position in the array and its intended position.

Experimental Test Case

To validate the effectiveness of our sorting algorithms, we conducted experimental tests on the first 10 values in File 1. Remarkably, all three algorithms consistently yielded 20 inversions. This result was cross verified through manual inspection, affirming the accuracy of our implementations.

Inversions Count

	File 1	**File 2**	**File 3**	**File 4**	**File 5**
Merge Sort	17443718	17577216	17503256	17779229	17459088
Quick Sort	17443718	17577216	17503256	17779229	17459088
Insertion Sort	17443718	17577216	17503256	17779229	17459088

Results and Conclusion

Based on the results, we conclude that Search Engine 1 is the most reliable engine since it has the fewest inversions. Among the sorting algorithms, Merge Sort stands out as the preferred choice for Web Search Engine 1. Merge Sort's divide-and-conquer approach and stable sorting nature make it highly efficient for handling web search queries with minimal inversions. We recommend the use of Search Engine 1 for future web searches due to its speed and reliability.

Future Work

In the future, we suggest further research into optimizing the sorting algorithms for even greater efficiency. Additionally, exploring additional factors that contribute to source reliability beyond inversions could lead to more robust assessments.

Teamwork

Our team collaborated effectively, with each member contributing their expertise. Some team members focused on coding, while others provided the methodology and tested the code. One team member documented the code and prepared this report. This collaborative effort ensured the timely completion of the project, and we all had the opportunity to apply the methodology to our future endeavors.