CMP2003 TERM PROJECT REPORT

-KHAIREDDINE ARBOUCH
-MUSTAFA SAID YEŞILDAĞ

GitHub Repository

December 2023

Introduction

This report outlines the implementation details and results of our CMP2003 term project, where we developed a program to analyze web server logs using both a custom hash table and std::unordered_map.

Program Output and Execution Time

```
Top 10 Most Visited Pages from Custom Hash Table:
index.html - 139201 visits
3.gif - 24001 visits
2.gif - 23590 visits
4.gif - 8014 visits
244.gif - 5147 visits
5.html - 5005 visits
4097.gif - 4874 visits
8870.jpg - 4492 visits
6733.gif - 4278 visits
8472.gif - 3843 visits
```

Screenshot 1: Run using our own hash table.


```
Top 10 Most Visited Pages from Custom Hash Table:
index.html - 139201 visits
3.gif - 24001 visits
2.gif - 23590 visits
4.gif - 8014 visits
244.gif - 5147 visits
5.html - 5005 visits
4097.gif - 4874 visits
8870.jpg - 4492 visits
6733.gif - 4278 visits
8472.gif - 3843 visits
```

Screenshot 1: Run using our own hash table.

Hash Table Implementation

- Data Structures: Our hash table was implemented using a vector of pointers to HashNode structures. Each HashNode contains a string key representing the webpage, an integer visits to count page visits, and a next pointer to form a linked list for collision resolution.

- Collision Resolution Method: We employed the chaining technique. When multiple keys hash to the same index, they are stored in a linked list at that index, allowing efficient handling of collisions.

- Hash Function: A custom hash function was designed, using a prime multiplier approach. This function calculates a hash value by iterating over each character of the key, combining it with a running total using a prime number. The result is then modulo-ed with the size of the table to ensure it fits within our hash table's range. This approach was chosen for its simplicity and effectiveness in distributing a wide range of string keys evenly across the hash table.

- Efficiency Considerations: Special attention was given to ensure that the hash function minimizes collisions and distributes keys evenly, leading to efficient data retrieval. The chaining method ensures that even when collisions occur, the impact on performance is minimized.

Finding Top 10 Most Visited Pages

- Data Structures Used: To identify the top 10 most visited pages, we utilized a max heap implemented via std::priority_queue. This structure is ideal for maintaining a dynamically sorted set of elements.

- Algorithm & Approach: The algorithm involves iterating over all entries in the hash table or unordered_map, pushing each entry (visit count and page name) into the max heap. The max heap automatically orders the entries based on visit count, placing the most visited pages at the top. After populating the heap, we extract the top 10 elements. These represent the pages with the highest visit counts. This method ensures efficient retrieval of the top visited pages, leveraging the heap's ability to maintain the highest visited counts at the top with each insertion, thus providing quick access to the most visited pages without the need to sort the entire dataset.

Conclusion

Despite extensive efforts and iterations, the results of our custom hash table and std::unordered_map do not perfectly align, indicating subtle differences in data handling or algorithmic implementation. Assistance from AI tools like GPT-4 and Bard was sought for various tasks. This discrepancy remains an intriguing aspect of our project and showcases the complexities encountered in real-world data processing tasks.