Team Tiffany Vo (TTV): Project 3 Report

Tiffany Vo: @tiffanyyvo | **Evan Stice:** @evanstice / @vanko987 | **Ethan King:** @ethanholtking

GitHub Repository | **Video Demo**

Extended and Refined Proposal

- Problem: Many different song titles use the same words or phrases, yet there is no easy way to see which of these words or phrases are the most commonly used.
- Motivation: Our motivation was to see if there were any common trends within song titles and if there were any prevalent themes within songs. We wanted to provide a way to show interesting data comparisons on such a large dataset of songs that many of us are familiar with.
- Features implemented
  - Finding the percentage of songs that use a certain word or phrase in their title.
  - Finding the top five most used words throughout all song titles and their percentage of use.
  - Search each data set for a particular song title.
- Description of data
  - The data consists of 158,352 songs with their artist and title of the song.
- Tools/Languages/APIs/Libraries used:
  - Language used: C++
  - Used standard libraries.
  - 150K Lyrics Labeled
- Data Structures/algorithms used
  - Ordered map (red/black tree)
  - Unordered map (hash table)
    - Chaining
    - Hash function based on titles ASCII character total
- Distribution of Responsibility and Roles:
  - Evan Stice: Hash table
  - Tiffany Vo: Main function and github management/cleanup
  - Ethan King: Red black tree

Analysis

- The main change we made to our project was the calculations that we chose to make with our data and the extent of the dataset used. Initially, we wanted to focus more on the lyrics of each song. However, this quickly became very resource intensive. For this project's purposes, we decided

to focus on just song titles, and perform calculations with both of our data structures from there. We also slightly modified some of the calculation functions that are performed on each data structure to accommodate the change in the dataset used.

- **Time Complexity Analysis**
  - Loading in the dataset:
    - Unordered map (Hash table) O(n*m): While the add function itself is in constant time (except in the case where the hash table capacity must be increased, then it would be O(n*m) time), since we must iterate through every item in the CSV table, loading the dataset is O(n*m) time. The n is contributed from each bucket in the array, and m is contributed from iterating through the lists in each bucket (chaining).
    - Ordered map (Red black tree) O(m * n * log(n)): n is the size of the dataset and m is the size of the longest string. The insertion function will take at worst log(n) time. This is repeated n times, thus nlog(n) time is the worst case to load the dataset. Each of these has to compare the strings to find where in the tree it should go which takes m time. This adds up to O(m * n * lon(n)).
  - Top 5 words in song titles:
    - Unordered map (Hash table) O(m*n*k): This function must traverse through the entire hash table, which contributes O(m) time for each bucket. Within each bucket, traversal also occurs due to the chaining structure of the map, which contributes O(n) time. Within each bucket, each word of the song title string must be traversed through, which contributes O(k) time.
    - Ordered map (Red black tree): O(m * n): Where n is the size of the dataset and m is the size of the longest string. Since an inorder traversal is used to visit each node (n time) and at each node the song title is decompiled into its individual words (m time), this function is repeated for each node so it becomes O(m * n) time.
  - Percentage of specific word in song titles:
    - Unordered map (Hash table) O(m*n*k): This function must traverse through the entire hash table, which contributes O(m) time for each bucket. Within each bucket, traversal also occurs due to the chaining structure of the map, which

contributes O(n) time. Within each bucket, each word of the song title string must be traversed through to check if it matches the input, which contributes O(k) time.
        - Ordered map (Red black tree) O(n * m): Where n is the size of the dataset and m is the size of the longest string. n since an inorder traversal is used to find all nodes that have a particular word and each node has to be compared with the target string to see if the word is in the song.
    - Search for a song:
        - Unordered map (Hash table) O(n*m): The initial hash function must traverse through each ASCII character of the input string, which contributes O(n) time. Additionally, while a hash key will allow you to visit a specific bucket in constant time, chaining is utilized and thus you must traverse through the list located in the bucket to find the song title, which contributes O(m) time.
        - Ordered map (Red black tree) O(m * log(n)): log(n) Since half of the tree can be eliminated since the string will either be greater than, less than, or equal to the target allowing only the left or right branch to be taken if the target and current node title are not equal the data is then compared with the target string taking O(m) complexity which is repeated all O(long(n)) times.

Reflection

Overall, this project was challenging yet rewarding. With such broad guidelines, we initially had a difficult time not only finding a usable dataset, but also deciding what our project would do that also adhered to the project's guidelines. Additionally, none of us were very familiar with Git, and this project certainly helped us become more familiar with the process of using it. If we were to start once again as a ground, we may have mapped out the structure of our program more thoroughly in the beginning. Of course, we did do initial planning and wireframing, however the actual functionality of the program was at times ambiguous. A more concrete outline could have removed some uncertainty from our work.

"For me personally, I'm very glad that I got some more experience using GitHub, despite how frustrating it got during the process. Since working collaboratively is so important, and Git is used so extensively in the workforce, I think it was important for me to get hands-on experience with it, and I hope to continue using it more in future projects." - Evan Stice

“I felt that this GitHub experience was much more enjoyable than using it in Programming 1. I also liked how we could create new branches within the repository so then each of us could work on each task that needed to be done for the project. I learned how to implement the two data structures we used and loading in data into the two structures from a csv and being able to iterate through to be able to search through the words of the songs.” - Tiffany Vo

“Throughout this project I learned the ins and outs of red black trees. I learned when and why rotations and color flips occur. I also learned more about how to work with recursion and a recursive stack while working with the pointers from one node to another. I also became more familiar with using github and how github commits, pulls/pushes, and merges work” - Ethan King

References

- GeeksforGeeks
- Red black tree visualization (University of San Francisco)
- cplusplus.com
- Clemson University
- Stack Overflow