Technical Report: Final Project
EECE 2560: Fundamentals of Engineering Algorithms

Joshua Kim
Department of Electrical and Computer Engineering
Northeastern University
kim.joshua1@northeastern.edu

December 5, 2024

Contents

1 Project Scope

The goal of this project The project's main objectives are:

- To allow an easy to view look at overall player rankings
- To let users see which players excel in different categories.
- Allow users to sort players by tournaments as well as players stats during specific time periods

The expected outcomes include a functioning command line interface with data intake through a csv file.

2 Project Plan

The overall timeline for the project is divided into phases:

- **Week 1 (October 7 - October 13)**: Define project scope, establish main project goals, begin outlining pseocode for the project as a whole
- **Week 2 (October 14 - October 20)**: Begin development, create data intake system and begin outline of the ranking algorithm.
- **Week 3 (October 21 - October 27)**: Ensure data intake functionality with different data sets, implement data sorting by dates. Begin implementation of algorithm
- **Week 4 (October 28 - November 3)**: Begin presentation, ensure ranking system functions properly
- **Week 5 (November 4 - November 10)**: Finalize the system, add user filters and continue work on presentation and final report.
- **Week 6 (November 11 - November 17)**: Fianlize all aspects of project, finish revising and cleaning up code, complete presentation and report.
- **Week 7 (November 18 - November 28)**: Final presentation, report submission, and project closure.

3 Background

The data used for this project is from the VCT circuit, a professional esports league for Valorant consisting of 4 regional leagues. The dataset was specifically obtained from Ryan Luong on Kaggle [1]. The dataset was obtained by scraping the website vlr.gg which gets its data from the official API. Similar existing solutions do exist such as the aforementioned vlr.gg or liquidpedia but my program is more focused on very specific stats over certain time periods as

well as creating a simply ranking system in order to easily identify a players performance. The data set does have some issues as individual map data is unavailable as well as some data being recorded twice necessitating filtering during data intake.

4 Data structure

The data structures used to store the data in this project are a series of 5 connected structs.

Algorithm 1: master_stat

name ← string containing the name of the player;
tournaments ←
vector of data type "tournaments", list of tournaments the player has played;
teams ← vector containing all the teams the player has played for;
stats ← data type "stats", used to store individual tournament stats;

Algorithm 2: tournaments

tname ← string containing the name of the tournament;
stages ← vector of data type "stages", list of stages in the tournament;
top_stats ← vector of data type master_stat, ranking of top players;
booloperator ==← operator overloading for == in this struct;

Algorithm 3: stages

sname ← string containing the name of the stage;
mtypes ← vector of data type "mtypes", list of matches in the stage;

Algorithm 4: mtypes

sname ← string containing the name of the match;
stats ← structure "stats", containing player stats for the match;

Algorithm 5: stats

structure of 20 specific stats, a structure is used to make each stat more easily accessible

Structures were used for this project as it made all stats easy to axis, in addition to allowing for better organization of data rather than just storing them in an individual vector.

5 Pseudocode and analysis

5.1 file_intake

Algorithm 6: File Intake: Function to open the csv file and read the raw data, time complexity of O(n)

Input: *data* ← vector of datatype vector of datatype strings
Output: integer representing status of the file intake
Open player stats file;
if *File is not open* **then**
    Print error and return 1;
end
for *each line in the data set* **do**
    row ← vector of strings to temporarily store the data;
    Split *line* by commas and push the element into "row";
    Adjust agents data to combine consecutive entries into a single cell;
    Add push row to data vector;
end
Close the file;
erase headings from data vector **Return:** 0;

5.2 data_trim

Algorithm 7: data_trim: function to organize raw data into data structures, time complexity of O(nLogn)

Input: Raw data vector,
players ← vector of type master_stats to store player data,
tournies ← vector of type tournaments to store tournament data
Output: n/a
Begin Algorithm: Data_Trim
Sort *data* alphabetically ;
Iterate through raw data ;
if *the player already has a data entry* **then**
    for *All the tournaments the player has played* **do**
        Add their overall tournament stats to the tournies vector
    end
end
Check for duplicate tournaments, stages or matches in player data;
Assign necessary non duplicate data to a temporary vector;
Push the temp vector to the player data vector;
End Algorithm: Data_Trim

5.3 num_check

Algorithm 8: num_check: Function to ask the user for a number and check that it is valid, time complexity of O(1)

Input: *range* ← int: range of possible values for the user to chose
Output: user chosen integer
Begin Algorithm: num_check
print "Select a number: ";
pos ← user input;
if *cin.fail() or pos* ≤ 0 *or pos* > *range* **then**
    clear input errors, ignore extra input;
    print "Invalid input. Try again.";
    return num_check(range);
end
return *pos* − 1;
End Algorithm: num_check

5.4 search_player

Algorithm 9: search_player: Function to search for a selected player and display their stats over a specific time period. Time complexity O(n)

Input: vector of sorted player data
Output: n/a
Begin Algorithm: search_player
Ask user what chatacter the players name starts with;
List tournaments the players has played at;
Ask user what tournament the user would like to search for;
List stages of the tournament the player has played;
Ask user what stage the user would like to search for;
List the matches of the stage the player has played in;
Ask user what match the user would like to search for;
Print stats for that match;

5.5 search_tournament

Algorithm 10: search_tournament: Function to search for a selected tournament and rank players for that tournament. Time complexity O(n)

Input: vector of tournament data
Output: n/a
Begin Algorithm: search_player
List all tournaments that occurred ;
Ask user what tournament the user would like to search for;
Sort tournament players by highest to lowest rating;
Print player name and rating from highest to lowest;

6 Conclusion

As it currently stands the project was an overall success as the full project scope was accomplished. Some challenges that were faced was that no available public API made data difficult to get meaning the program is heavily reliant on specific data formatting. If more time was allotted for this project I would add more search functionality such as specific maps as well as modifying the code and data structures in order to allow for more modularity.

7 References

[1]: https://www.kaggle.com/datasets/ryanluong1/valorant-champion-tour-2021-2023-data?resource=download