Final Project Group 101 Report

Administrative

Team name: Movie Night
Team members: Jonathan Lijewski, Noah Johnson, Rudolph Civil
GitHub URL: https://github.com/jlijewski/COP3530-Final-Project.git
Video Link: https://youtu.be/xY0GSzvqTI8

Proposal

The problem that our group decided to solve is the difficulty of discovering new highly rated movies to watch in a seamless and easy to understand way. The solution was to create a program that allows the user to create a personalized list of movies. The motivation behind deciding to tackle this problem was that attempting to find high rated movies to watch can be a difficult, tiresome, and time-consuming task. The program was created to streamline this process.

The user will be available to access a sorted database of movies based on their average rating and desired genre. The desired movies will be presented to the user starting from the highest to lowest rating, which can be added to a personalized list of movies the user is interested in. This list can also be managed. The user is able to swap movie order in the list from most interested to least, and to delete movies from the list.

The public data set that we used for the project is the datasets located on the IMDb website. One file that detailed each movie's title, genre and more, and another file that contained the ratings and votes of each movie. The entire project was done in the C++ language for reading, sorting and presenting this data.

To sort the data in this project we created a max heap structure and sorted it using heap sort and a shell sort. Both the heap and shell sort used a vector data structure to store the list of movies. A vector is also used to create the user's personal list of movies. Lastly, we used a map data structure to store the rating data input.

The responsibilities and roles of the project were done on a who was available basis. Due to us having to work around everyone's busy schedules, most of the work done was on what a

member believed what was needed to be done at the moment. All of the members contributed to the coding of the program, and the report in some way.

Analysis

When implementing the project, most of the proposed features remained the same. The core of the project still revolved around a movie sorter that found the best rated movies of a particular genre. However, some of the project had to be added to or changed in order to meet the deadline of the project.

The first major addition was the shell sort method being incorporated into the project. Originally, the proposal only mentioned a heap sort algorithm and a max heap structure to hold the movies. After learning that this would need to be compared to another algorithm or structure in order to meet the requirements of the project, we decided to find another way to sort and look at the movies. As a group we decided to create a shell sort algorithm that similarly to the heap sort algorithm, took the movies and sorted them into a list that could be viewed by the user. This was then compared to the heap sort algorithm and the time to complete each task is displayed to the user as a final method to compare the two.

The second change to the project was a removal of a particular feature of the intended project proposal. It was mentioned in the first proposal that the movies chosen by the user would be compared with their availability on various popular streaming platforms. This seemed like a good idea, but was eventually removed due to the added complexity it would bring. After researching the possibility of adding this feature, it was determined that the scope of creating a functional version of this idea would be too big and would inhibit the completion of the project.

In terms of complexity analysis, the two functions that will be measured in this project are the heap sort algorithm and the shell sort algorithm. The heap sort algorithm that was implemented in the project has a Big O complexity of O(nlogn) in the worst case, where n is the number of movies that are in the list. The Shell sort algorithm has a complexity of O(n^2) where again, n is the number of movies that are in the list. The Shell sort algorithm has a larger complexity that the heap sort algorithm, and this is supported by the fact that in the program, the time taken for the shell sort algorithm to finish is always longer than the heap sort algorithm.

Reflection

Overall, this project was not as difficult in terms of time taken or general complexity of the concepts used to implement the project however, it did present some new challenges that are important to become accustomed to. This is generally one of the first times that students participate in a group setting to complete a coding project, and this is an important feature of coding to become familiar with when looking toward the industry.

The main difference between working on a project like this individually and with a group is the communication and coordination needed to complete it. Unlike working on a personal project, each member has a unique schedule and needs to participate in the making of the project, so it becomes important to delegate certain tasks and communicate on which parts of the project are done at which times, in order to complete it efficiently. This was done fairly well by our group, but nevertheless still was a challenge, and ultimately a learning experience moving forward.

If we were to start once again as a group, the main changes that would be made would be giving ourselves more time to complete the project and a more concrete delegation of certain tasks. In terms of time, given that this project was completed during the fast paced ending of the semester, it was difficult to begin working on the project very far in advance, but the project was generally completed within a timely manner. However, the delegation of tasks is also definitely something that can be improved looking forward. All group members decided to work on the coding part of the project together, and while this did improve the time taken to complete the project, there were various confusing moments where multiple group members were working on the same task. This was not that difficult of an issue to revsolve, but is something that could be ironed out in the future with more time being dedicated to deciding who is working on what.

What We Learned

Jonathan Lijewski- Personally I learned that working with a group is less stressful than working on a project individually, and is something that I look forward to doing in the future again. I also learned that working with a group of people can bring different challenges, and with that comes new skills that must be worked on in order to be a useful and efficient member.

Noah Johnson - I've learned that being proactive is better than being reactive in group project settings. Everyone being able to step up and do work to benefit the group as a whole will always be advantageous, and I would like to be able to contribute sooner than later in future projects. Having group members to rely on is less stressful and allows for a better project experience!

Rudolph Civil - I've learned a lot about working on an interactive/ proactive coding group like this group, however, the most important thing I learned is that you need to know where you fit into a group. Sometimes you need to find the best way to help your group without hurting it, because team success is success for everyone. Also, with a team dividing the work and having trust in your teammate is very stress free.

References

Public dataset: https://www.imdb.com/interfaces/