# DATA STRUCTURES PROJECT REPORT

**May 7, 2019**

Hinan Bilal Shah
Namal Institute

# Contents

# 1 Problem Statment

Our task was to build a search engine for the given dataset from IMDB, the dataset was composed of around a million objects. Each object had three attributes: movie ID, average rating and the number of voters of each movie. Our task was to create a search engine capable enough to answer users questions efficiently.

The ultimate objective of the project was to critically analyze how theoretical concepts are applied in real time scenarios.

# 2 Data Analysis

The provided dataset consists of around a million objects of movies more precisely 922770 records, each object has three data points, movie ID, average rating and the number of voters. Thus making the whole datapoints count around 3 million. The ID number of each movie are not consecutive, they are spaced apart starting from 1 to 10 million. The provided data was pre-sorted according to the id number of each movie. In order to make the title of the movie of good use, we had to split the integer part from the string part. The number of voters section has very interesting statistics. Minimum voters count of a movie is 5 and the maximum is around 20,70,726.Furthermore, the average number of voters is 93. A thorough analysis of data before deciding the design of its storage is very useful as it assists us in making smarter decisions and in the conclusion provides us with better efficiency of our overall systems.

# 3 Performance of Differnt Algorithams and DataStructures

We are using three Data structures in the backend of our search engine. The selection of all of them is based upon the objective of examining the implementation of a poor, good and the best data structuring technique.

For this, we have selected LinkedList as the very first data structure as it would be ideal while contrasting with better and best performing techniques. The second one is Separate Chaining and the third one is my own designed extension of the separate chaining data structure, which proves to be way better than both of them in performance.

## 3.1 Linked List

In the LinkedList, the time complexity of insertion is O(1 ) as we are adding the node at the start. On the other hand, we will have to deal with O(n) complexity while searching. The only advantage of adding the node at the start would be automatic sorting of voters id in descending order otherwise rating or the number of voters will not be placed in

any order. In a nutshell, LinkedList will not assist us in making our search efficient. The only assistance we will get is we'll only traverse the single dimensional linked list up to the point when the current node has greater id than the id of movie we are searching.

## 3.2 Seperate Chaining

Another data structure is separate chaining which uses array indexes as a head of the whole linked list. We are using separate chaining by dividing all the incoming movies objects data with respect to its rating, inserting the rating of movies on the different LinkedList head. Thus making well-distributed storage of movies with respect to the rating. Separate chaining is assisting us in a design where titleID and ratings of a movie are being placed in an orderd manner. In the worst case scenario, all the movies could be placed in the same head of LinkedList but its highly unlikely after thorough data analysis. In the average case, there is a high probability that the whole data would be evenly divided into 10 chunks each chunk having all the movie objects having the same range of ratings. Thus helping us while in searching we will be able to skip many portions of data to reach directly where our user ask for. With all these advantages we are only

able to store movies with respect to their ID and their ratings in a shape. But when it comes to placing movies with respect to their voters we lag behind with the separate chaining.

## 3.3 Self Designed DS

By modifying the existing Data Structure we have been able to design a scheme in which we can place movies with respect to their title, rating and number of votes thus letting us place whole data in a sorted manner. This will help us in searching the movies as we will be able to directly jump on the location where the demanded data is being placed. This was not possible earlier now the physical cost of the searching will be improved drastically whereas insertion sort is still the same which was negligible.

# 4 head-to-head Comparison

When it comes to ranking three of them on the basis of the physical time. The newly crafted data structure proves its worth as it is customised specifically for the data we are getting after thorough data analysis. Here mostly we'll be talking about the physical time based on iterations taken by each DS, as by taking iterations as the standard of any search we can standardise our results globally.

For the very first question when we had to find the most popular movie we defined the most popular movie whose product of rating and voters is greater than every other movie. We have made sure the most popular movie is the one which has maximum voters vice versa. Linked List is taking 842785 iterations to get to the movie which was most popular whereas separate chaining took 36925 iterations but our data structure took precisely 17 iterations to declare the answer.

In the second question when we had to find the least popular movie results were as below. Linked List is taking 25505 iterations to get to the movie which was least popular whereas separate chaining took 81 iterations but our data structure took precisely 79 iterations to declare the answer.

The third question which was about finding how many movies

had the same ratings the designed algorithm ran in O(n) complexity for second and third DS. When we had to search movies above any given range algorithm was capable enough to directly jump on the portion which was demanded rather than traversing through all nodes. But as we could not sort number of voters in separate chaining the efficiency of the algorithm was very low as in the fifth question accuracy of linked list to target desired movies was below around 0 percent whereas separate chaining had accuracy of 5.3percent and our data structure had more than 99percent of accuracy on the test case of asking for movies whose voters are greater than 1000.

# 5 Conclusion

The very first lesson we learned in the implementation of the project was that if we have perfectly analysed the incoming data we can manipulate the data in any way possible way thus making its structure efficient enough for any use.