Netflix Search Engine

Data Structures and Algorithms

PROJECT REPORT

NAME: Mashal Ashfaque

BSCS 10-A

Objective:

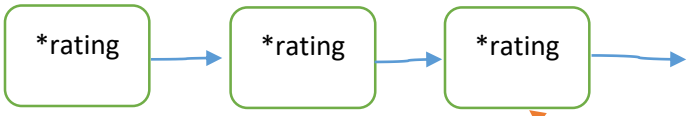
Using Netflix Data Set, stores data into structures, contains algorithms for searching through the data, with minimum time and space complexity cases in search.

Description:

This project uses the netflix dataset
First the data is parsed from the csv file in the readcsv() function in source.cpp
AVL trees are used to store the data, the avl_movies is used to create a global avl tree containing all movies and shows with their respective data.
Inside each movie/show node, there are avls of cast containing addresses of those relevant actors in the global avl of all actors present in the data set.
Same case for directors, countries, genres as for actors.
Two different type of search complexities implemented in this project:
In one type, a simple avl of countires/actors/directors is stored in each movie node, if I want to search for all movies of one specific country, then I will traverse through
the global avl of all movies, at each movie, check if the avl countries inside it contains that specific country or not, if so, then show that movie in results
This has time complexity of O(n) where n is total no. of movies/shows
Other way to reduce time complexity to O(log(n)) is implementation of a child class of avl of countries/directors etc.
This avl now contains an avl of movies inside of it as well. As line of the csv is read, the movie's address is added to global list of all movies, its countries' addresses
are added to the avl of countries inside that movie node, the country's address is also added to the global avl of all countries, and that movie's address is added to the avl of movies
inside that country node.
Now while searching for all movies of a specific country, search for that country in global avl of all countries, when found, display all movies present inside avl_movies in that
particular country.
Same case with other multi-valued attributes like directors, actors, genres.

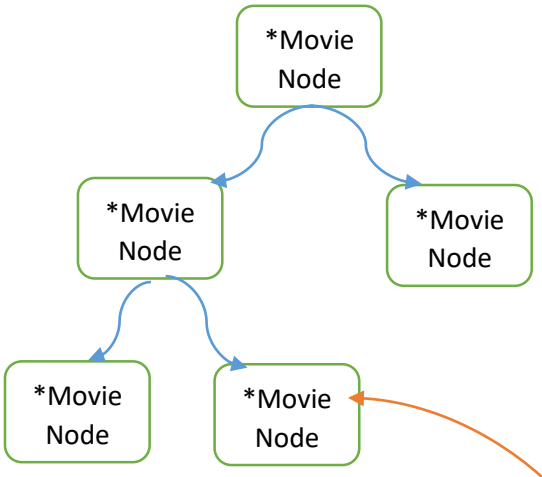
Concept map:

A Global list of all kinds of ratings



2X

A global AVL Tree of all movies:

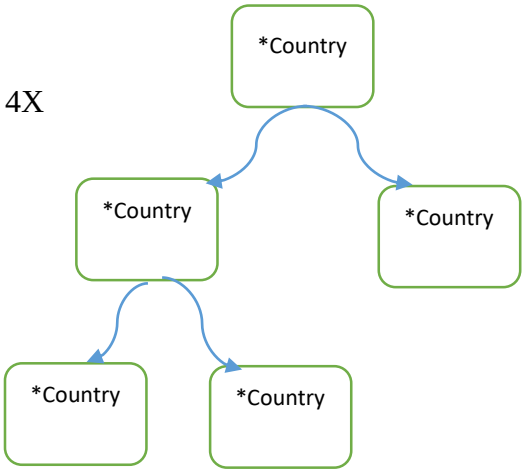


Movie Node Data:

```
String: Title,Show_id,
date_added,duration,description

bool type;

linklist_string_node*
rating,
release_year;

 AVL_strings
 countries
 directors
 actors;
 genres;
```

A global AVL Tree of all countries:

4X



Country Node:

```
String: Name


AVL_movies
 movies
```

*Implemented by utilizing concepts of forward declaration, inheritance, and polymorphism

IMPORTANT DESCRIPTION OF CONCEPT MAP

- As drawn for country, same is for directors, actors and genres too.
- As drawn for ratings, same for release years too.
- Basically, every movie contains avl tree of addresses of the country nodes, involved in that particular movie. **For e.g. movie "Cake" has an AVL tree of countries inside it, that avl tree only has two nodes: Pakistan, and United Kingdom.**
- That county node, Pakistan for e,g, is also present in a global list of all countries and this node contains an AVL of all movies of that country.
- **Like movie node "Cake" for eg. Has an avl tree of countries containing a country node's address, Pakistan. That country node , Pakistan, is present in a global AVL of countries and contains an AVL of movies like Cake, Humsafar, Jaanan etc. other movies and shows of Pakistan.**
- Exact same case for actors, directors and genres too as done for countries.

SEARCH FUNCTIONS:

Two kinds of search functions are made available for same functionality. One has Time complexity of O(n) and other is O(log(n)) where n is number of movies.

- Search movies by entering movie name
    1. Shows all data of that movie, including countries, directors, cast, etc.
- Search all movies of a particular country
- Search all movies of an actor
- Search all movies of a director
- Search movies by GENRE
- Search movies by Ratings.

In the following pages is the code for source.cpp and the screenshots. The project also has 4 other files called linkedlist.h, linkedlist.cpp, avl.h, and avl.cpp, but those are added as separate fies, and code not pasted.

Source.cpp

```
#include <string>
#include <sstream>
#include <fstream>
#include "avl.h"

//following have complexity nlog(m)  (n is no. of movies and m its no. of countries
void searchmovibytitle(movie* root, string mname);
void searchmovibycast(movie* root, string aname);
void searchmovibyrate(movie* root, string rname);
void searchmovibygenre(movie* root, string aname);

//complexity of following searches are log(n)
void searchbycountry(string cname);
void searchbydirector(string dname);

AVL_movies all_movies_in_dataset;
movie* movies_Root = NULL;

AVL_strings all_countries;
TreeNode* countries_Root = NULL;

AVL_strings all_actors;
TreeNode* actors_root = NULL;

AVL_strings all_genre;
TreeNode* genreroot = NULL;

AVL_strings alld;
TreeNode* dRoot = NULL;

linklist_string all_ratings;
linklist_string all_ry;  //Release Years

AVL_child all_countries2;
childnode* countries_Root2 = NULL;

AVL_child all_directors2;
childnode* directors_root2 = NULL;

void Readcsv()
{
        string data;
        string show_id, type, title, director, cast, country, date, release, rating, duration, genre,
desc;
        fstream netflix; //File Pointer

        netflix.open("C:\\Users\\HP\\OneDrive\\Desktop\\DSA Project\\Netflix-2\\Netflix 2 Dataset\\project
dataset\\input\\netflix-shows\\netflix_titles.csv", ios::in);
        if (netflix.fail()) {
                cout << "COULD NOT READ FILE\n";
                exit(1);
        }
        getline(netflix, data);
        int line = 0;
        while (!netflix.eof()) {
                movie* movienode = new movie();    //create a movie node, first insert all data, then
we'll add it to the avl containing this movies address
                line++;

                getline(netflix, show_id, ',');
                movienode->show_id = show_id;

                getline(netflix, type, ',');
                if (type == "Movie")
                        movienode->type = true;
                else
```

```
                        movienode->type = false;

                getline(netflix, title, ',');
                movienode->title = title;

                getline(netflix, director, ',');

                string tempd;
                istringstream ss1(director);
                ss1 >> ws;
                while (getline(ss1, tempd, '|'))
                {
                        if (tempd != "")
                        {
                                TreeNode* tc = new TreeNode();
                                tc->dataname = tempd;
                                dRoot = alld.enter(dRoot, tc);
                                alld.search(dRoot, tempd);
                                movienode->movidirectors_root = movienode->movidirectors.enter(movienode-
>movidirectors_root, alld.loc);

                                childnode* tc2 = new childnode();
                                tc2->dataname = tempd;
                                directors_root2 = all_directors2.enter(directors_root2, tc2);
                                all_directors2.search(directors_root2, tc2->dataname);

                                movie* tempmov = new movie();
                                //we cant add movienode because currently its LC,RC are NULL but wont
remain NULL afterwards
                                //so create a copy
                                tempmov->title = movienode->title; //and store only data, not the links
                                all_directors2.loc->cmroot = all_directors2.loc-
>cmovies.enter(all_directors2.loc->cmroot, tempmov); //and add it to the avl of movies of a specific
country
                        }
                }
                getline(netflix, cast, ',');
                string tempact;
                istringstream dd(cast);
                dd >> ws;
                while (getline(dd, tempact, '|'))
                {
                        if (tempact != "")
                        {
                                TreeNode* tc = new TreeNode();
                                tc->dataname = tempact;
                                actors_root = all_actors.enter(actors_root, tc);
                                all_actors.search(actors_root, tempact);
                                movienode->moviactors_root = movienode->moviactors.enter(movienode-
>moviactors_root, all_actors.loc);
                        }
                }
                getline(netflix, country, ',');
                string tempcountry;
                istringstream ss(country);
                ss >> ws;
                cout << "INDEX:\t\t" << line << endl;
                while (getline(ss, tempcountry, '|'))
                {
                        if (tempcountry != "")
                        {
                                TreeNode* tc = new TreeNode();
                                tc->dataname = tempcountry;
                                countries_Root = all_countries.enter(countries_Root, tc);
                                all_countries.search(countries_Root, tempcountry);
                                movienode->movicountries_root = movienode->movicountries.enter(movienode-
>movicountries_root, all_countries.loc);

                                childnode* tc2 = new childnode();
```

```
                                tc2->dataname = tempcountry;
                                countries_Root2 = all_countries2.enter(countries_Root2, tc2);
                                all_countries2.search(countries_Root2, tc2->dataname);

                                movie* tempmov = new movie();
                                tempmov->title = movienode->title;
                                all_countries2.loc->cmroot = all_countries2.loc-
>cmovies.enter(all_countries2.loc->cmroot, tempmov);
                        }
                }

                getline(netflix, date, ',');
                movienode->date_added = date;

                getline(netflix, release, ',');
                all_ry.Insertwithoutduplic(release);
                all_ry.search(release);
                movienode->release_year = all_ry.loc;

                getline(netflix, rating, ',');
                all_ratings.Insertwithoutduplic(rating);
                all_ratings.search(rating);
                movienode->rate_movie = all_ratings.loc;

                getline(netflix, duration, ',');
                movienode->duration = duration;

                getline(netflix, genre, ',');
                string tempg;
                istringstream gg(genre);
                gg >> ws;
                while (getline(gg, tempg, '|'))
                {
                        if (tempg != "")
                        {
                                TreeNode* tc = new TreeNode();
                                tc->dataname = tempg;
                                genreroot = all_genre.enter(genreroot, tc);
                                all_genre.search(genreroot, tempg);

                                movienode->groot = movienode->genres.enter(movienode->groot,
all_genre.loc);
                        }
                }
                getline(netflix, desc);
                movienode->description = desc;
                //save the movie node's address to the global list of  movies
                movies_Root = all_movies_in_dataset.enter(movies_Root, movienode);
        }
        cout << "\nFile parsed succesfully.\n" << line << " entries have been read and saved.\n";
        netflix.close();
}
```

```
INDEX:          6231
INDEX:          6232
INDEX:          6233
INDEX:          6234
INDEX:          6235

File parsed succesfully.
6235 entries have been read and saved.
```

```
int main() {

	cout << "Welcome to our Netflix Search program" << endl;
	Readcsv();
	while (1) {
		cout << "\nSelect an option how you want to search for movies and shows.\n";
		cout << "1-Search by name(All Details).\n2-Search by country.\n3-Search by directors.\n4-
Search by ratings(All Details).\n5-Search by cast(All Details).\n6-Show all\n7-By Genres\n";
```

```
Select an option how you want to search for movies and shows.
1-Search by name(All Details).
2-Search by country.
3-Search by directors.
4-Search by ratings(All Details).
5-Search by cast(All Details).
6-Show all
7-By Genres
```

```
		int choice; cin >> choice; string search;
		switch (choice) {
		case 1: {
			getchar(); cout << "Enter movie name.\n";
			getline(cin, search);
			searchmovibytitle(movies_Root, search);
```

(See movie.printmoviedata() in avl.h to see how all data members are printed. The multivalued like countries etc. are printed using avl.inOrder())

```
1
Enter movie name.
Stranger Things
80057281
2019
TV Show
Stranger Things
Directors:
Actors:
Caleb McLaughlin
Cara Buono
Charlie Heaton
Dacre Montgomery
David Harbour
Finn Wolfhard
Gaten Matarazzo
Joe Keery
Matthew Modine
Millie Bobby Brown
Natalia Dyer
Noah Schnapp
Paul Reiser
Sadie Sink
Sean Astin
Winona Ryder
Ratings:TV-14
Countries:
United States
Genres:
TV Horror
TV Mysteries
TV Sci-Fi & Fantasy
Duration: 3 Seasons
Description: "When a young boy vanishes, a small town uncovers a mystery involving secret experiments, terrifying supernatural forces and one strange little girl
```

```
        break;
}
case 2: {
        getchar();
        cout << "Enter country name.\n";
        getline(cin, search);
        cout << "Movies of " << search << endl;
        searchbycountry(search);
        break;
```

```
2
Enter country name.
Pakistan
Movies of Pakistan
7 Din Mohabbat In
Abdullah|The Final Witness
Balu Mahi
Cake
Chalay Thay Saath
Chupan Chupai
Dukhtar
Ho Mann Jahaan
Humsafar
Janaan
Khaani
Moor
Pari
Pinky Memsaab
Rangreza
Saawan
Sadqay Tumhare
Teefa in Trouble
Virsa
Waar
Wrong No.
Zindagi Gulzar Hai
Zindagi Kitni Haseen Hay
```

```
}


case 3: {
        getchar();
        cout << "Enter  directors.\n";
        getline(cin, search);
        cout << "Movies of " << search << endl;
        searchbydirector(search);
        break;
}
```

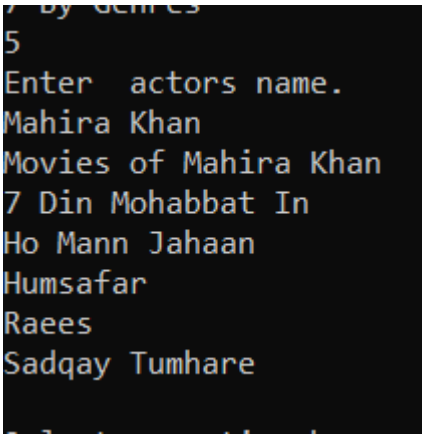
```
3
Enter  directors.
Stacie Passon
Movies of Stacie Passon
Concussion
We Have Always Lived in the Castle
```

```
case 4: {
        getchar();
        cout << "Enter  rating.\n";
        getline(cin, search);
        cout << "Movies of " << search << endl;
        searchmovibyrate(movies_Root, search);
        break;
}
```

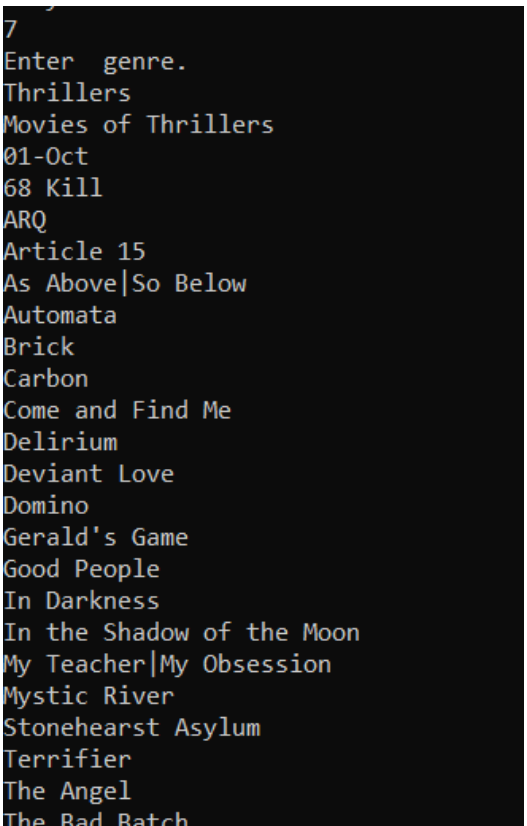
```
7-By Genres
4
Enter  rating.
PG-13
Movies of PG-13
10|000 B.C.
16 Blocks
20 Feet From Stardom
21
3 Generations
3 Idiots
5 Flights Up
9
A Murder in the Park
A Night at the Roxbury
A Separation
A Tale of Love and Darkness
About a Boy
Across the Universe
After
Always Be My Maybe
Anger Management
Ant-Man and the Wasp
Apollo 18
Arthur
As Good as It Gets
Austin Powers in Goldmember
Austin Powers: International Man of Mystery
Austin Powers: The Spy Who Shagged Me
Avengers: Infinity War
Barely Lethal
Bathtubs Over Broadway
Bean: The Ultimate Disaster Movie
Beavis and Butt-head Do America
Bebe's Kids
Becoming Jane
Before I Wake
Benchwarmers 2: Breaking Balls
Black Panther
Blackfish
Blackway
Blue Jasmine
Brain on Fire
Bring It On: Worldwide Showdown
Burlesque
Carriers
Catch Me If You Can
```

```
C:\Users\HP\source\repos\Project202\Debug\Project202
Catch Me If You Can
Center Stage
Chappaquiddick
Charlie's Angels
Charlie's Angels: Full Throttle
Cirque du Freak: The Vampire's Assistant
Clash of the Titans
Click
Cloverfield
Coach Carter
Code Name: The Cleaner
Crouching Tiger|Hidden Dragon
Dante's Peak
Die Another Day
Dinner for Schmucks
Doctor Zhivago
Doubt
Dovlatov
Dragonheart
Dragonheart 3: The Sorcerer
Dragonheart: Battle for the Heartfire
Dumb and Dumberer: When Harry Met Lloyd
Dumplin'
Eat Pray Love
Echo in the Canyon
Emo the Musical
Employee of the Month
Enter the Warriors Gate
Equals
Evolution
Ferris Bueller's Day Off
Flash of Genius
Generation Iron 2
Get Smart
Getting Played
Ghost Rider
Ginger & Rosa
GoldenEye
Grand-Daddy Day Care
HALO Legends
Happy as Lazzaro
He Named Me Malala
Heartbreakers
Hellboy
Hitch
Honey 2
Honey: Rise Up and Dance
Hot Rod
How to Be a Latin Lover
How to Make an American Quilt
```

```
case 5: {
        getchar();
        cout << "Enter  actors name.\n";
        getline(cin, search);
        cout << "Movies of " << search << endl;
        searchmovibycast(movies_Root, search);
        break;
}
```




```
case 6: {
        all_movies_in_dataset.inorder(movies_Root, 2);
        break;
}
case 7: {
        getchar();
        cout << "Enter  genre.\n";
        getline(cin, search);
        cout << "Movies of " << search << endl;
        searchmovibygenre(movies_Root, search);
        break;
}
```



```
                }
        }
        system("Pause");
}

void searchmovibycast(movie* root, string aname)
{
        //traverse all movies in order
        if (root) {
                searchmovibycast(root->lc, aname);
                //search the AVL Tree of directrs inside the movie node (movidirectors)
                root->moviactors.search(root->moviactors_root, aname);
                if (root->moviactors.loc != NULL)
                {
                        root->printmoviedata();
                }
                searchmovibycast(root->rc, aname);
        }
}
void searchmovibytitle(movie* root, string mname)
{
        all_movies_in_dataset.loc = root;
        if (root == NULL) {
                all_movies_in_dataset.loc = NULL;
        }
        else {
                if (root->title == mname)
                {
                        root->printmoviedata();
                }
                else if (root->title > mname)
                {
                        searchmovibytitle(root->lc, mname);
                }
                else if (root->title < mname)
                {
                        searchmovibytitle(root->rc, mname);
                }
        }
}

void searchmovibyrate(movie* root, string rname)
{
        //traverse all movies in order
        if (root) {
                searchmovibyrate(root->lc, rname);
                if (root->rate_movie->r == rname)
                {
                        root->printmoviedata();
                }
                searchmovibyrate(root->rc, rname);
        }
}

void searchmovibygenre(movie* root, string aname)
{
        //traverse all movies in order
        if (root) {
                searchmovibygenre(root->lc, aname);
                //search the AVL Tree of directrs inside the movie node (movidirectors)
                root->genres.search(root->groot, aname);
                if (root->genres.loc != NULL)
                {
                        root->printmoviedata();
                }
                searchmovibygenre(root->rc, aname);
        }
}
```

```
void searchbycountry(string cname)
{
        all_countries2.search(countries_Root2, cname);
        if (all_countries2.loc) {
                all_countries2.loc->cmovies.inorder(all_countries2.loc->cmroot, 1);
        }
}

void searchbydirector(string cname)
{
        all_directors2.search(directors_root2, cname);
        if (all_directors2.loc) {
                all_directors2.loc->cmovies.inorder(all_directors2.loc->cmroot, 1);
        }
}
```