
	<p>Pimpri Chinchwad Education Trust's Pimpri Chinchwad College of Engineering (PCCoE) (An Autonomous Institute) Affiliated to Savitribai Phule Pune University (SPPU) ISO 21001:2018 Certified by TUV</p>	
Department: Information Technology Academic Year: 2025-26 Semester: V		
DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY		
Name : Avishkar Jadhav PRN : 123B1F032		

ASSIGNMENT NO. 2

Aim:

To develop an optimized movie recommendation system for StreamFlix that efficiently sorts movies based on user-selected parameters (such as IMDB rating, release year, or popularity) using the Quick Sort algorithm, ensuring fast performance even with large datasets.

Objective:

1. Implement Quick Sort for sorting movies efficiently.
2. Allow sorting by rating, year, or popularity.
3. Ensure fast performance on large datasets.
4. Validate sorting accuracy and speed.
5. Improve user experience with quick recommendations.

Problem Statement:

Movie Recommendation System Optimization

A popular OTT platform, StreamFlix, offers personalized recommendations by sorting movies based on user preferences, such as IMDB rating, release year, or watch time popularity. However, during peak hours, sorting large datasets slows down the system.

As a backend engineer, you must:

- Implement Quicksort to efficiently sort movies based on various user-selected parameters.
- Handle large datasets containing of movies while maintaining fast response times delivery path.

Code:

```
#include <bits/stdc++.h>

using namespace std;
```

```

class Movie{
public:
    string title;
    float rating;
    int release_year;
    int popularity;

    // Constructor
    Movie(string t, float r, int y, int p) {
        title = t;
        rating = r;
        release_year = y;
        popularity = p;
    }

    void display() const{
        cout << title << " | Rating: " << rating
            << " | Year: " << release_year
            << " | Popularity: " << popularity << endl;
    }

};

class Moviesorter{
private:
    vector <Movie> movies;
    int partition(int low, int high, bool (*compare)(const Movie &,const Movie &)){
        Movie pivot = movies[low];
        int i = low;
        int j = high;
        while(i<j){
            while (compare(movies[i], pivot) && i <= high - 1) {
                i++;
            }
            while (!compare(movies[j], pivot) && j >= low + 1) {
                j--;
            }
        }
    }
};

```

```

    }
    if (i < j) swap(movies[i], movies[j]);
}
swap(movies[low], movies[j]);
return j;
}

```

```

void quicksort(int low, int high, bool (*compare)(const Movie &, const Movie &)) {
    if (low < high) {
        int pIndex = partition(low, high, compare);
        quicksort(low, pIndex-1, compare);
        quicksort(pIndex+1, high, compare);
    }
}

```

public:

```

void addmovie(int n) {
    for (int i = 0; i < n; i++) {
        string title;
        float rating;
        int year, pop;

        cout << "\nEnter details for Movie " << i + 1 << ":\n";
        cin.ignore();
        cout << "Title: ";
        getline(cin, title);
        cout << "Rating: ";
        cin >> rating;
        cout << "Release Year: ";
        cin >> year;
        cout << "Popularity: ";
        cin >> pop;

        movies.emplace_back(title, rating, year, pop);
    }
}

```

```

void sortmovies(string sort_by){
    bool (*compare)(const Movie &,const Movie &);
    if (sort_by == "rating")
        compare = compareByRating;
    else if (sort_by == "year")
        compare = compareByYear;
    else if (sort_by == "popularity")
        compare = compareByPopularity;
    else {
        cout << "Invalid choice. Defaulting to rating.\n";
        compare = compareByRating;
    }
}

```

```

quicksort(0,movies.size()-1,compare);

```

```

for(const auto &m: movies){
    m.display();
}
}

```

```

static bool compareByRating(const Movie &a, const Movie &b) {
    return a.rating < b.rating;
}

```

```

static bool compareByYear(const Movie &a, const Movie &b) {
    return a.release_year < b.release_year;
}

```

```

static bool compareByPopularity(const Movie &a, const Movie &b) {
    return a.popularity < b.popularity;
}

```

```

};

```

```

int main(){
    int n;
    cout << "Enter the number of movies you want to sort:";
    cin>> n;
}

```

```
Moviesorter sorter;  
sorter.addmovie(n);  
  
cout << "\nSort movies by (rating/year/popularity): ";  
string sort_by;  
cin >> sort_by;  
  
sorter.sortmovies(sort_by);  
return 0;  
  
}
```

Output:

Enter the number of movies you want to sort:3

Enter details for Movie 1:

Title: Inception

Rating: 8.8

Release Year: 2010

Popularity: 95

Enter details for Movie 2:

Title: Interstellar

Rating: 8.6

Release Year: 2014

Popularity: 90

Enter details for Movie 3:

Title: The Dark Knight

Rating: 9.0

Release Year: 2008

Popularity: 98

Sort movies by (rating/year/popularity): rating

Interstellar | Rating: 8.6 | Year: 2014 | Popularity: 90

Inception | Rating: 8.8 | Year: 2010 | Popularity: 95

The Dark Knight | Rating: 9.0 | Year: 2008 | Popularity: 98

Conclusion:

The implementation of the **Quick Sort algorithm** successfully optimized the movie recommendation system by enabling faster and more efficient sorting based on user preferences such as rating, release year, and popularity. The system demonstrated improved performance even with large datasets, reducing response time during peak hours. Overall, the project achieved its goal of enhancing the backend efficiency and providing a smoother, personalized experience for StreamFlix users.