2024-2025 FALL

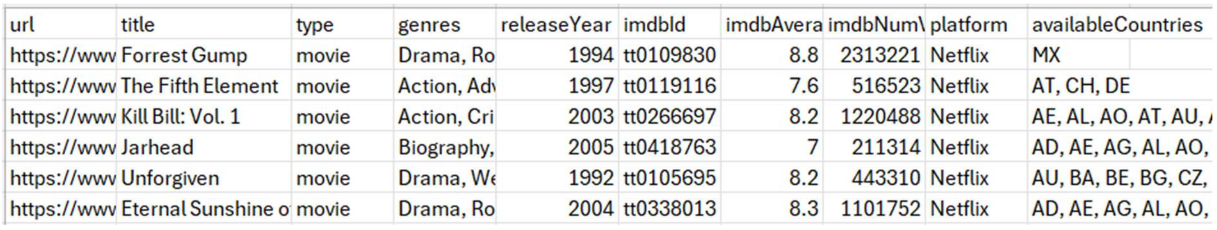
CME 2201 - Assignment 1

MEDIA ARCHIVE MANAGEMENT SYSTEM BY USING HASH TABLES

The media and entertainment industry handles massive volumes of media content from production houses and film studios to television networks and streaming platforms. Media archive management involves the systematic organization, storage, and retrieval of various types of media assets. Media archive management aims to ensure that media assets are securely stored, easily accessible, and can be efficiently repurposed for future use and monetization. In this project, you are expected to develop a software system that allows users to catalog their film stock and search for movies quickly and efficiently.

1. Dataset

The dataset provides a comprehensive collection of all titles (Movies and TV Series) available on popular streaming services, including Netflix, Amazon Prime, Apple TV+, Hulu, and HBO Max. In addition to basic information such as URL, Title, Type, Genres, Release Year, Platform, and Available Countries, it includes IMDb-specific data like IMDb ID, Average Rating, and Number of Votes. The dataset consists of 93,584 total records of 74,532 unique media content. A snapshot of the dataset can be seen in Figure 1.



url	title	type	genres	releaseYear	imdbId	imdbAvera	imdbNum\	platform	availableCountries
https://www	Forrest Gump	movie	Drama, Ro	1994	tt0109830	8.8	2313221	Netflix	MX
https://www	The Fifth Element	movie	Action, Adv	1997	tt0119116	7.6	516523	Netflix	AT, CH, DE
https://www	Kill Bill: Vol. 1	movie	Action, Cri	2003	tt0266697	8.2	1220488	Netflix	AE, AL, AO, AT, AU,
https://www	Jarhead	movie	Biography,	2005	tt0418763	7	211314	Netflix	AD, AE, AG, AL, AO,
https://www	Unforgiven	movie	Drama, We	1992	tt0105695	8.2	443310	Netflix	AU, BA, BE, BG, CZ,
https://www	Eternal Sunshine o	movie	Drama, Ro	2004	tt0338013	8.3	1101752	Netflix	AD, AE, AG, AL, AO,

Figure 1. Sample data from movie dataset

2. Data Structure

In this project, you must use a hash table to organize the media archive. You will use your own hash table implementation in Java programming language. The aim is to access specified media records rapidly.

Initially, you will read the whole dataset and store all media information in a hash table. The hash table structure should look like as shown in Figure 2. You should use a list data structure for each media item to store the platforms containing it. You can also store media-enabled country codes and media genres in the sorted list data structure.

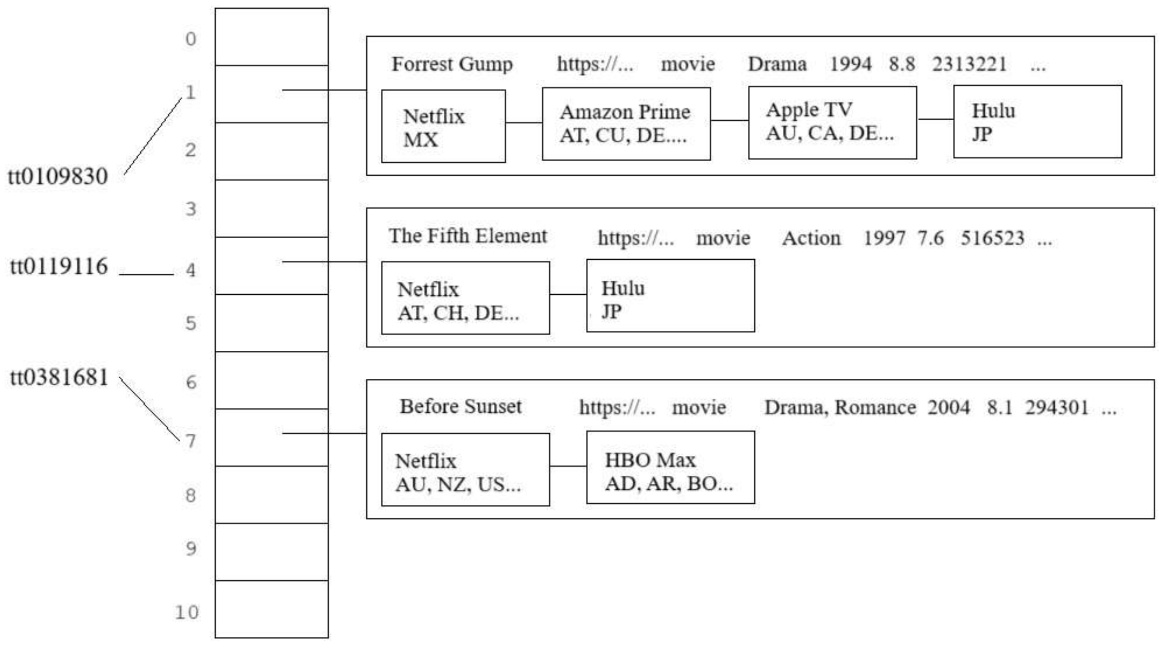

0
1
2
3
4
5
6
7
8
9
10
tt0109830
tt0119116
tt0381681
Forrest Gump https://... movie Drama 1994 8.8 2313221 ...
Netflix MX
Amazon Prime AT, CU, DE....
Apple TV AU, CA, DE...
Hulu JP
The Fifth Element https://... movie Action 1997 7.6 516523 ...
Netflix AT, CH, DE...
Hulu JP
Before Sunset https://... movie Drama, Romance 2004 8.1 294301 ...
Netflix AU, NZ, US...
HBO Max AD, AR, BO...

Figure 2. An illustration of the hash table

3. Functionalities

- **add(Key k, Value v)**

If a media is already present in the hash table, add the new platform information to the media platform list; otherwise, create a new hash table entry. ImdbId of media items must be used as keys for the hash table entries.

- **Value get(Key k)**

Search for the given media id in the hash table. If the media is available in the table, then return an output as shown in Figure 3; otherwise, return a "not found" message to the user.

```
>Search: tt0109830

Type: Movie
Genre: Drama, Romance
Release Year: 1994
IMDb ID: tt0109830
Rating: 8.8
Number of Votes: 2313221

4 platforms found for Forrest Gump

Netflix - MX
Amazon Prime - AT, CU, DE, IN, JP, SN
Apple TV+ - AU, CA, DE, US
Hulu - JP
```

```
> Search: tt6105885

Media not found!
```

Figure 3. Sample queries on the hash table

- **remove(Key k)**

Remove the given media id and the associated value from the hash table.

- **resize(int capacity)**

Make the hash table dynamically growable. *The add* method should double the current table size if the hash table reaches the maximum load factor.

4. Hash Function

First, you should generate an integer hash code using a special function to specify an index corresponding to a given media id. Then, the resulting hash code must be converted to the range 0 to N-1 using a compression function, such as the modulus operator (N is the size of the hash table).

You are expected to implement two hash functions: the simple summation function and the polynomial accumulation function.

4.1.Simple Summation Function (SSF)

You can generate the hash code of a string *s* with length *n* simply by the following formula:

$$\mathrm{h(s)} = \sum_{k=0}^{n-1} ch_k$$

4.2.Polynomial Accumulation Function (PAF)

The hash code of a string *s* can also be generated by using the following polynomial:

$$\mathrm{h(s)} = ch_0 * z^{n-1} + ch_1 * z^{n-2} + \ldots + ch_{n-2} * z^1 + ch_{n-1} * z^0$$

where ch_0 is the leftmost character of the string, characters are represented as numbers in 1-26 (case insensitive), and *n* is the length of the string. The constant *z* is usually a prime number (33, 37, 39, and 41 are particularly good choices). When the *z* value is chosen as 33, the string "car" has the following hash value:

$$h(\mathrm{car}) = 3 * 33^2 + 1 * 33 + 18 * 1 = 3318$$

Note: Using this calculation on the long strings will result in numbers that will cause overflow. You should ignore overflows or use Horner's rule to calculate and apply the modulus operator after computing each expression in Horner's rule.

2024-2025 FALL

CME 2201 - Assignment 1

5. Collision Handling

5.1. **Linear Probing (LP)**

Linear probing handles collisions by placing the colliding item in the next (circularly) available table cell.

5.2. **Double Hashing (DH)**

Double hashing uses a secondary hash function *d(k)* and handles collisions by placing an item in the first available cell of the series.

$$d(k) = q - k \bmod q$$

$$h_2(k) = \big(h(k) + j\, d(k)\big) \bmod N$$

where $q < N$ (table size), q is a prime, and $j = 0, 1, \ldots, N-1$.

The secondary hash function *d(k)* cannot have zero values. The table size N must be a prime to allow probing of all the cells.

Example:

N = 13,
k= 31,
q= 7,
h(k) = k mod 13 = 5,
d(k) = 7 - k mod 7 = 4.
	The 1st lookup index: 5
	The 2nd lookup index: 5+ 1*4 = 9 % 13 = 9
	The 3rd lookup index: 5+2*4 = 13 % 13 = 0
	…

6. Performance Monitoring

You are expected to fill the performance matrix (Table 1) by running your code under different conditions, including two different load factors (50% and 80%) to decide on resizing the hash table, two different hash functions (SSF and PAF), and two different collision handling techniques (LP and DH).

You should count the total number of collision occurrences and measure expended time while loading media information into the hash table under each condition. In addition, you should calculate avg. search times using the “search.txt” file containing 1000 media ids to search (Search time means the time expended to find a particular key in the hash table. It does not include the time spent on outputs. To calculate avg. search time, divide the total expended time by the total number of searches). In the table, you should provide time measures with the same time units (ns, ms, s, etc.) and right-align numbers.

Load Factor	**Hash Function**	**Collision Handling**	**Collision Count**	**Indexing Time**	**Avg. Search Time**
α=50%	SSF	LP			
		DH			
	PAF	LP			
		DH			
α=80%	SSF	LP			
		DH			
	PAF	LP			
		DH			

Table 1. Performance matrix

7. Program Menu

You should provide a user-friendly Program Menu for easy use. The menu should include

1. Load dataset
2. Run 1000 search test
3. Search for a media item with the ImdbId.
4. List the top 10 media according to user votes
5. List all the media streams in a given country
6. List the media items that are streaming on all 5 platforms

8. Provided Resources

- Movie dataset (movies_dataset.csv).
- 1000 media ids for search operations ("search.txt")

9. Project Groups

You can develop the project individually or create a group of up to two students. In both cases, you must enter your group information into the file linked below.

⚠️ PROJECT GROUPS

10. Due date

05.12.2024 Thursday 23:55. Late submissions are not allowed. The due date will not be extended.

11. Requirements

- Usage of Java programming language and Abstract Data Types (ADT) are required.
- Object Oriented Programming (OOP) principles must be applied.
- Exception handling must be used when it is needed.

12. Submission

You must prepare a project report describing your data structure, java code, and performance matrix. Your report should contain a cover page specifying group members.

You must upload your all ‘.java’ files and report as an archive file (.zip or .rar). Your archived file should be named as ‘studentnumber_name_surname.rar.zip’ (e.g., 2007510011_Ali_Yılmaz.rar or 2007510011_Ali_Yılmaz_2007510012_Aras_Aydin.rar) and should be uploaded via **online.deu.edu.tr**.

13. Code Control

Control of the project will be on **06 of December** in laboratory sessions. A schedule will be announced before the code control date. You must be in the laboratory on time. You will have 10 minutes to show your assignment. Please do not forget to bring your laptops while coming to the assignment control!

14. Plagiarism Control

The submissions will be checked for code similarity. Copy assignments will be graded as zero.

15. Grading Policy

Job	**Percentage**
Usage of ADT, OOP and Try-Catch	%30
Hash table implementation	%50
Performance monitoring	%20

GOOD LUCK!