COS 226

Hash something out

20pts

Due: Nov 25, 2025, 11:59pm

1 Assignment Description

For this homework, you will implement two hash tables to store fake movie records from a large input dataset. Your goal is to create efficient hash tables that minimize wasted space and minimize collisions through careful design of your hash functions.

You will need to create a GitHub repository to track your optimization attempts, showing at least 5 different variations of your hash functions with commit descriptions explaining each optimization.

2 Hash Table Requirements

You must create **two separate hash tables**:

2.1 Hash Table 1: Movie Title as Key

This hash table will use the movie title as the key to store and retrieve movie records.

2.2 Hash Table 2: Movie Quote as Key

This hash table will use the movie quote as the key to store and retrieve movie records.

3 Optimization Goals

Your hash functions should be designed to:

- **Minimize wasted space**: Design your hash table size and hash function to use space efficiently
- **Minimize collisions**: Design your hash function to distribute keys evenly across the hash table buckets

4 Statistics Tracking

Your program must track and display the following statistics for **each** hash table:

- Amount of wasted space (unused buckets or slots)

- Number of collisions that occurred during construction
- Time taken to construct the hash table

Display these statistics after constructing each hash table so you can compare the performance of different hash function implementations.

5 GitHub Repository Requirements

You must create a **public GitHub repository** for this project with the following requirements:

5.1 Version Control

- Create a GitHub repository for this project
- Make at least **5 commits** showing different optimization attempts
- Each commit must have a descriptive commit message and comment explaining:
  - What optimization you tried
  - Why you chose that approach
  - What the results were (statistics)
  - Any other relevant information
- The repository must be public so it can be accessed for grading

5.2 Optimization Attempts

For each of your 5 optimization attempts, you should:

- Modify your hash function(s)
- Run your program and record the statistics
- Commit the changes with a descriptive message
- Take a screenshot of the results showing the statistics

5.3 Hash Function Implementation Requirements

Your 5 optimization attempts must use **fundamentally different approaches** to hash function design. Simply modifying numbers or parameters in the same hash function does not count as a different approach. Each attempt should represent a distinct hashing strategy or methodology. Examples of fundamentally different approaches might include:

- Using different mathematical operations or algorithms

- Changing the fundamental structure of how the hash is computed
- Using different collision resolution strategies
- Applying different preprocessing techniques to the input keys

Important: New attempts do not need to be "better" than previous ones—they just need to be fundamentally different. The goal is to explore different approaches, not necessarily to improve performance with each attempt.

Your commit messages should clearly explain how each approach is fundamentally different from the previous attempts. Additionally, you must include a **reflection** in your `README.md` file that discusses how well each method worked, analyzing the statistics you collected and comparing the performance of your different hash function approaches.

6 Implementation Guidelines

6.1 Data Loading

- Load the movie data from the provided input file
- Parse the data to extract movie titles and quotes
- Handle any edge cases in the data (empty fields, special characters, etc.)

7 Submission Requirements

You must submit the following:

7.1 GitHub Repository

- A link to your **public GitHub repository**
- The repository must contain all your code files
- The repository must show your commit history with at least 5 optimization attempts

7.2 Screenshots

- Screenshots showing the results (statistics) of each of your 5 hash function variations
- Each screenshot should clearly show:
  - Which hash table (title or quote)
  - The statistics (wasted space, collisions, construction time)
  - Which optimization attempt this represents
- Include screenshots for both hash tables for each optimization (10 screenshots total, or combined screenshots showing both tables)

7.3 Reflection

- A written reflection that discusses how well each of your 5 hash function methods worked
- Analyze the statistics you collected for each approach
- Compare the performance of your different hash function approaches
- Discuss which methods were most effective and why
- The reflection must be included in your GitHub repository as a `README.md` file

8 Additional Requirements

- Include comprehensive comments at the top of your program explaining your approach
- Include comments explaining your hash function implementations
- Handle edge cases appropriately
- Ensure your code is well-organized and readable
- Test your implementation thoroughly