ECE242 PROJECT 2: Hash Tables (The Last One!)

Due: December 4, 2014, 11PM on Moodle

Introduction

With the success of Web 2.0 and the popularity of online social networks, huge amount of data, such as Facebook posts, Web pages, Twitter tweets, and Amazon sales records, are being collected. You may notice that a term "big data" has become very popular in recent years. As the amount of data increases, the ability to use simple arrays and linked lists for storage is limiting due to the inefficiency of searching data. This is especially true for character strings which are often long and varied in content. As a result, hash tables are a preferred way of finding objects which contain character strings. Of course, as you have seen in class, the effectiveness of the hash table depends on the chosen hash function.

Task Overview

For this final project, you will revisit the simple book database you designed for Project 1 and make it more efficient. In Project 1, you stored books in an array. In this project, you are going to use a hash table to store books.

You will have the chance to implement a hash table and to develop a hash function. The data set you will use is the file with 10,000 books ("BX-Books.csv") used for Project 1 (you can reuse a lot of the file reading code from that project). Each line in the file contains all features of a book: ISBN, book title, author, year of publication, publisher and 3 image links for book cover. Like Project 1, all features of a book can be stored in a *Book* object.

To store all the 10,000 books, you will build a hash table which contains 2,000 entries. Note that you must write your own hash table class (Do not use the built-in Hashtable interface in Java). *Book* objects will be assigned to a specific entry in the hash table using a hash function that you will write. You should use the book title as the input to the hash function.

Since there are 10,000 books and only 2,000 entries in the hash table, multiple books will be assigned to the same entry. To accommodate these collisions, each entry in the hash table will contain a reference to a linked list. Each linked list will hold *Book* objects. After you have read in the book file and stored it in the hash table, you will read in a second file ("titles.csv") which contains 5 book titles. Your code will search for each of these titles in the hash table and print out the stored features about the book. In addition, you will need to update some feature of a book in the hash table.

More specifically, you will perform the following tasks.

1. Build a hash table of 2,000 entries to store 10,000 books.

2. Write a hash function which will convert a book title (string) into an integer between 0 and 1999. This integer will be used to find a specific entry in the hash table.

3. For each book, add a *Book* object (which has all the features of the book) to the selected linked list.

4. When you search for a specific book from the "titles.csv" file, you will use the hash function and the book title to locate the appropriate linked list. Then, you can iterate over the list using the book title string to match the appropriate *Book* object.

5. Update the ISBN of the book with title "Classical Mythology" to be 0199997322.

Getting started

The best way to complete any programming assignment (including this one) is to take tasks step-by-step. The first decision you need to make is the number of classes you will write. For this assignment, you should have at least three classes, including the Book class, hash table class, and a driver class. Please do not place all the methods for your entire project in one class.

Required Output

1. Print out the features (including ISBN, book title, author, year of publication and publisher) of each book for the 5 books listed in the "titles.csv" file.

2. Print out the number of entries of the hash table used by your program to store the 10,000 books. In other words, print out the number of nonempty entries, and a nonempty entry stores at least one book.

3. Print out the size of the longest linked list in your hash table (e.g. the linked list with the most books).

4. Print out the features (ISBN, book title, author, year of publication and publisher) of book "Classical Mythology" after you update its ISBN to be 0199997322.

Hints and suggestions

Successfully completing the project and achieving a good grade requires completing the

project as described above and clearly commenting the code. As always, it makes sense to start the project early. Build your project code step by step.

- Feel free to use the *Book* class and book file reading code from Project 1. Also, you can use the built-in LinkedList in Java. You might want to put the file reading code in the hash table class. For the code of reading the "titles.csv" file, you can implement it in the hash table class, the driver class, or a new separate class.

- You may have realized that separate chaining is the collision resolution technique in this project. Basically, the hash table is an array of linked lists. Each element of a linked list is a *Book* object. In the lecture, we have shown an example of hash table with separate chaining. You may borrow some ideas from this example.

What to submit:

Please submit all .java files for your project and the screenshots for all the printout results. All code should be well commented.

Reminder: The course honesty policy requires you to write all code yourself, except for the code that we give to you (or allow you to reuse). Your submitted code will be compared with all other submitted code for the course to identify similarities. Note that our checking program is not confused by changed variable or method names.

Grading

- Code works (50%)
- Comments (20%)
- Program structure (20%)
- Readability (10%)