Name: H.S.R. Dharmasena
Index : 110123N

CS 2022 – Mini project

Project Report

a) Main data structures and algorithms use in the program

- Data Structures
  - Hash Table Data Structure
  - Heap Data Structure
  - Modified Circular Stack Data Structure

- Algorithms
  - Heap Sort Algorithm
  - Hash Searching Algorithm
  - Hash code generating Algorithm

b) Reasons for choosing above Data Structures

Hash Table Data Structure

The main part of the program is the searching (Book and Vendor searching). So by using Hash Table searching can be done in constant time O (1+α).

If I used linked list instead of hash table space efficiency is better than hash table. But the main part of the program is searching products. If I used linked list, search should be linear searching O (n). So the hash table is better in searching. Also if I used linked list structure if size is bigger it takes memory to store pointers.

If I used simple array for this purpose it also take linear search time and space efficiency is also reduced.

Heap Data Structure

- Program is required top rated vendors of product. So by using heap data structure heap sorting can be done in O (nlogn) time (quickly).
- Same running time complexity for best, average, and worst case in a heap is same i.e. n logn.
- Time Stamp should be sorted to find most recent ratings for the product. So using heap sort it can be done easily and quickly.
- Heap sort does not need additional memory to sort like merge sort.
- Heap Data Structure is easy to implement (no recursive functions so can save stack spaces). If I used quick sort recursive functions exists and stack spaces get full.

Circular Stack Data Structure

This data structure is used for find 5 most recent ratings for the product. Every record is added to the stack and stack size is 5. So after stack is full new record is added to the beginning of stack and so on. So the finally items of the stack are the most recent ratings.

c) Calculate the running time of algorithms

Algorithm	Running time
Heap Sort	O (nlogn)
Hash Searching	O (1+α)

There are several methods running under the HeapSort method i.e. Max-Heapify takes O (logn) time, Build-Heap takes O (n) time. So the HeapSort takes O (n*logn) time

In Hash Searching algorithm time required to execute every line is a constant time. If collisions happen it will takes α additional time. So the running time of the hash searching is O (1+ α)

d) Problem I have faced

- Writing a algorithm to find 5 most recent ratings for the product/vendor
  First I get each record in file to a heap data structure and sort it by timestamp value (ascending order).Then I created circular stack (of 5 elements) and push each sorted record to the stack. So finally all the 5 most recent ratings are in stack.

- Find a algorithm to calculate hash code

  I used similar algorithm of java hash code method which is take string's each character and multiply it by 31^(n-i) and add it.

  There are several advantages by multiplying 31. 31 is prime odd number and 31*I can be done quickly.

  31*I = (I << 5) – I;

- When calculating overall aggregate rate of product/vendor

  To calculate aggregate rate I have to find users who have rated that product/vendor, number of times that user rate that product/vendor (k), sum of ratings of user to relevant product/vendor (r), number of time that user has rated to a product (n). So I create arrays to keep user, k, r in each book/vendor.

- When finding top rated vendors of the product

  Heap sort is used.

- Sorting time stamp

  Time stamp is a string value. How to sort it? I take long integer and put time Stamp value to that integer. Ex. 2013-01-15T22:45 -> 201301152245. It can be also done by using Date class in java. But I thought previous one is easier than that.

- If book/vendor does not exist in Hash Table null value will be return and program exit.

  I put bookSearch() and vendorSearch() methods under try, catch block. So the nullPointerException occurs it is caught and thrown.

- Creating user, book, and vendor objects and initialize each object , put them to their Hash tables

  This is the biggest problem I have faced. There are 8 cases to consider when adding objects. I took each case separately. So for an example if user and book doesn't exist and vendor exists, get the vendor from vendor hash table and put user and book to their hash tables.

User	Book	Vendor
X	X	X
X	X	✓
X	✓	X
X	✓	✓
✓	X	X
✓	X	✓
✓	✓	X
✓	✓	✓

X - Doesn't exist in Hash Table

✓ - exists in Hash Table

- Listing 5 most recent rating products

  Cannot put products object to circular stack. If same product rated again instance of object is changed. I created UserDetail class; put relevant information push UserDetail object to the circular stack.

e) Future improvements

- Make graphical user interface (GUI) for the program so can be used easily.
- Use open addressing for the Hash Table.
- Use linked list data structure to keep books in Vendor class so the memory spaces can be saved.
- There are several types of Hash Tables (User, Book, and Vendor) I used new class for each type of HashTable. It is better to make one generic HashTable class to several type of objects.
- I assumed that hash table is never get full. Improve code to handle fully Hash Table.

References

- CS2022 – lecture notes
- http://en.wikipedia.org/wiki/Hash_table
- http://stackoverflow.com/questions/299304/why-does-javas-hashcode-in-string-use-31-as-a-multiplier