BITS Pilani
Pilani | Dubai | Goa | Hyderabad
Work Integrated Learning Programmes

DESIGN DOCUMENT

(S1-18_DSEABZG519)
(Data Structures and Algorithms Design)
First Semester, 2018 -19
Assignment 1 –PS4-[StudentRecord] -1

Team Members -(TUSHAR GOYAL, SMRUTI RANJAN MANOHAR, AMIT KUMAR SHARMA, VIVEK KUMAR TIWARI)

Indexes

Brief problem statement

In this problem, we should write an application in JAVA that keeps track of student records in a university.

At a university, there is a need to store all the details of graduating students. For this exercise, let us consider that the CGPA of the student is stored against the student id.

The students ID has the following format <YYYYAAADDDD> where

YYYY - represents the year in which this student joined the university

AAA - a three letter (alphabet) representing degree program

DDDD - a four-digit number representing the students roll number

For instance, an ID can be of the form 2008CSE1223 corresponding to a student joined the university in the year 2008 in the CSE department with the roll number 1223. The university offers a 4-year graduate degree program in CSE (Computer Science and Engineering), MEC (Mechanical Engineering), ECE (Electronics and Communication Engineering) and ARC (Architecture). In the year 2008 the first batch of 20 students were admitted to the university. Now in the year 2018, 200 students were admitted across all departments. Create an input file input.txt with a random list of students per year and their corresponding CGPA.

The university now wants to use the details of all its past students to

1. Identify and commemorate their alumni on the 10th year anniversary of the University. For this they will need to get a list of all students who scored over x CGPA.
2. Extend a new course offering to selected students who have graduated in the past five years. For this they will need to get a list of all students who secured between CGPA x to CGPA y in the past five years.
3. Identify the maximum and average CGPA per department.

- **Design a hash table**, which uses student Id as the key to hash elements into the hash table. Generate necessary hash table definitions needed and provide a design document (2 page) detailing clearly the design and the details of considerations while making this design.

- **Design a hash function HashId()** which accepts the student-ID as a parameter and returns the hash value. You are only allowed to use basic arithmetic and logic operations in implementing this hash function. Write as a comment to this function, the reasons for the specific choice of hash function.

- **Create / populate the hash table from the list of student ID and the corresponding CGPA given in the input file**.

Design pre-considerations

1. The hash table does not have feature to remove elements, as it is not included into problem statement.
2. The hash table initial capacity is 32 by default, which get increased by a factor of 2 while resizing.

Design Strategy

The hash-table design has following three major components.

1) Hashcode Generation

We need a hash function that guarantees that the probabilities that larger numbers of hash values collide is the same as what one would get with a random function.
So we have chosen to use a random polynomial hash function, h, which is defined as below.

$$h(k) = ad + k(ad\text{-}1 + k(ad\text{-}2 + \cdots + k(a3 + k(a2 + ka1)) \cdots))$$

Where a1, a2 …a_d are the d elements of the key and k is a prime number(Which we have chosen as 31).

2) Compression Technique

To get an index in a range of the table capacity, we need a compression technique. We have chosen following compression technique to randomly distribute the keys.

$$c(k) = ((\textit{COMPRESSION\_A} * \text{hashCode}) + \textit{COMPRESSION\_B}) \bmod N$$

COMPRESSION_A = is a prime number (3 has been chosen)
COMPRESSION_B = another prime number (269)
N = capacity of the hash table.

3) Collision Resolution Technique

We have selected **separate chaining** as the preferred collision resolution technique. However, the implementation has been done using both **Open Addressing (linear probing)** and **separate chaining**. Which can be customized in the code by changing a value in the code.

Advantages & Disadvantages of Separate Chaining

Advantages	Disadvantages
Simple to implement.	
Hash table never fills up, we can always add more elements to chain	Cache performance of chaining is not good as keys are stored using linked list. Open addressing provides better cache performance as everything is stored in same table
Less sensitive to the hash function or load factors	Wastage of Space (Some Parts of hash table are never used)
It is mostly used when it is unknown how many and how frequently keys may be inserted or deleted	if the chain becomes long, then search time can become O(n) in worst case
Mostly the performance of separate chaining is competitive and sometimes better to open addressing schemes.	Uses extra space for links.

Performance Analysis of separate chaining

m = Number of slots in hash table

n = Number of keys to be inserted in hash table

Load factor α = n/m

Expected time to search = O(1 + α)

Expected time to insert/delete = O(1 + α)

So if we keep the load factor to 0.75 its O(1).

Program Prerequisites

1) The program uses **maven** as a dependency and build tool.
2) Java version - 1.8

Implemented Methods

StudentRecords.java

1) *public void initializeHash(StudentHash records)*
2) *public void insertStudentRec(StudentHash records, String studentId, float cgpa)*
3) *public void hallOfFame(StudentHash records, float CGPA)*
4) *public void newCourseList(StudentHash records, float cgpaFrom, float cgpaTo)*
5) *public void depAvg(StudentHash records)*
6) *public void destroyHash(StudentHash records)*

How to run the code

Prerequisite: Maven and Java 1.8

Step 1: Download the code and go to the root of the project using a cmd.

Step 2: run→ **mvn install**

Step 3: To run the Junit tests run -> **mvn test**

Step 4: Go to the target folder and run StudentRecords.class as **java StudentsRecords**.
Note : The program expects the input.txt file to be present in the current location.

Another way is to import the project as a maven project in eclipse and then use eclipse maven plugin to build, test and run the project.

You can also clone the code from here.

End