Middle East Technical University | Department of Computer Engineering

CENG 351

Data Management and File Structures

Fall 2019-2020

Programming Assignment 2

Due date: 05.01.2020, Sunday, 23:55

1 Lab Entry System

In this programming assignment, your task is to create a lab entry system that will be used to keep track of students that entered our laboratories. You will use extendible hashing structure to store the students. For detailed description, please refer to **10.2 EXTENDIBLE HASHING chapter in Database Management Systems** (by Raghu Ramakrishnan and Johannes Gehrke) book. Although extendible hashing in the lecture notes uses the first k bits, we will use a different flavor (the last k bits) in this assignment.

The lab entry system is summarized as follows:

- Students enter the laboratory by showing their ID cards to our lab entrance system.
- When a student enters the laboratory, we store his/her studentID (i.e., "e1234567").
- StudentIDs are stored in extendible hashing structure.
- When a student leaves the laboratory, we remove the related studentID from the system.

2 The Work

Only four functions and a constructor in the file LabDB.java are needed to be written. You are allowed to add public private members. In addition you are free to add additional classes.

2.1 public LabDB(int bucketSize)

Bucket size is defined as the number of entries that a bucket can store. Initialize an extendible hashing structure whose **global depth** (length of common hash **suffix** of bucket address table) is **1** and and bucket size is given as input parameter.

2.2 public void enter(String studentID)

In this function, store the studentID in the extendible hashing structure.

- The studentID is in the form e{numeric part}. (i.e., e1234567)
- While locating the studentIDs, use suffix of binary equivalent of the {numeric part} of the studentIDs but store the whole studentID(i.e, e1234567).

2.3 public void leave(String studentID)

Remove the given studentID from the extendible hashing structure. After removing the studentID please check the following conditions:

- If a bucket is empty and its **local depth** (length of common hash **suffix** of data bucket) is same as its buddy bucket's (which is referred as "split image" in the textbook) local depth, then merge the bucket with its buddy bucket and decrease its local depth.
- If all of the buckets' local depth are less then the global depth, then halve the directory and reduce the global depth.

2.4 public String search(String studentID)

Search the given studentID in the extendible hashing structure. If the student exists, then return the bucket address (hash suffix) of the bucket that contains the student. If the given studentId does not exist, then return "-1". Output samples for this function are given in the Examples section.

2.5 public void printLab()

Print the extendible hashing structure to the screen. Output samples for this function are given in the Examples section. Since the grading will be black-box, you should **strictly** obey the sample output.

3 Examples

- In the examples given below, ␣ represents a single **space** character.
- Suppose that bucket size is given as 4 initially. (Please note that, I may test different bucket sizes while evaluating your code.)

printLab()
Global␣depth␣:␣1
0␣:␣[Local␣depth:1]
1␣:␣[Local␣depth:1]

- The student e4 enters the lab.

printLab()
Global␣depth␣:␣1
0␣:␣[Local␣depth:1]<e4>
1␣:␣[Local␣depth:1]

- The students enter the lab with the following order: e12, e32, e16, e1, e5, e21, e10, e15, e7, e19.

printLab()
Global␣depth␣:␣2
00␣:␣[Local␣depth:2]<e4><e12><e32><e16>
01␣:␣[Local␣depth:2]<e1><e5><e21>
10␣:␣[Local␣depth:2]<e10>
11␣:␣[Local␣depth:2]<e15><e7><e19>

- Search for student e21.

System.out.println(labdb.search("e21"))
01

- Student e13 enters the lab.

printLab()
Global␣depth␣:␣2
00␣:␣[Local␣depth:2]<e4><e12><e32><e16>
01␣:␣[Local␣depth:2]<e1><e5><e21><e13>
10␣:␣[Local␣depth:2]<e10>
11␣:␣[Local␣depth:2]<e15><e7><e19>

- Student e20 enters the lab.

printLab()
Global␣depth␣:␣3
000␣:␣[Local␣depth:3]<e32><e16>
001␣:␣[Local␣depth:2]<e1><e5><e21><e13>
010␣:␣[Local␣depth:2]<e10>
011␣:␣[Local␣depth:2]<e15><e7><e19>
100␣:␣[Local␣depth:3]<e4><e12><e20>
101␣:␣[Local␣depth:2]<e1><e5><e21><e13>
110␣:␣[Local␣depth:2]<e10>
111␣:␣[Local␣depth:2]<e15><e7><e19>

- Student e9 enters the lab.

printLab()
Global␣depth␣:␣3
000␣:␣[Local␣depth:3]<e32><e16>
001␣:␣[Local␣depth:3]<e1><e9>
010␣:␣[Local␣depth:2]<e10>
011␣:␣[Local␣depth:2]<e15><e7><e19>
100␣:␣[Local␣depth:3]<e4><e12><e20>
101␣:␣[Local␣depth:3]<e5><e21><e13>
110␣:␣[Local␣depth:2]<e10>
111␣:␣[Local␣depth:2]<e15><e7><e19>

- Search for non-existent student.

System.out.println(labdb.search("e101010"))
-1

- Search for student e16.

System.out.println(labdb.search("e16"))
000

- Student e32 leaves the lab.

printLab()
Global␣depth␣:␣3
000␣:␣[Local␣depth:3]<e16>
001␣:␣[Local␣depth:3]<e1><e9>
010␣:␣[Local␣depth:2]<e10>
011␣:␣[Local␣depth:2]<e15><e7><e19>
100␣:␣[Local␣depth:3]<e4><e12><e20>
101␣:␣[Local␣depth:3]<e5><e21><e13>
110␣:␣[Local␣depth:2]<e10>
111␣:␣[Local␣depth:2]<e15><e7><e19>

- Student e16 leaves the lab.
  NOTE: Bucket 000 becomes empty. Its local depth and its buddy bucket's(100) local depth are both 3. After deleting e16, merge 000 and its buddy bucket (100) and decrease the local depth.

printLab()
Global␣depth␣:␣3
000␣:␣[Local␣depth:2]<e4><e12><e20>
001␣:␣[Local␣depth:3]<e1><e9>
010␣:␣[Local␣depth:2]<e10>
011␣:␣[Local␣depth:2]<e15><e7><e19>
100␣:␣[Local␣depth:2]<e4><e12><e20>
101␣:␣[Local␣depth:3]<e5><e21><e13>
110␣:␣[Local␣depth:2]<e10>
111␣:␣[Local␣depth:2]<e15><e7><e19>

- Student e10 leaves the lab.
  NOTE: Bucket ␣10 becomes empty. Its local depth and its buddy bucket's(␣00) local depth are both 2. After deleting e10, merge ␣10 and its buddy bucket(␣00) and decrease the local depth.

printLab()
Global␣depth␣:␣3
000␣:␣[Local␣depth:1]<e4><e12><e20>
001␣:␣[Local␣depth:3]<e1><e9>
010␣:␣[Local␣depth:1]<e4><e12><e20>
011␣:␣[Local␣depth:2]<e15><e7><e19>
100␣:␣[Local␣depth:1]<e4><e12><e20>
101␣:␣[Local␣depth:3]<e5><e21><e13>
110␣:␣[Local␣depth:1]<e4><e12><e20>
111␣:␣[Local␣depth:2]<e15><e7><e19>

- Student e9 leaves the lab.

printLab()
Global depth : 3
000 : [Local depth:1]<e4><e12><e20>
001 : [Local depth:3]<e1>
010 : [Local depth:1]<e4><e12><e20>
011 : [Local depth:2]<e15><e7><e19>
100 : [Local depth:1]<e4><e12><e20>
101 : [Local depth:3]<e5><e21><e13>
110 : [Local depth:1]<e4><e12><e20>
111 : [Local depth:2]<e15><e7><e19>

- Student e1 leaves the lab.
  NOTE: Bucket 001 becomes empty. Its local depth and its buddy bucket's(101) local depth are both 3. After deleting e1, merge 001 and its buddy bucket(101) and decrease the local depth.
  NOTE2: After decreasing the local depth, all of the buckets' local depth are less then the global depth. So we halve the directory and reduce the global depth.

printLab()
Global depth : 2
00 : [Local depth:1]<e4><e12><e20>
01 : [Local depth:2]<e5><e21><e13>
10 : [Local depth:1]<e4><e12><e20>
11 : [Local depth:2]<e15><e7><e19>

- Students e4, e12 and e20 leave the lab respectively.

printLab()
Global depth : 2
00 : [Local depth:1]
01 : [Local depth:2]<e5><e21><e13>
10 : [Local depth:1]
11 : [Local depth:2]<e15><e7><e19>

- Students e5 and e21 leave the lab respectively.

printLab()
Global depth : 2
00 : [Local depth:1]
01 : [Local depth:2]<e13>
10 : [Local depth:1]
11 : [Local depth:2]<e15><e7><e19>

- Student e13 leaves the lab.
  NOTE: Bucket 01 becomes empty. Its local depth and its buddy bucket's(11) local depth are both 2. After deleting e13, merge 01 and its buddy bucket(11) and decrease the local depth.
  NOTE2: After decreasing the local depth, all of the buckets' local depth are less then the global depth. So we halve the directory and reduce the global depth.

printLab()
Global depth : 1
0 : [Local depth:1]
1 : [Local depth:1]<e15><e7><e19>

4 Regulations

1. Programming Language: Java.
2. Late Submission: Late submission policy is stated in the course syllabus.
3. Cheating: We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.
4. Newsgroup: You must follow Odtu Class for discussions and possible updates on a daily basis.

5 Submission

Submission will be done via OdtuClass. Create a compressed file named **ceng.zip** that contains **LabDB** class and all other classes, created by you. You will **not** submit Evaluate.java. The compressed file should contain a directory tree same as the package tree. That is, you should compress the directory named 'ceng' which contains a directory named 'ceng351' which contains a directory named 'labdb' which contains your source files.

```
ceng
└── ceng351
    └── labdb
        ├── LabDB.java
        ├── AnotherClassIfYouNeed1.java
        ├── AnotherClassIfYouNeed2.java
        ├── ..
        ├── ...
        └── AnotherClassIfYouNeedN.java
```

6 Useful Links

- Java Documentation:
  http://docs.oracle.com/javase/tutorial/java/index.html