Data Structures (Spring 2020)

Open Hashing (6th Lab)

2020.04.24

Seoul National University

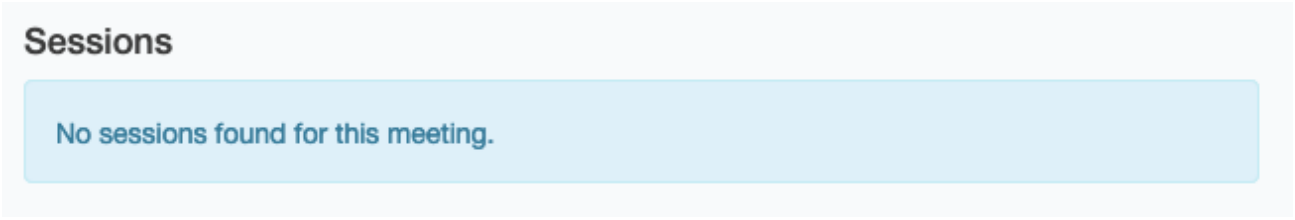
Database Systems Lab

Today's Lab

- Announcement
  - Attendance
- Open Hashing
  - Implementation

Announcement

- Attendance
  - There are some issues about attendance at eTL
  - According to the university, eTL and Zoom are not well-synchronizing each other
  - We will upload new instructions as soon as possible



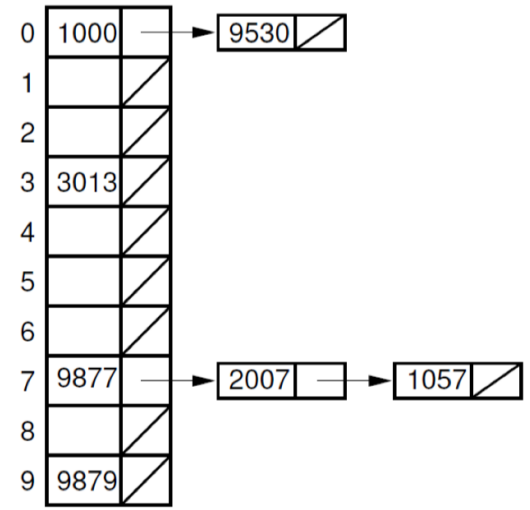
... 🤷

Open Hashing

- Hashing
  - Hashing is a popular technique to store and search key values in a table.

- *Open Hashing* is one of hashings
  - Each slot (or entry) of the hash table is the head of a linked list.
  - A collision is resolved by chaining an overflow entry.
  - Also known as chained hashing.

Open Hashing -->



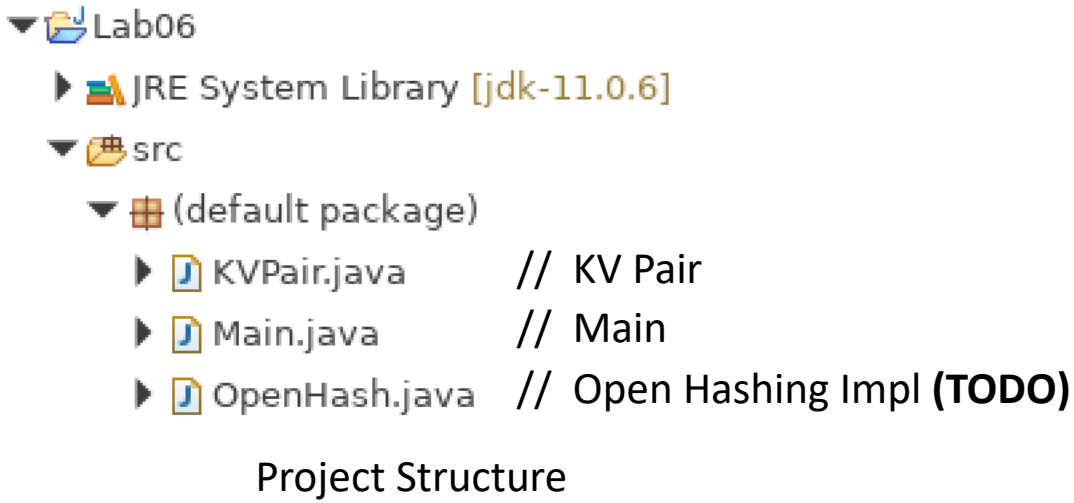
From: Bongki Moon, "Lecture Notes on Data Structures: Hash"

Exercises

- Implement Open Hashing
  - int size()
    - Return the number of entry (without overflowed ones)
  - int h(Integer key)
    - Hash function (key mod size)
  - boolean insert(Integer k, E r)
    - Insert value r with key k
    - If duplicate key found then return false, else return true
  - E search(Integer k)
    - Return value of key k
    - If not found, return null
  - E remove(Integer k)
    - Remove key-value pair and return value of key k
    - If not found, return null

Exercises

- Fill the blank of codes
  - Write your code into "// TODO:" section (`OpenHash.java`)
  - `Implement size(), h(), insert(), search(), remove()`



Project Structure

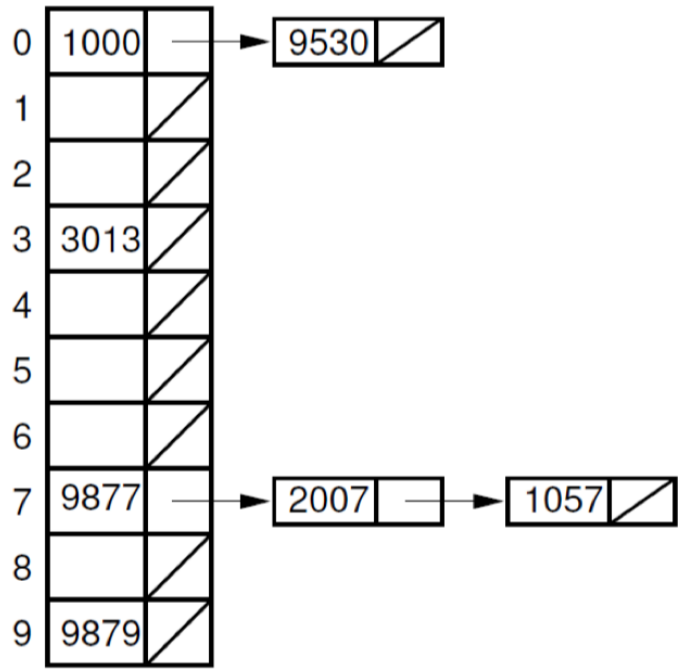
Main.java -->

```
public static void main(String[] args) {
    int size = Integer.parseInt(args[0]);
    OpenHash<Integer, String> T = new OpenHash<Integer, String>(size);
    System.out.println("Hash table size: " + T.size());

    // Insert test
    int[] keys1 = {1000, 9530, 3013, 9877, 2007, 1057, 9879, 1000};
    String[] vals1 = {"A", "B", "C", "D", "E", "F", "G", "H"};
    for (int i = 0; i < keys1.length; i++) {
        boolean res = T.insert(keys1[i], vals1[i]);
        if (res) System.out.println("inserted: <" + keys1[i] + ", " + vals1[i] + ">");
        else System.out.println("insert: <" + keys1[i] + ", " + vals1[i] + "> Failed!");
    }

    // Search test
    int[] keys2 = {1000, 9530, 2007, 2000};
    for (int i = 0; i < keys2.length; i++) {
        String res = T.search(keys2[i]);
        if (res != null) System.out.println("search: " + keys2[i] + " -> " + res);
        else System.out.println("search: " + keys2[i] + ": Not Found!");
    }

    // Delete test
    int[] keys3 = {2007, 1057, 9877, 2007};
    for (int i = 0; i < keys3.length; i++) {
        String res = T.remove(keys3[i]);
        if (res != null) System.out.println("delete: " + keys3[i] + " -> " + res);
        else System.out.println("delete: " + keys3[i] + ": Not Found!");
    }

    // Last search
    String res = T.search(2007);
    if (res != null) System.out.println("search: 2007 -> " + res);
    else System.out.println("search: 2007: Not Found!");
}
```

Exercises

- Result

```
$ java Main 10
Hash table size: 10
inserted: <1000, A>
inserted: <9530, B>
inserted: <3013, C>
inserted: <9877, D>
inserted: <2007, E>
inserted: <1057, F>
inserted: <9879, G>
insert: <1000, H> Failed!
search: 1000 -> A
search: 9530 -> B
search: 2007 -> E
search: 2000: Not Found!
delete: 2007 -> E
delete: 1057 -> F
delete: 9877 -> D
delete: 2007: Not Found!
search: 2007: Not Found!
```



+ Additional Problem

How can we perform the insertion operation with worst-case execution time O(1)?
It can be a different result with respect to the same input.