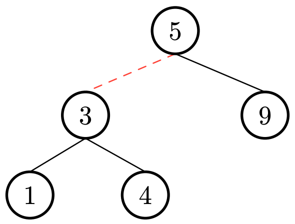
CS 61B
Spring 2025

B-Trees, LLRBs, Hashing
Exam-Level 07: March 10, 2025

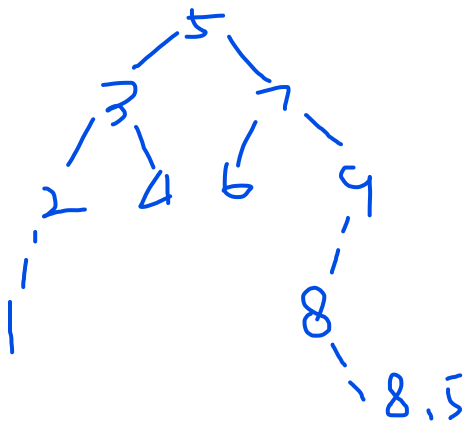
1 LLRB Insertions

Given the LLRB below, perform the following insertions and draw the final state of the LLRB. In addition, for each insertion, write the balancing operations needed in the correct order (**rotateRight**, **rotateLeft**, or **colorFlip**). If no balancing operations are needed, write "Nothing". Assume that the link between 5 and 3 is red and all other links are black at the start.



(a) 1. Insert 7

nothing



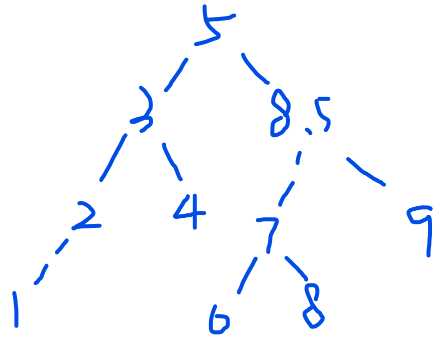
2. Insert 6
rotateright(9);
colorflip(7);
colorflip(5);

3. Insert 2

rotateleft(1)

4. Insert 8

nothing



5. Insert 8.5

rotateleft(8);
rotateright(9);
colorflip(8.5);
rotateleft(7);

6. Final state

(b) Convert the final LLRB to its corresponding 2-3 Tree.

2 Hashing Gone Crazy

For this question, use the following **TA** class for reference.

```
public class TA {
    int semester;
    String name;
    TA(String name, int semester) {
        this.name = name;
        this.semester = semester;
    }
    @Override
    public boolean equals(Object o) {
        TA other = (TA) o;
        return other.name.charAt(0) == this.name.charAt(0);
    }
    @Override
    public int hashCode() { return semester; }
}
```

Assume that the **ECHashMap** is a **HashMap** implemented with **external chaining** as depicted in lecture. The **ECHashMap** instance begins with 4 buckets.

Resizing Behavior If an insertion causes the load factor to reach or exceed 1, we resize by doubling the number of buckets. During resizing, we traverse the linked list that correspond to bucket 0 to rehash items one by one, and then traverse bucket 1, bucket 2, and so on. Duplicates are **not** checked when rehashing into new buckets.

Noah.seme=22;
Noah.name=Noah;

jasmine.sem=12;
jasmine.name=jasmin

Draw the contents of map after the executing the insertions below:

```
ECHashMap<TA, Integer> map = new ECHashMap<>();
TA jasmine = new TA("Jasmine the GOAT", 10);
TA noah = new TA("Noah", 20);
map.put(jasmine, 1);
map.put(noah, 2);

noah.semester += 2;
map.put(noah, 3);

jasmine.name = "Nasmine";
map.put(noah, 4);

jasmine.semester += 2;
map.put(jasmine, 5);

jasmine.name = "Jasmine";
TA cheeseguy = new TA("Sam", 24);
map.put(cheeseguy, 6);
```

0 cheeseguy:6 n2 j5

1

2 j1 n4

3

refer to offical solusion

4 jasmine:5 -

5

6 Noah:4 ->

7

8

3 Buggy Hash

The following classes may contain a bug in one of its methods. Identify those errors and briefly explain why they are incorrect and in which situations would the bug cause problems.

(a) The **Timezone** class below:

```
class Timezone {
    String timeZone; // "PST", "EST" etc.
    boolean dayLight;
    String location;
    ...
    public int currentTime() {
        // return the current time in that time zone
    }

    public int hashCode() {
        return currentTime();
    }

    public boolean equals(Object o) {
        Timezone tz = (Timezone) o;
        return tz.timeZone.equals(timeZone);
    }
}
```

the hashcode is not corresponding
the equals method

(b) The **Course** class below:

```
class Course {
    int courseCode;
    int yearOffered;
    String[] staff;
    ...
    public int hashCode() {
        return yearOffered + courseCode;
    }

    public boolean equals(Object o) {
        Course c = (Course) o;
        return c.courseCode == courseCode;
    }
}
```

the hashcode is not corresponding to the equals method