Neal Noble  
IT333  
Hashtables  
April 2016

Answer the following questions by implementing the code samples and/or answering the questions in a word document. Upload all project files and word documents zipped to the exercise dropbox.

Read the following article on how to write a correct equals() method: [article (Links to an external site.)](https://www.artima.com/lejava/articles/equality.html). In particular focus on pitfall #4, which highlights how an equals method should mirror a equivalence relation.

1. Given input {4371, 1323, 6173, 4199, 4344, 9679, 1989}, a hash table of length 10 and a hash function h(x) = x mod 10, show the resulting:

a) Separate chaining hash table

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 6 | 8 | 9 |
|  | 4371 |  | 1323 6173 | 4344 |  |  |  |  | 4199 9679 1989 |

b) Hash table using linear probing

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 6 | 8 | 9 |
| 9679 | 4371 | 1989 | 1323 | 6173 | 4344 |  |  |  | 4199 |

c) Hash table using quadratic probing

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 6 | 8 | 9 |
| 9679 | 4371 |  | 1323 | 6173 | 4344 |  |  | 1989 | 4199 |

2. Show the result of rehashing the hash tables in (1). Assume the size of each hash table is doubled.

a) Separate chaining hash table (doubled)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  | 1323 | 4344 |  |  |  |  | 1989 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|  | 4371 |  | 6173 |  |  |  |  |  | 4199 9679 |

b) Hash table using linear probing (doubled)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 9679 |  |  | 1323 | 4344 |  |  |  |  | 1989 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|  | 4371 |  | 6173 |  |  |  |  |  | 4199 |

c) Hash table using quadratic probing (doubled)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 9679 |  |  | 1323 | 4344 |  |  |  |  | 1989 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|  | 4371 |  | 6173 |  |  |  |  |  | 4199 |

3. For the following class, write a custom hashcode() and equals() method. Both should be based on the fields of the class.

public class Address  
{  
    private int number;  
    private String street;  
    private String city;  
    private String state;  
    private int zip;  
  
    //constructors, getters, setters, toString()...  
}

Note: Your equals method should address null parameters, the case of a.equals(a), and non-Address objects (see the article above).

Note: Your hash codes should be generated directly from the fields in the Address class. If the fields in two Address objects are the same, then their hash codes should be the same. Use your creativity to find a solution!

Note: You should not be using Eclipse to generate your hashcode() method. It should be based on your own design!

4. Make an argument for why you think your hashcode() function from (3) is a strong or weak candidate.

5a. Write a class called HashSet that stores an internal hash table of Objects and uses linear probing to address collisions. Assume your hash table is never resized. If a new element is added to a full table, then throw a new IllegalStateException. You class should have the following methods:

a)   add(x)  
b)   contains(x)  
c)   size()  
d)   isEmpty()

5b. Add the remove(x) method to your HashSet class above. This will require you to support a placeholder value in your hash table for removed elements.