CSE 250 Recitation

November 18~19: Hash Tables

Sets vs Maps

Remember: A hash table is a data structure...it can be used to implement multiple ADTs, like Sets and Maps

How would you implement Sets using a hash table? What about Maps?

- What are the differences?
- What are the runtimes of the main operations?

Come up with some examples of Sets vs Maps.

hashCode vs equals

Remember: Just because two objects map to the same hash code or same hash bucket, does not mean they are equal!

Consider **`BZPair`** in PA3 — we have overridden both the **`hashCode`** and **`equals`** functions so that BZPair can be used as a Key in our hash table

- **`hashCode`** returns an integer used to determine the bucket — two **`BZPairs`** with different birthday/zipcode COULD have the same hash code
- **`equals`** returns true **only if the birthday and zipcode are equal**

Hashing w/Chaining

hash(A) = 636

hash(B) = 712

hash(C) = 459

hash(D) = 12

hash(E) = 154

1. Start with a 5-bucket hash table (with chaining) and insert the above items
    a. What is the load factor?
2. Rehash the table, doubling its size to 10
    a. What is the load factor?
3. Think about how you would lookup something in the table? Something not in the table? Remove something?

Hashing w/Open Addressing

hash(A) = 636

hash(B) = 712

hash(C) = 459

hash(D) = 12

hash(E) = 154

1. Start with a 5-bucket hash table (with open addressing) and insert the above items
2. Ensure that lookup works for all 5 keys
    a. What if we try to lookup F which hashes to 72?
3. Remove B…ensure that lookup still works
4. Rehash the table, doubling its size to 10

Hashing w/Cuckoo Hashing

$h_1(A) = 636$ $h_2(A) = 312$

$h_1(B) = 712$ $h_2(B) = 242$

$h_1(C) = 459$ $h_2(C) = 684$

$h_1(D) = 12,$ $h_2(D) = 871$

$h_1(E) = 154$ $h_2(E) = 939$

1. Start with a 5-bucket hash table (using Cuckoo Hashing) and insert the above items
2. Rehash as needed…

Cuckoo Hashing Exercise

Imagine we are inserting **A**, **B**, and **C** into a hash table using Cuckoo Hashing…

1. Come up with unique hash values for **A**, **B**, and **C** that would require the hash table to rehash if there are 10 buckets
2. Do the same that would require the hash table to rehash for 20 buckets
3. Can you pick a set of unique hash values that would require the hash table to resize for both 10 **and** 20 buckets, but not 40?