

CSE 250 Recitation

4/22-4/23 : 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? 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

$$\text{hash}(A) = 636$$

$$\text{hash}(B) = 712$$

$$\text{hash}(C) = 459$$

$$\text{hash}(D) = 12$$

$$\text{hash}(E) = 154$$

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

$$\text{hash}(A) = 636$$

$$\text{hash}(B) = 712$$

$$\text{hash}(C) = 459$$

$$\text{hash}(D) = 12$$

$$\text{hash}(E) = 154$$

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

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

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

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

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

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

1. Start with a 5-bucket hash table (with open addressing) and insert the above items
2. Rehash as needed...