CS 310: Maps and Sets

Chris Kauffman

Week 9-1
Announcement

Mason Women in Computer Science: A Networking Event

- The event will take place in Sub 1 – 3B on Monday, November 14th 10am-12pm.
- We are organizing an event to encourage networking and community involvement among women undergraduate and graduate students and faculty.
- The event is indeed open to all students and faculty, though we do want to encourage underrepresented students in CS foremost.

– Amarda Shehu & Foteini Baldimtsi
Logistics

Goals Today
- Exams back
- Claire says "Work at Google!"
- Finish Hash Tables
- Maps and Sets

HW2 Questions?
- Due Tue 10/25
- Don’t forget ANALYSIS.txt

Reading from Weiss
- Today: Ch 6.7-9 Maps & Sets
- Upcoming: Trees
- Ch 18 Trees
- Ch 19 Binary Search Trees
- Weiss Ch. 7 Recursion
### Exams Back

<table>
<thead>
<tr>
<th>Stat</th>
<th>Val</th>
<th>Cut</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>110</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>81.71</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Median</td>
<td>85.00</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>StdDev</td>
<td>12.98</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>70</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>27</td>
</tr>
</tbody>
</table>
Operation Complexities (Speed)

- **add(x):** put x in the DS
- **removeLast():** get rid of "last" item
- **remove(x):** take x out of DS
- **contains(x):** is x in DS?

<table>
<thead>
<tr>
<th></th>
<th>add(x)</th>
<th>removeLast()</th>
<th>remove(x)</th>
<th>contains(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>0(1)</td>
<td>0(1)</td>
<td>0(n)</td>
<td>0(n)</td>
</tr>
<tr>
<td>LinkedList</td>
<td>0(1)</td>
<td>0(1)</td>
<td>0(n)</td>
<td>0(n)</td>
</tr>
<tr>
<td>Hash Table</td>
<td>0(1)</td>
<td>X</td>
<td>0(1)</td>
<td>0(1)</td>
</tr>
</tbody>
</table>

This table is slightly misleading
- Careful of semantics of each operation
- Presence/lack of sorting property
- Set/Map distinctions
- What about *space* complexity of each?
Idea and Implementation

List

- List Idea: Ordered collection, accessible by numeric index: \( l.get(i) \)
- List Idea is formalized in Java’s interface List
- ArrayLists and LinkedLists are both implementation of the List idea with different operational tradeoffs (describe)
- Could one implement a List with a hash table: HashList?

Set and Map

- Both useful ideas: set of unique items, mapping of keys to values
- Can implement set or map with a variety of data structures
- These two share structure so implementing a Set from a Map and vice versa is a good software engineering exercise
<table>
<thead>
<tr>
<th>Set: HashSet and TreeSet</th>
<th>Map: HashMap and TreeMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Collection of distinct objects</td>
<td>▶ (key,value) pairs</td>
</tr>
<tr>
<td>▶ Supports add(x), remove(x), contains(x), sometimes get(x)</td>
<td>▶ Each key has exactly one value</td>
</tr>
<tr>
<td>▶ x is either in the set or not in the set</td>
<td>▶ Insert value into a map according to its key</td>
</tr>
<tr>
<td></td>
<td>▶ Same key maps to same &quot;place&quot; in the data structure</td>
</tr>
<tr>
<td></td>
<td>▶ Supports put(k,v), get(k), remove(k), contains(k)</td>
</tr>
</tbody>
</table>
Examples of Sets and Maps

A data type

class Student{
    String name;
    int gNumber;
}

A set of students

Contents

{Kyle, 1234}
{Stan, 4321}
{Eric, 2486}
{Kenny, 1313}
{Stan, 1357}

A map of IDs to students

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>{Kyle, 1234}</td>
</tr>
<tr>
<td>4321</td>
<td>{Stan, 4321}</td>
</tr>
<tr>
<td>2486</td>
<td>{Eric, 2486}</td>
</tr>
<tr>
<td>1313</td>
<td>{Kenny, 1313}</td>
</tr>
<tr>
<td>1357</td>
<td>{Stan, 1357}</td>
</tr>
</tbody>
</table>

A map of Students to Majors

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Kyle, 1234}</td>
<td>-&gt; World Religions</td>
</tr>
<tr>
<td>{Stan, 4321}</td>
<td>-&gt; Geology</td>
</tr>
<tr>
<td>{Eric, 2486}</td>
<td>-&gt; Nutrition</td>
</tr>
<tr>
<td>{Kenny, 1313}</td>
<td>-&gt; Mortuary Sciences</td>
</tr>
<tr>
<td>{Stan, 1357}</td>
<td>-&gt; Genetics/Cloning</td>
</tr>
</tbody>
</table>
Questionable Sets and Maps

**A set of majors?**

Contents:
- World Religions
- Geology
- Nutrition
- Mortuary Sciences
- Genetics/Cloning

**A set of names?**

Contents:
- Kyle
- Stan
- Eric
- Kenny
- Stan

**A map of IDs to names?**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Kyle</td>
</tr>
<tr>
<td>4321</td>
<td>Stan</td>
</tr>
<tr>
<td>2486</td>
<td>Eric</td>
</tr>
<tr>
<td>1313</td>
<td>Kenny</td>
</tr>
<tr>
<td>1357</td>
<td>Stan</td>
</tr>
</tbody>
</table>

**A map of names to IDs?**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kyle</td>
<td>1234</td>
</tr>
<tr>
<td>Stan</td>
<td>4321</td>
</tr>
<tr>
<td>Eric</td>
<td>2486</td>
</tr>
<tr>
<td>Kenny</td>
<td>1313</td>
</tr>
<tr>
<td>Stan</td>
<td>1357</td>
</tr>
</tbody>
</table>
Array Analogy

Arrays and ArrayList are like a Map where

- Keys are integers: store at array index
- Values are the objects at those indices

A Set of Integers is naturally represented as an array of booleans

- Represent sets of Integers 0 to 10
- Use arrays of size 11
- The set \{1, 8, 9\} is the array

```java
boolean set1[] = new boolean[]{
    false, true, false, false, false, false,
    // 1
    false, false, true, true, false,
    // 8 9
};
```

More efficient with BitSet if you’re willing...
General Observations

Set

- A set must guarantee uniqueness of elements
- Typical approach is during `add(x)`, check `contains(x)` and don’t add duplicates but there are other approaches
- Efficient implementation of `contains(x)` and `get(x)` becomes important for sets

Map

- The collection so keys is a set - each key must be unique
- `contains(k)/get(k)` important - make them efficient
- Collection of values is not unique
- Usually not efficient to look up whether a given value is present
- Collection of (key,value) pairs is unique due to keys being unique
General Implementations

How would you implement a Set<T>?
  - Using an ArrayList?
  - Using an LinkedList?
  - Using a hash table?

How would you implement a Map<K,V>?
  - Using an ArrayList?
  - Using an LinkedList?
  - Using a hash table?
Have Set, Build Map

Q: If I have Set, how would I build Map?

Given: SimpleSet

- Collection of distinct objects
- Uniqueness determined by equals() method
- Operations add(x), get(x), remove(x), contains(x)
- SimpleSet implementation may be based on arrays, hash tables, trees, linked lists... you don’t know

Build: SimpleMap

- Set of (key,value) pairs
- Compare pairs only on whether their key is equal
  - (a,x) == (a,x)
  - (a,x) == (a,y)
  - (a,x) != (b,y)
  - (a,x) != (b,x)
- Use the set to ensure no redundant keys enter
- Implement put(k,v), get(k), remove(k), contains(k)
Trick 1: Use an internal class

```java
public class MapFromSet<K, V>{
    // Trick: Use a nested class
    // Class to carry around (key,val) pairs
    public static class KeyVal<K, V>{
        public K key; public V value;
        public KeyVal(K key, V value){
            this.key = key; this.value = value;
        }
        // Required for any set to work
        // Compare only based on key
        public boolean equals(Object o);
        // Required for HashSet to work right
        public int hashCode();
        // Required for TreeSet to work
        public int compareTo(KeyVal<K,V> kv);
    }
}
```
Trick 2: Use a set of the key/val pairs

Prototypes

```java
public interface SimpleSet<T> {
    boolean contains(T x);
    boolean add(T x);
    boolean remove(T x);
    T get(T x);
}
```

```java
public class MapFromSet<K, V> {
    // Trick 1: Use internal key/val class
    public static class KeyVal<K, V> {...}
    // Trick 2: Given a working Set class: use it!
    private SimpleSet<KeyVal<K,V>> theSet;

    // Implement these using theSet
    public MapFromSet();
    public void put(K key, V value);
    public void remove(K key);
    public boolean contains(K key);
    public V get(K key);
}
```

Exercise

Implement the `put()`, `remove()`, `contains()`, `get()` methods
The other Direction: Build a Set from a Map

Given a SimpleMap

- (key, value) pairs
- Each key is unique
- Insert value into a map according to its key
- Same key maps to same "place" in the data structure
- Supports put(k, v), remove(k), contains(k)

Build a SimpleSet

- Use an internal SimpleMap
- Implement SimpleSet methods
- void add(T x)
- void remove(T x)
- T get(T x)
- boolean contains(T x)

A Great Exam Question

- Tests your use of generics
- Illustrates abstraction skills
- Show you’re a proper software engineer
Java Does it

In Java: Map → Set

- `java.util.TreeMap` is a red-black tree
- `java.util.TreeSet` uses a `TreeMap`
- `java.util.HashMap` is a separate chained hash table
- `java.util.HashSet` uses a `HashMap`

In Weiss: Set → Map

- `weiss.util.TreeSet` is an AA-tree (another balanced tree)
- `weiss.util.TreeMap` uses `TreeSet`
- `weiss.util.HashMap` uses `HashSet`

Lesson

- Re-use when it makes sense
- Think hard about when it makes sense to re-use