CS 2230
CS II: Data structures

Meeting 28: SortedListMap
Brandon Myers
University of Iowa
Today’s big ideas

• SortedMap is an extension of Map that includes additional methods that depend on keys having an ordering

• Two approaches to implementing SortedMap are with sorted lists or search trees; we’ll look at SortedListMap

Helpful to have the code in front of you today: Go to 4/3 lecture materials on the website to access the Java code
DictionaryApp.java

Input: Text file with "WORD":"definition"
{"DIPLOBLASTIC":"Characterizing the ovum when it has two primary
erminallayers.",
"DEFIGURE":"To delineate. [Obs.]These two stones as they are here defigured.
Weever.",
"LOMBARD":"Of or pertaining to Lombardy, or the inhabitants of Lombardy.",
"BAHAISM":"The religious tenets or practices of the Bahais.",
"FUMERELL":"See Femerell.",
"ROYALET":"A petty or powerless king. [R.]there were at this time two other
royalets, as only kings by hisleave. Fuller.",
"TROPHIED":"Adorned with trophies.The trophied arches, storied halls, invade.
Pope.",...}
SortedMap.java

This interface extends Map, so a SortedMap can do everything a Map can do:

get, put, remove, ... 

in addition to methods that take advantage of some ordering on the Keys.
SortedMap class hierarchy

- SortedMap
- AbstractSortedMap
  - BSTMap
    - aka, AVLTreeMap
  - SortedListMap
AbstractSortedMap.java

Provides methods/constructors that are probably useful for many implementations of SortedMap

```java
protected AbstractSortedMap(Comparator<K> c) {
    comp = c;
}
```

The Comparator<K> interface provides the int compare(K o1, K o2) method

```java
/** Initializes the map with a default comparator. */
protected AbstractSortedMap() {
    // default comparator uses natural ordering
    this(new DefaultComparator<K>());
}
```

An ordering on the keys needs to be defined. AbstractSortedMap provides two constructors that allow you to define that ordering.
SortedMap class hierarchy

- SortedMap
- AbstractSortedMap
  - BSTMap
    - aka, AVLTreeMap
  - SortedListMap
    - aka, AVLTreeMap
SortedListMap<Integer, String>

entries

ArrayList<MapEntry<Integer, String>>

elements

count 2

MapEntry<Integer, String>

k "A"
v "The first letter ..."

MapEntry<Integer, String>

k "A CHEVAL"
v "Astride; with a..."
Peer instruction

What running time should we aim for for SortedListMap.get()?

a) O(1)
b) O(log n)
c) O(n)
d) O(nlogn)
e) O(n^2)

BONUS: running time for...
put
ceiling(key)
floor(key)
higherEntry(key)
lastEntry()
Iterating through the result of subMap
• put(k,v)
  • step 1: find the index, j, where k would go in sorted order
  • step 2:
    • if entries[j] already holds the key k, then overwrite the value of that entry with v
    • otherwise, add a new entry (k,v) at index j. The ArrayList.add method must copy everything after the insertion point!

• running time
  • step 1: O(logn) with binary search
  • step 2: O(1) in the first case, O(n) to do the copy in the second case; therefore worst case O(n)
  • therefore put is worst case O(n)
SortedListMap.java

• higherEntry(k)
  • step 1: find the index, j, where k would go in sorted order
  • step 2:
    • if entries[j] holds the key k, then return entries[j+1] if it exists
    • otherwise, return entries[j], which must have a larger key than k

• running time
  • step 1: $O(\log n)$ with binary search
  • step 2: $O(1)$ in both cases
  • therefore higherEntry is $O(\log n)$
subMap(“Do”, “Egg”) returns a snapshot iterable

Copy the portion of the array where the keys fall in range “Do” to “Egg” into a buffer (the “snapshot”)

Why? our copy will be safe. Removes removes on the map that occur before we return all elements from the iterator won’t affect the iterator (however, since we copied only references, puts that overwrite entries will be seen)
subMap runtime?

Let’s assume $n=$ number of entries and $s=$ number of entries returned in the subMap.

• $O(\log n)$ to find the start key using binary search
• $O(s)$ to iterate from the start key to the first key that is too big

therefore, subMap is $O(s + \log n)$
Administrivia

HW 7 out, due on 4/13

See announcement on ICON for signing up HW 8 partners (pair programming project)
Using version control in HW8

**Git**

You will get experience in managing versions of your code and sharing your code with a partner in HW8.

Next week’s discussion (4/10-4/12) is a crash course on git. Bring your laptop with git ready to go! (instructions forthcoming on ICON)
Today’s big ideas

- SortedMap is an extension of Map that includes additional methods that depend on keys having an ordering

- Two approaches to implementing SortedMap are with sorted lists or search trees; we’ll look at SortedListMap

Helpful to have the code in front of you today: Go to 4/3 lecture materials on the website to access the Java code
resources

• For an animation of binary search see http://www.cs.usfca.edu/~galles/visualization/Search.html