CSC207.01 2014S, Class 38: Implementing Dictionaries

Overview

- Preliminaries.
  - Upcoming work.
  - Admin.
  - Extra credit.
  - Questions.
- A dictionary ADT, continued.
- Review: Thinking about data structures.
- Association lists.
- Techniques for improving the implementation.
- Binary search trees.

Preliminaries

Upcoming Work

- No writeup today (particularly since there's no lab today)!
- HW 6 remains due Wednesday night.
- Exam 2 makeup due Thursday night (printed due in class Friday).

Admin

- Exams returned.
- I hope that you understand that yesterday's rant was intended as a positive action.

Extra Credit

- http://www.strikingly.com/pioneerweekend
- Review session tonight, 7 pm in 3821.
- Any one Spring into Humanities (or is it vice versa) talk.
- CS extra Thursday: Software to enhance wellness: The DavisJan team.
- CS table Friday: TBD.
- CS extra next Monday: Walker and Liberto on bluetooth.
- Get and wear one of the "1 in 4" shirts next week.
- UofI HackaThon this weekend.

Questions

Talk to us above model/view/controller, particularly for this.

See the whiteboard for details.

A dictionary ADT, continued

A slight change, to follow common terminology

- key, rather than "index"

Note: keys can be any type, rather than just string.

- Sam suggests that it might be useful to have dictionaries in which the keys are integers.
  - This seems odd, since arrays also map integers to values
  - We've made our dictionaries expand automatically, so that's nice.
    - But vectors expand automatically, too.
  - If our indices are not 0 ... n, we typically waste a lot of space if we use vectors or arrays.

The interface

```
/**
 * A dictionary of values of type V keyed by values of type K.
 */
public interface Dictionary<K,V>
{
  /**
   * Confirm that a key is valid.
   */
  public boolean containsKey(K key);

  /**
   * Change the entry for a particular key.
   *
   * @pre key is a valid key for the dictionary
   * @post get(key) == value
   */
  public void set(K key, V value)
    throws NoSuchKeyException;

  /**
   * Add the entry for a particular key.
   *
   * @pre key is not a valid key for the dictionary
   * @post get(key) == value
   */
  public void add(K key, V value)
    throws Exception;

  /**
   * Get the entry for a particular key.
```

```
 *
 * @throws NoSuchElementException
 *   If the key is not a valid key for the dictionary.
 */
public V get(K key)
  throws NoSuchElementException;

/**
 * Add a bunch of keys and values.
 *
 * @pre
 *   keys.length == values.length
 * @pre
 *   No element of keys is null.
 * @pre
 *   No two elements of keys are equal.
 * @pre
 *   No elements of keys are already valid.
 */
public void addAll(K[] keys, V[] values)
  throws SomeRandomException;

/**
 * Remove a value based on its key.
 */
public void remove(K key);

/**
 * Remove everything.
 */
public void clear();

/**
 * Remove all the entries with a given value.
 */
public void removeAll(V val);

/**
 * Get all of the keys in the dictionary.
 */
public Iterator<K> keys();

/**
 * Get all of the values in the dictionary.
 */
public Iterator<V> values();

/**
 * Get all of the key/value pairs in the dictionary.
 */
public Iterator<Pair<K,V>> pairs();

/**
 * Update all of the values.
```

```
   */
  public void map(BinaryFunction<K,V,V> update);

} // interface Dictionary<K,V>
```

Some design questions, major and minor

- What else are we missing?
  - Nothing, in the maximalist approach
- What is the relationship between `set` and `add`?
  - One expects the key already there, the other expects the key not to be there.
  - Is it worth having both?
    - Yes. Makes it clearer to someone who is using it that one simply changes the entry while another likely expands the dictionary.
    - Yes. Helps catch errors.
    - No. They are similar enough that it's pointless to have both.
    - No. Complicates the semantics of addAll.
- Think about arrays of keys
- What does `sort` do?
- Should the function that returns an iterator for values be called `values` or `iterator`?
- Should we have `remove` in the structure or in the iterator (or both)?

Deprecated

```
  /**
   * Create an array of keys.
   */
  public K[] keyArray();

  /**
   * Create an array of values.
   */
  public V[] valuesArray();
```

Review: Thinking about data structures

Association lists

Techniques for improving the implementation

Binary search trees