Lecture 9: Iterators

CS 62
Spring 2013
Kim Bruce & Kevin Coogan

Assignment 3

- On-disk sorting: What to do when more data than can fit in memory of computer?
- Read info on File I/O in Java and file systems in appendix to assignment. See on-line Streams cheat sheet!
- Lab 3: More complexity/timing

Sort Review

- Mergesort:
  - Algorithm: Divide in half, sort each half, then merge them in order
  - Complexity: O(n log n)
    - Intuition
    - Prove by induction
- Quicksort: New divide & conquer:
  - Algorithm: Move small elts to left, large to right
    Sort left elts, sort right elts, done!
  - Complexity: O(n log n) on average, $O(n^2)$ in worst case

Iterators

- Provide elements of data structure one at a time so can iterate through elts performing operations.
- Interface in standard Java

Iterator in Java

```
public interface Iterator<E> {
  // Returns true if the iteration has more elements.
  boolean hasNext()

  // Returns the next element in the iteration.
  E next()

  /**
   * Removes from the underlying collection the last element
   * returned by this iterator (optional operation).  */
  void remove()
}
```

Iterator Rules

- Remove is optional (we won't use it)
- Only allowed to call it once and then must terminate iteration.
- Never change a collection in middle of an iteration
  - Behavior is officially undefined if do!
  - Iterator often copies data structure before iterating, so changes may not appear to original!

Iterable

- Data structures with an iterator, satisfy interface Iterable:
  - Has method Iterator<T> iterator()
- Example: ArrayList<E> has method
  - Iterator<E> iterator()
- See definition and use of of iterator in ArrayIndexList<E>.

List Iterator

- Notice can have two iterators going through list independently!
- Never modify a data structure when iterating through elements as may get unpredictable results.
  - Most classes in Java collection classes have iterators which are designed to "fail fast". Throw an exception if continue with iterator (e.g., next()) after add or delete.

Linked Lists

- Alternate implementation of lists
- Trade-offs in complexity
  - With ArrayList expensive to add at beginning of list
  - Linked lists inexpensive to add early
  - However, slow to access ith element.

Linked List

- Composed of Nodes
  - Think of as pop-beads
  - See code in structure5 library
    - From documentation page!
- SinglyLinkedList (not std Java!)
  - keep track of head and size
  - Extends AbstractList -- look at on own!
    - Vector also extends AbstractList
  - Do algorithms on board!