CISC 3115 EWQ6
Sorting and Searching

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Module Outline

- Concept of data structure

- Use data structures

  - List

  - Sorting and searching in lists

  - Stack

  - Queue and priority queue

  - Set and map

Outline of This Lecture

- Sorting and searching with List, ArrayList, and LinkedList

  - Review of the Comparator and Comparable interfaces

  - Collections and Arrays classes

  - Sorting

  - Searching

Review: Comparable and Comparator

- Searching and sorting require comparison

- The Comparable interface

- The Comparator interface

Review: The Comparator Interface

- Defined as,

package java.util.Comparator;

public interface Comparator<T> {

      public int compare(T lhs, T rhs);

}

- The compare method: returns a negative value if lhs is less than rhs; a positive value if lhs is greater than rhs; and zero if they are equal.

Review: The Collections Class

- Recall previous lecture and examples

  - Comparable and Comparators

  - Example programs

- The Collections class contains various static methods

  - for operating on collections and maps,

  - for creating synchronized collection classes,

  - and for creating read-only collection classes

The Collections Class: Some Additional Methods

java.util.Collections	
List	
+sort(list: List): void	Sorts the specified list.
+sort(list: List, c: Comparator): void	Sorts the specified list with the comparator.
+binarySearch(list: List, key: Object): int	Searches the key in the sorted list using binary search.
+binarySearch(list: List, key: Object, c: Comparator): int	Searches the key in the sorted list using binary search with the comparator.
+reverse(list: List): void	Reverses the specified list.
+reverseOrder(): Comparator	Returns a comparator with the reverse ordering.
+shuffle(list: List): void	Shuffles the specified list randomly.
+shuffle(list: List, rmd: Random): void	Shuffles the specified list with a random object.
+copy(des: List, src: List): void	Copies from the source list to the destination list.
+nCopies(n: int, o: Object): List	Returns a list consisting of n copies of the object.
+fill(list: List, o: Object): void	Fills the list with the object.
Collection	
+max(c: Collection): Object	Returns the max object in the collection.
+max(c: Collection, c: Comparator): Object	Returns the max object using the comparator.
+min(c: Collection): Object	Returns the min object in the collection.
+min(c: Collection, c: Comparator): Object	Returns the min object using the comparator.
+disjoint(c1: Collection, c2: Collection): boolean	Returns true if c1 and c2 have no elements in common.
+frequency(c: Collection, o: Object): int	Returns the number of occurrences of the specified element in the collection.

Sorting

- static <T extends Comparable<? super T>> void sort(List<T> list)

  - Sorts the specified list into ascending order (i.e., standard order), according to the natural ordering of its elements.

- static <T> void   sort(List<T> list, Comparator<? super T> c)

  - Sorts the specified list according to the order induced by the specified comparator.

Sorting ArrayList, LinkedList, and arrays

- Examples
    - For lists, use Comparable, Comparator, and Collections
    - For arrays, use Comparable, Comparator, and Arrays
    - Question:
        - Which one is fast, sorting an ArrayList or sorting an LinkedList when the two lists are the same length, contain the same collections of values in the same order?

Questions?

- Sorting a list?

  - Key?

  - ArrayList vs LinkedList?

- "[what is the most efficient way to sort 1 million 32-bit integers?]"

Searching

- Sequential search

  - Iterate through the list, and find the key

  - On average, how many elements do we need to visit?

- Binary search

  - Java API provides binary search

    - Recall we implemented the binary search algorithm using recursion.

  - Requirement: a list must be sorted

  - On average, how many elements do we need to visit?

Searching: Examples

- Binary search on sorted lists

- Which one, ArrayList or LinkedList is more appropriate to do binary searches on lists?

  - Hint: which list is a "random access" list?

  - Compare computational time between binary search an ArrayList and a LinkedList

How about arrays?

- The Arrays class

- Sorting and binary search

Questions?

- Collections and Arrays

- Comparator and Comparable

- Sorting and searching lists and arrays

- Binary search on lists

  - What type of list should we select?