Overview

- Preliminaries.
  - Upcoming work.
  - Admin.
  - Questions on the homework.
- The problem of sorting.
- An object-oriented approach.
- Testing our sorting algorithm.

Preliminaries

Upcoming Work

- Homework 5 is due March 5.
- Reading for tomorrow: Sorting basics.

Admin

- Have fun with Earnest!
- Labs for the week are written. Readings are coming soon.
- I will take volunteers for note takers for Tuesday, Wednesday, and Friday. Extra credit for note taking.
  - MH Tuesday
  - KS Wednesday
  - EB Friday
- Extra credit:
  - Convocation, noon, Wednesday.
  - Presentations on Grinnell institutional image, noon on Thursday or Friday.
  - "We’re cool, we’re east campus, we just get B’s” hosts quizbowl at Lyle’s Tuesday night (we think)
- Things you should do
  - Balancing acts Friday, Saturday, Sunday
  - Neverland
- Don’t forget mentor session tomorrow night
Questions on HW5

The problem of sorting

- The goal of sorting: To put things in an order.
- How does that relate to the goal of lists, in which you also put things in order?
  - Lists the client controls the order
  - For sorting, there’s a specific order that you want to use
- For example you might sort an array
  - An array of strings in alphabetical order
  - An array of integers in ascending order
  - An array of integers in descending order
- We might also sort
  - Lists
  - Two-dimensional arrays, perhaps along two dimensions
- We probably wouldn’t want to sort
  - Stacks
- When sorting arrays, how does the sorting algorithm know what order to put them in?
  - We need a way to compare each element

Let’s write a signature

```java
public AN ARRAY sort(AN ARRAY, THE THING THAT DOES THE COMPARISONS)
```

Design issues:

- Is this a static method, or is it a method of the array class (a mutator)?

Let’s make that look more like Java

```java
public int[] sort(int[] vals, Comparator<Integer> order)
```

Using this model

```java
sort(grades, new Comparator<Integer>()
{
    public int compare(Integer x, Integer y)
    {
        return x-y; // Hack. Dangerous
    }
} // Comparator<Integer>
);
```

Whoops. Might overflow.

Let’s generalize
public static <T> T[] sort(T[] vals, Comparator<T> order)

An object-oriented approach

Perhaps sort should be a mutator of the array class.

public class JavaArray
{
    public void sort(Comparator<T> order)
}

● Whoops. We can’t (usually) extend the built-in classes

Object-oriented strategy: Make objects that know how to sort

public interface Sorter {
    * *Sort an array without mutation, return the sorted version. * * @pre * No elements may be null. */
    public T[] pureSort(T[] vals, Comparator order); /* * Sort an array in place. */
    public sortInPlace(T[] vals, Comparator order); } // class

Do we also have sorting routines for things with a natural order, such as BigIntegers or Strings.

public T[] pureSort(Comparable<T>[] vals) // Approximate syntax

Another design decision: Is the sorting algorithm "stable"?

● Stable sorting algorithms guarantee that if A precedes B before sorting, and A is equal to B
(according to the comparator), then A is still before B in the sorted array.

Four algorithms to study

● Insertion sort
● Selection sort
● Merge sort
● Quicksort

Testing our sorting algorithm

Unit tests for sorting algorithms

● Edge cases:
  ○ Something already in order
  ○ Something in backwards order
  ○ Something with extreme values (e.g., an array of integers using Integer.MAXVALUE and Integer.MINVALUE)
  ○ Empty array
  ○ All the same.
  ○ Containing some strange values, like null
● Normal cases
• Making lots of cases
  • Random test
    • Generate a random array
    • Sort it
    • Check if it’s sorted
    • Check if the result is a permutation of the original
  • Generate a sorted array
    • Randomize it
    • Sort it
    • Compare to the original
  • Excessive
    • Try the previous with *every* permutation of the array

Copyright (c) 2013-14 Samuel A. Rebelsky.

[Creative Commons Attribution 3.0 Unported License] To view a copy of this license, visit [http://creativecommons.org/licenses/by/3.0/] or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.