Sorting, Searching, and Manipulating Arrays

2-8-2005

Opening Discussion

- What did we talk about last class? Do you have code to show?
- Do you have any questions about assignment #2?
- For the minute essay last time I asked you to allocate two different 2D arrays of shape. Let's take a second to look at how that is done.
- What do you remember about sorts from PAD1?

Sorting

- One of the things that we want to be able to do on computers with any sequential collection is to sort the elements of that collection. Here are several "slow" sorts that you should know of.
- Bubble Sort: Repeated passes swapping out of order adjacent elements. Elements "bubble" to far end. $O(n^2)$

Meaning of O

- You probably saw this O-notation in PAD1, but didn't have it formally introduced.
- Formally an algorithm is O(f(n)) for a given operation if for input size n, the number of operations, g(n), obeys the following rule.

$$\exists n, c : \forall m > n, c * f(m) > g(m)$$

- Notice this only matters for large input sizes.

More Sorting

- Selection Sort: Each pass you find the min/max of what is left unsorted and swap it to the end. $O(n^2)$
- Insertion Sort: Walk through each elements, inserting it into the earlier elements. $O(n^2)$
- Optional: Shell's Sort: Repeatedly does insertion sort on subsets made with "diminishing gaps". $O(n^{3/2})$ or so.

Searching

- The other activity we are typically interested in doing with collections is searching for items in them.
- With a general collection all we can do is a sequential search where we walk through all the elements until we find what we want. Obviously, if you do this frequently, it can be a bit slow. O(n)

Fast Searching

- If you have a sorted array you have other options. The most common of which is a binary search.
- Start by looking in the middle of the array. If it is what you are looking for then you are done. If it is bigger than what you want you only have to look on the bigger side and the opposite is true for smaller values. Repeat this always cutting the size in half. O(log(n))

java.util.Arrays

- This is a “Utility” class that has only static methods in it. All the methods in it work with arrays.
- It includes methods for sorting and searching. While you probably won’t be using this directly in this class, it is a very good thing to know about because it can make your work faster in the future.

Polymorphism and Code

- One of the big things we'd like to be able to get from Java that was hard for us to get in C is polymorphic sorting. It would be nice to write one sort and have it work with many types. There are two ways to do this.
  - Work with Comparable interface.
  - Pass in a comparator.
- Now let's write up two sorts and a binary search that will work polymorphically.

Minute Essay

- Do you have any questions about the sorting or searching algorithms? Is it clear how polymorphism lets us write one algorithm for any object type in Java?
- Remember that design #2 is due today and the working code is due on Thursday. Your design needs to have all the methods you expect to write with comments.