Sorting and Searching

4-2-2012

Opening Discussion

- Minute essay comments:
  - Alphabetical sorting?
  - Writing collision detection.
  - Anonymity in electronic evals.
  - What drugs do you sell to an astrophysicist?
  - When will we use an IDE? Which one?
  - Relative speed of comparison and swaps.

Selection Sort

- This is often called a min-sort or a max-sort depending on how you write it. I'll describe a min-sort here.
- Inner loop:
  - Find the smallest element and SWAP it into position if not already there.
- Outer loop:
  - Repeat n-1 times so all elements are in the right place.
- Does only O(n) swaps, but still $O(n^2)$ comparisons.

Insertion Sort

- Inner loop:
  - Take the next element and shift it down to the right spot.
- Outer loop:
  - Run through all the elements starting with the second.
- This sort is actually a bit faster (factor of 2) on random data. It is really efficient on nearly sorted data.

Watching Them Work

- One advantage of doing graphics before sorting is that we can write code to visualize what is happening when we sort numbers with these sorts.
- Let's write this code and watch our sorts work.

Searching

- Many times we have to search in our data for where something is.
- If the data is not sorted, we have to use a linear search which will look at every element, one after another, to see if any matches what we want.
- This is O(n).
- The methods on collections in Scala use this approach.

Binary Search

- If the data is sorted, we can do something much better.
- We check the middle to see if it matches. If it does, return it. Otherwise, see if what we want is above or below the middle and repeat the process on only that half.
- This continually divides the things we are searching in half.
- Order?

Performance of Binary Search

- Dividing something by a fixed fraction repeatedly leads to O(log n) speed.
- O(log n) is much better than O(n) when n is large. To see this, consider a base 2 log for 1000, 1000000, or 1000000000.

Computer Memory

- The memory that a program uses is broken into two different parts.
  - Stack – This holds local variables. Every function/method call gets a new “frame” on the stack. Efficient, but limited.
  - Heap – All objects in Scala are allocated on the heap. It is big and flexible, but disorganized.
- Other languages allow you more direct control over memory. This has the potential to lead to errors.

Classification of Bugs

- We classify the errors that occur in programs in three broad groups.
  - Compile Errors – Found by the compiler. Gets a reasonable error message and line number.
  - Runtime Errors – Program crashes while running for a particular input. Gives type of error and line.
  - Logic Errors – Code runs fine, but does wrong thing. No information given to help you.
- You want to have your errors be higher up on this list because it gives you more information and makes it easier to fix.

Minute Essay

- What are examples when you use a computer to “search” in your life.