CSSE 220 Day 26

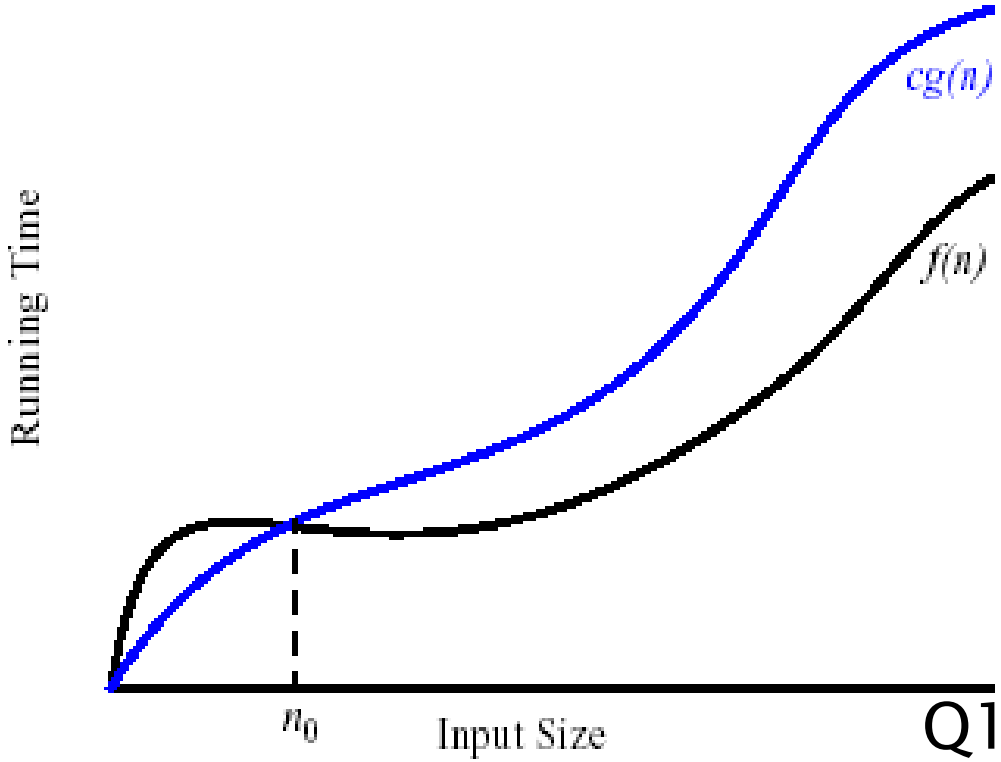
Sorting Wrap-up
Function Objects and the Comparator Interface
Linked Lists

Checkout *LinkedList* project from SVN

Questions

Big-Oh Notation

- We write f(n) = O(g(n)), and say "f is big-Oh of g"
- if there exists positive constants c and n_0 such that
- $0 \leq f(n) \leq c\ g(n)$ for all $n > n_0$
- g is a ceiling on f



Shortcut: Take highest order term in f and drop the coefficient.

Course Goals for Sorting: You should…

- Be able to describe basic sorting algorithms:
  - Selection sort
  - Insertion sort
  - Merge sort
  - Quicksort
- Know the run-time efficiency of each
- Know the best and worst case inputs for each

Selection Sort

- Basic idea:
  - Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)

  - Find the smallest number in the unsorted part
  - Move it to the end of the sorted part (making the sorted part bigger and the unsorted part smaller)

Repeat until unsorted part is empty

Insertion Sort

- Basic idea:
  - Think of the list as having a sorted part (at the beginning) and an unsorted part (the rest)

  - Get the first number in the unsorted part
  - Insert it into the correct location in the sorted part, moving larger values up to make room

Repeat until unsorted part is empty

Q2b

Merge Sort

- Basic recursive idea:
  - If list is length 0 or 1, then it's already sorted
  - Otherwise:
    - Divide list into two halves
    - Recursively sort the two halves
    - **Merge** the sorted halves back together
- Let's profile it…

Analyzing Merge Sort

- Use a recurrence relation again:
  - Let T(n) denote the worst-case number of **array access** to sort an array of length n
  - Assume n is a power of 2 again, $n = 2^m$, for some m

- Or use tree-based sketch…

Q3,2c

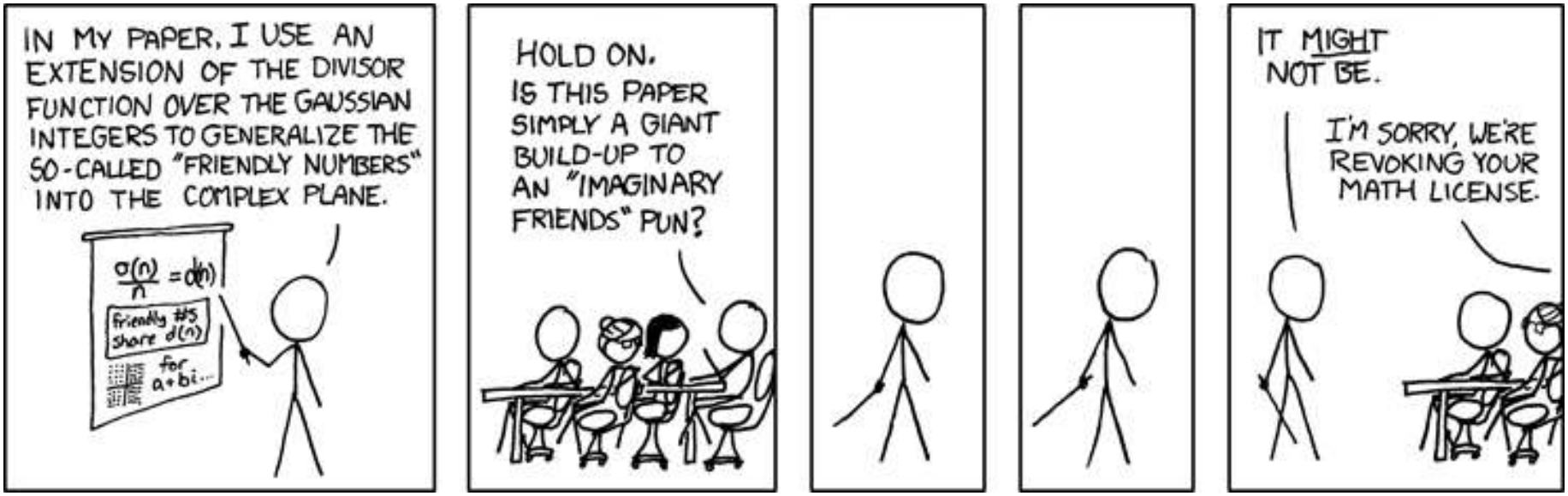
Quicksort

- Basic recursive idea:
  - If length is 0 or 1, then it's already sorted
  - Otherwise:
    - Pick a "pivot"
    - Shuffle the items around so all those less than the pivot are to its left and greater are to its right
    - Recursively sort the two "partitions"
- Let's profile it…

Analyzing Quicksort

- Using recurrence relation involves some seriously heavy lifting
  - See CSSE/MA 473
- But we can sketch the idea using trees…

Math Paper



That's nothing. I once lost my genetics, rocketry, and stripping licenses in a single incident.

Function Objects

- Another way of creating reusable code

A Sort of a Different Order

- Java libraries provide efficient sorting algorithms
  - `Arrays.sort(…)` and `Collections.sort(…)`
- But suppose we want to sort by something other than the "natural order" given by `compareTo()`
- Function Objects to the rescue!

Function Objects

- Objects defined to just “wrap up” functions so we can pass them to other (library) code
- We’ve been using these for awhile now
  - Can you think where?
- For sorting we can create a function object that implements `Comparator`

Data Structures

- Understanding the engineering trade-offs when storing data

Data Structures

- Efficient ways to store data based on how we'll use it
- The main theme for the last 1/6 of the course
- So far we've seen `ArrayLists`
  - Fast addition **to end of list**
  - Fast access to any existing position
  - Slow inserts to and deletes from middle of list

Another List Data Structure

- What if we have to add/remove data from a list frequently?

- `LinkedLists` support this:
  - Fast insertion and removal of elements
    - Once we know where they go
  - Slow access to arbitrary elements

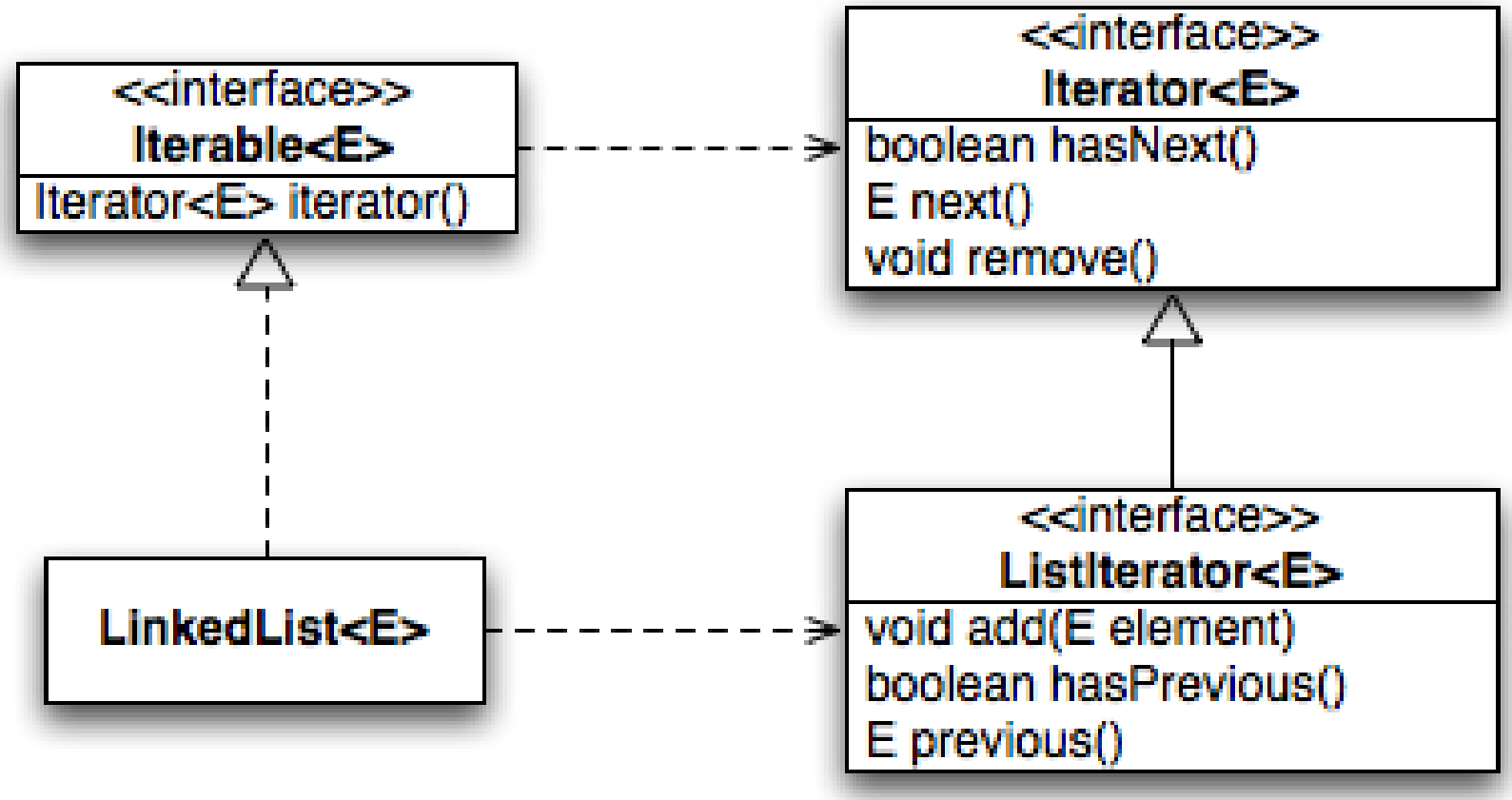
Q5,6

LinkedList<E> Methods

- `void addFirst(E element)`
- `void addLast(E element)`
- `E getFirst()`
- `E getLast()`
- `E removeFirst()`
- `E removeLast()`

- What about the middle of the list?
  - `LinkedList<E> implements Iterable<E>`

Accessing the Middle of a LinkedList



An Insider's View

```
for (String s : list) {
  // do something
}
```

```
Iterator<String> iter =
    list.iterator();

while (iter.hasNext()) {
  String s = iter.next();
  // do something
}
```

Enhanced For Loop

What Compiler Generates

Next Time

- Implementing `ArrayList` and `LinkedList`
- A tour of some data structures
  - Including one that will come in handy for storing a dictionary!

Vector Graphics Demos