Big Oh and Unit Testing

CSSE 221

Fundamentals of Software Development Honors

Rose-Hulman Institute of Technology

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Announcements

- Please commit your BigRational code as you go
- Roll call again
- Lab hours Sunday – Thursday, 7:00 – 9:00 pm
- Don’t need to bring book to class if you are familiar with the reading for that day and don’t need to reference it in class.
- Any questions?
  - Course mechanics? Syllabus? Angel?
  - BigRational?
  - Interfaces?

Capsule Teams: Section 1

- Research & Summary
  - Inheritance: Bismayer, Savkovich, Venezia
  - Polymorphism: Guilford, Memering, Spry
  - 1D and 2DArrays and ArrayLists: Goldthorpe, McCormack, Piergivianni
  - GUI using Swing: Juneau, Spry, Taylor
  - EventListeners: D'attilio, Katz, Roetker
  - Shape classes: Bismayer, Goldthorpe, Venezia

Capsule Teams: Section 2

- Research & Summary
  - Inheritance: Cox, Nuanes, Uphus
  - Polymorphism: Kulczak, Rudich, Stewart
  - 1D and 2DArrays and ArrayLists: Andrews, Burns, Carter
  - GUI using Swing: Harbison, Morrison, Nuanes
  - EventListeners: Kulczak, Shah, Singer
  - Shape classes: Alves, Jones, Morrison

This week: BigRational assignment

- Yesterday:
  - API (Application Programming Interface)
  - Interfaces: writing to a contract
- Today:
  - **Unit Testing: searching for logic errors**
  - **Introduction to efficiency analysis: "big-Oh"**
- Thursday:
  - Exceptions: throwing and catching

Unit Testing

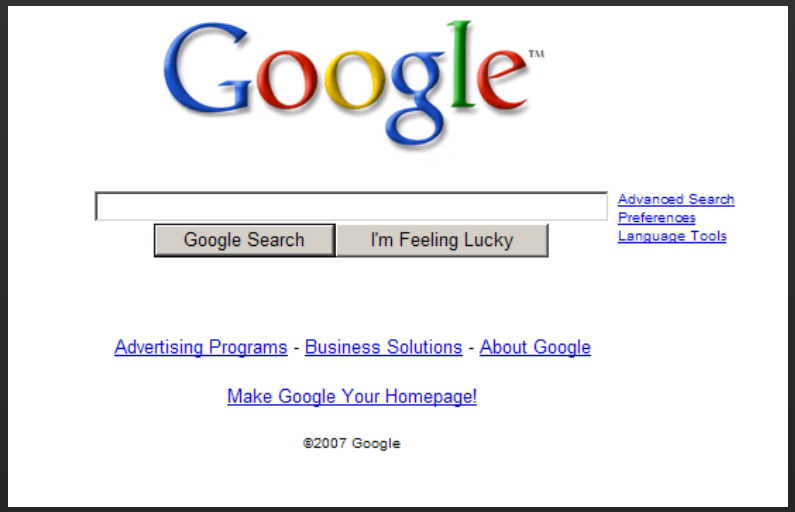
- What do you think it is?
  - Testing parts of your code in isolation before putting them all together
- Why is it a good thing?
- How do I write good test cases?
- How easy is it to do it in Eclipse?
  - Fairly so, with JUnit
- Let's see. Open Projects/UnitTesting and do it together now.

Break

- http://xkcd.com/489/

Efficiency is important

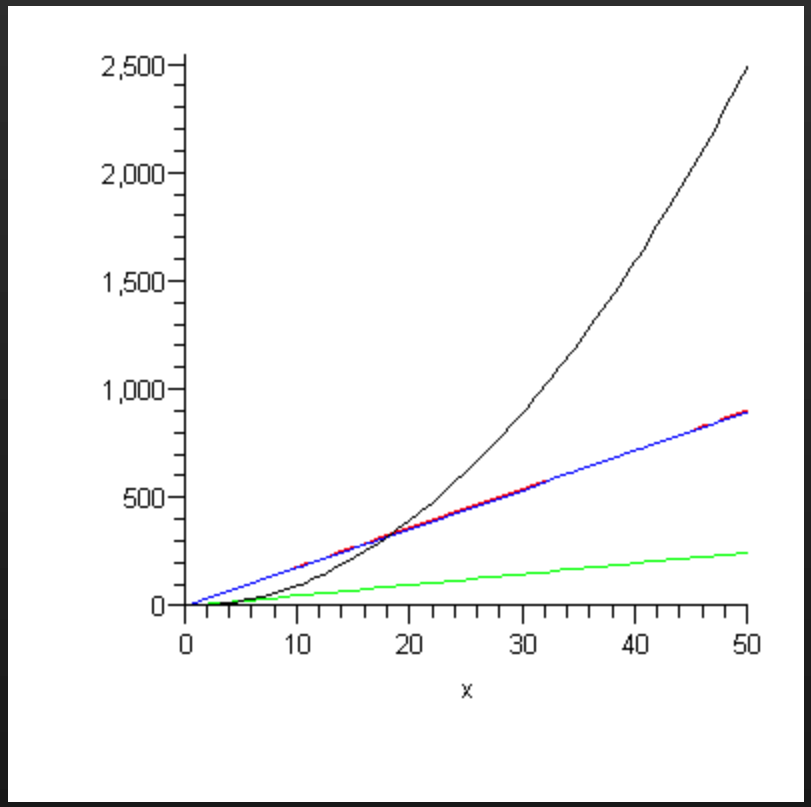
- Example?
- Not all a software problem
- Algorithms
  - Inherent complexity
  - Assume time spent is a function of the size of the input
  - Big-Oh focuses on the **most important** part of the function!



Now: plot y=18x + 5, y=18x, y = 5x, $y=x^2$ Which grows most quickly?

Efficiency is important

```
plot([18 x + 5, 18 x, 5 x, x^2 ]
    , x = 0..50, color = [red, blue, green, black] )
```



$y=x^2$

$y=18x + 5$
$y=18x$

$y = 5x$

- Simple Rule: Drop lower order terms and constant factors.
  - $7n - 3$ is $\boldsymbol{O}(n)$
  - $8n^2 \log n + 5n^2 + n$ is $\boldsymbol{O}(n^2 \log n)$

- Special classes of algorithms:
  - logarithmic: $\boldsymbol{O}(\log n)$
  - linear $\boldsymbol{O}(n)$
  - quadratic $\boldsymbol{O}(n^2)$
  - polynomial $\boldsymbol{O}(n^k)$, k 1
  - exponential $\boldsymbol{O}(a^n)$, $n>$ 1

Also: **constant: O(1)**

Figure 5.1
Running times for small inputs

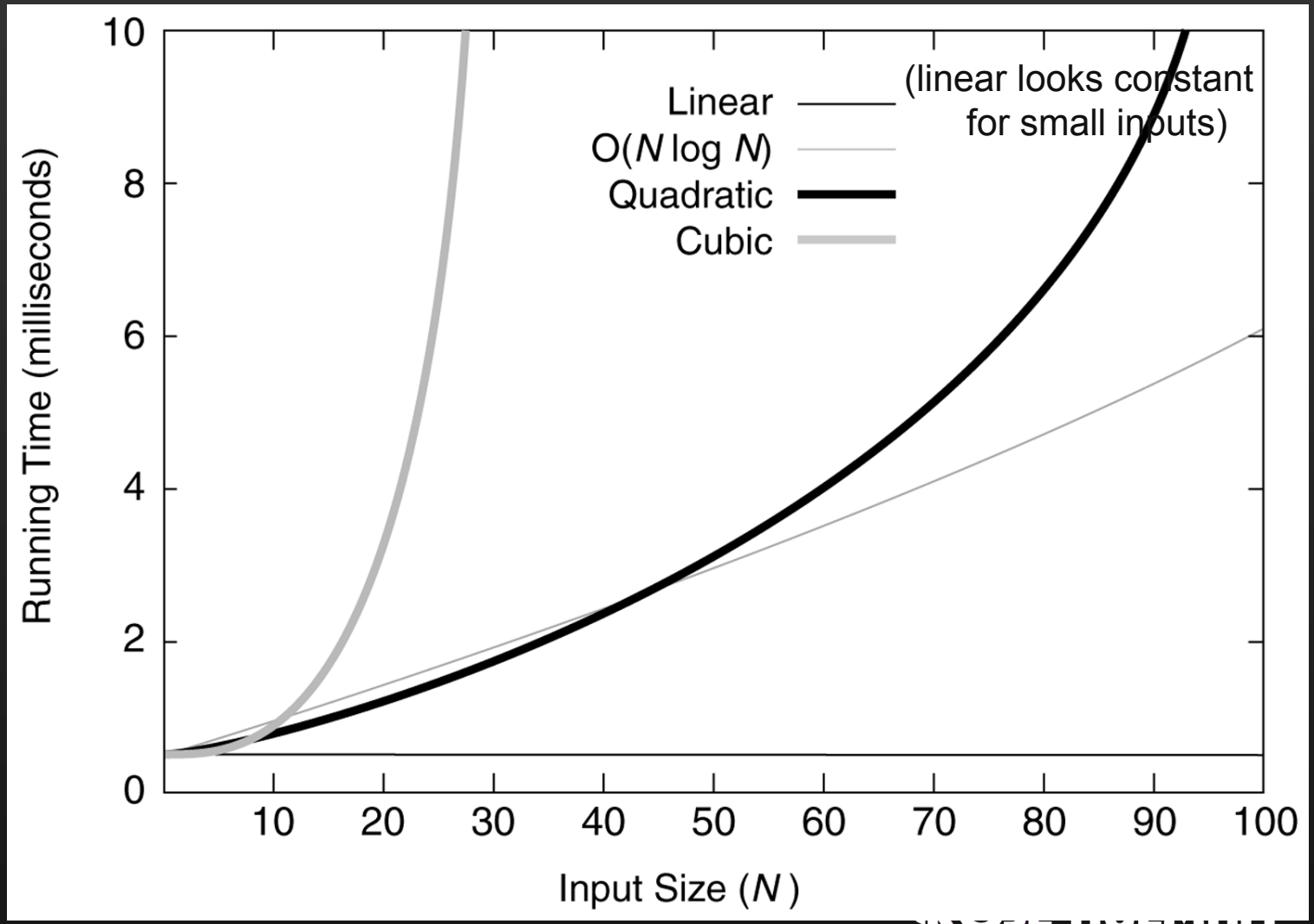
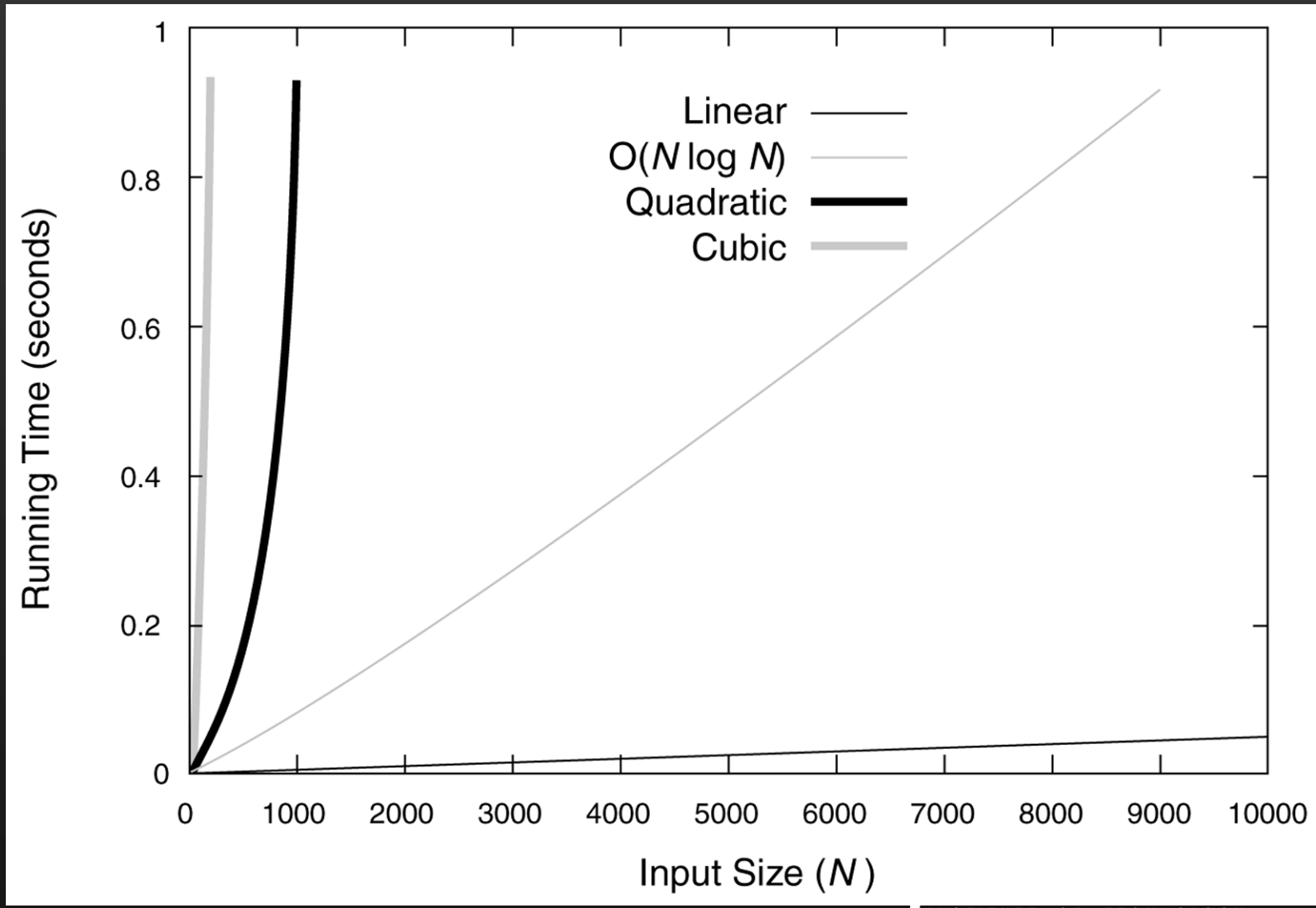


Figure 5.2
Running times for moderate inputs



Exercise

- Generate graphs like this from **simple code that you write and time**. Our goal is to be able to complete these statements, given the patterns you see:
  - a single loop is O(???)
  - a nested loop is O(???)
  - …

- Go to Projects/Runtime Exploration.
- This mini-homework will be handed in next class.