CSCI 136
Data Structures & Advanced Programming

Fall 2020

Instructors

Bill Jannen & Bill Lenhart

Outline

- Course Materials and Tools
- Course Preview

Course Materials and Tools

Where is everything!?

- GLOW
  - CSCI 136 R1 (the 'lecture' section)
    - Link to course website
    - Announcements, surveys, …
  - CSCI 136 R2/03/R4 (the conference sections) and R5/R6/R7/R8/R9 (the lab sections)
    - Zoom meeting information
- Course website:
  - http://cs.williams.edu/~cs136/index.html
  - Syllabus, schedules, links to (virtually) all content

Tools

- GLOW/Course Website
  - Course content and announcements
- Zoom
  - Remote conferences, labs, faculty/TA office hours
- Email
  - Announcements, formal communication with instructors/staff
- Slack
  - Quick messaging, conversation threads
- Atom/Git/Java
  - Your programming environment
- Williams VPN
  - Only needed to interact with GitLab server

Zoom Etiquette

Course Preview

Why Take CS136?

- To learn about:
  - Data Structures
    - Effective ways to store and manipulate data
  - Advanced Programming
    - Combine data structures, programming techniques, and algorithmic design to write programs that solve interesting and important problems
  - Basics of Algorithm Analysis
    - Measuring algorithm complexity
    - Establishing algorithm correctness

This is So Goals*

- Identify basic data structures
  - list, stack, array, tree, graph, hash table, and more
- Implement these structures in Java
- Learn how to evaluate and visualize data structures
  - Linked lists and arrays both represent lists of items
  - Different representations of data
  - Different algorithms for manipulating/accessing/storing data
- Learn how to design larger programs that are easier to modify, extend, and debug
- **Have fun!**

*Bill L has a college-aged daughter....

Course Outline

- Java review
- Basic structures
  - Lists, vectors, queues, stacks
- Advanced structures
  - Graphs, heaps, trees, dictionaries
- Foundations (throughout semester)
  - Vocabulary
  - Analysis tools
  - Recursion & Induction
  - Methodology

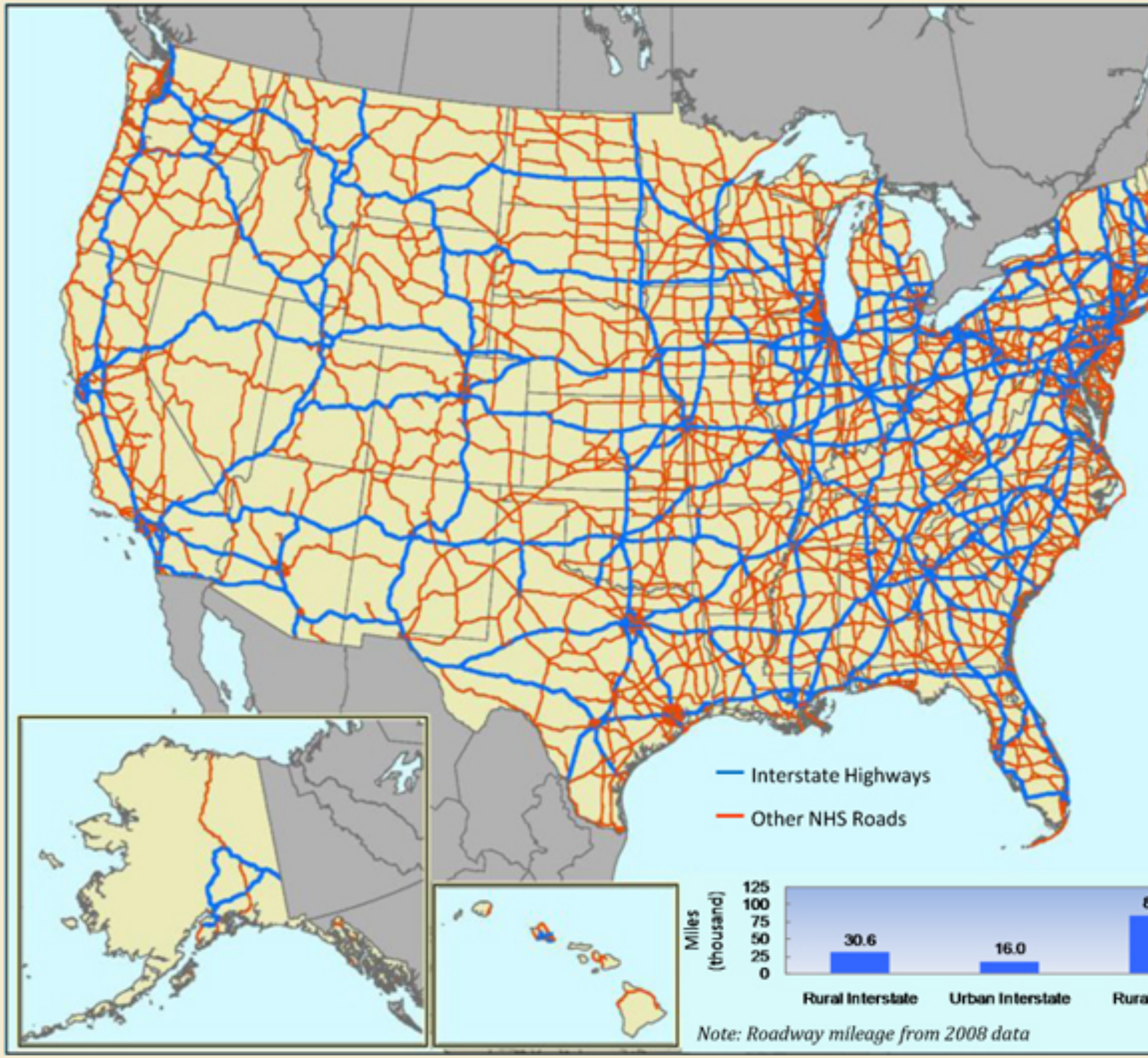
Why Java?

- There are lots of programming languages…
  - C, Pascal, C++, Java, C#, Python
- Java was designed in 1990s to support Internet programming
- Why Java?
  - It's easier (than predecessors like C++) to write correct programs
  - Object-oriented – good for large systems
  - Good support for abstraction, extension, modularization
  - Automatically handles low-level memory management
  - Very portable

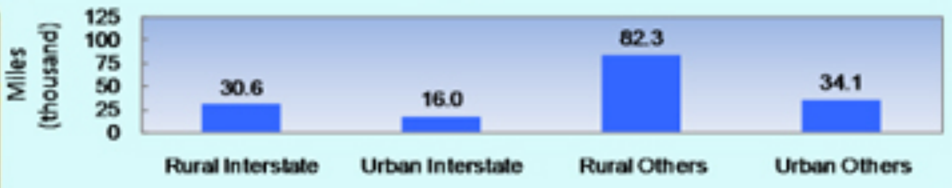
Common Themes

1. Identify data for problem
2. Identify questions to answer about data
3. Design data structures and algorithms to answer questions *correctly* and *efficiently*
   - Note: not all correct solutions are efficient
   - And vice versa!
4. Implement solutions that are robust, adaptable, and reusable

Example: Shortest Paths in Networks



National Highway System (NHS) roadways are important to the economy, defense, and mobility. The NHS includes all Interstate highways (arterials), the Strategic Highway Network (defense purpose), intermodal connectors (roads connecting to major intermodal facilities), and other principal arterials. The NHS includes over 163,000 miles of highways.



Note: Roadway mileage from 2008 data

Finding Shortest Paths

- The data: road segments
  - Road segment: Source, destination, length (weight)
- The question
  - Given source and destination, compute the shortest path from source
- The algorithm: Dijkstra's Algorithm
- The data structures (spoiler alert!)
  - Graph: holds the road network in some useful form
  - Priority Queue: holds not-yet-inspected edges
  - Also uses: Lists, arrays, stacks, ...
- A demo….