CS2: A Twofold Approach

Benjamin Kuperman
Oberlin College

About myself

- No prior background in CS2 education
- Taught CS2 at Swarthmore College 4 times
- Taught CS2 at Oberlin College 1.5 times

Course goals

- Language independent understanding of data structures and their algorithms
- Implementation of data structures and an application using the structure
- Refinement of skills and techniques from CS1

Idea behind 2-fold approach

- **Lectures** cover concepts and algorithms
  - Try to keep it language neutral
  - But use terminology consistent with lab
- **Labs** cover implementation via hands-on learning
  - Tie in language specifics

Lecture style

- Almost no computer or code
  - ~6 times in the semester
- Learn Big-O early, use it often
  - Everything discussed in terms of Big-O
- Make them doubt themselves to become confident in themselves

Exams

- **Short answer questions**
  - Definitions, concepts, trade-offs, solution design, spot-check labs
- **Data structure questions**
  - Ask for outlines of algorithms
  - Demonstrate actions on data structures

Laboratory assignments

- 2 primary components to each
  - Data structure
  - Application (problem?)
- Most allow pair programming
  - Usage alternates by semester
- Use large, real-world data set to make it worthwhile

Data structures

- Implementations of Java collections classes
  - MyArrayList, MyHashMap, etc.
    - Identical method signatures and behavior
- Trying out partial implementations via abstract
  - Unclear which is better

Other topics

- Eclipse IDE
- Pair programming
- Documentation
- Code reuse/maintainability
- Version control

Group assignments

- Tradeoff: larger, 2-stage assignments
- Many worked on their own
  - Difficulty scheduling
  - Wanted to learn
- Setup SVN repository for pairs to share code
  - more useful than expected

Assignments Spring 2008

- MyArrayList/Testing
- Algorithm Timing
- Maze Solver (stacks, queues, recursion)
- Email directory (linked lists)
- Binary Tree methods
- Word frequency tree
- Processing search queries (Binary heaps)
- Caching results, GUI (Hashtables)
- Boggle solver (Tries)
- Kevin Bacon Game (Graphs)

Student Feedback

- Favorite: email database, Kevin Bacon game
- Common regrets:
  - “Wish I’d started earlier”
  - “Wish I had worked with a partner”
- Eclipse: “angry red line”, debugger, quick fix

Hints on making this work

- Use consistent Java collections syntax
- Use a book with complete implementations and Java syntax
- Be prepared to spend time in the lab
  - Consider hiring weekend helpers

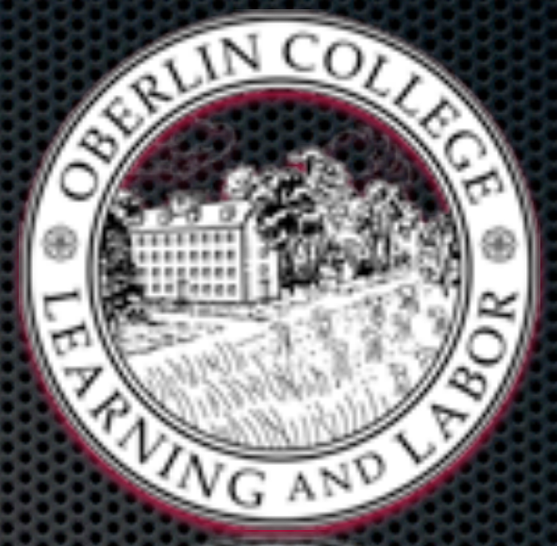
Large, real-world datasets

- Motivates the use of efficient algorithms
  - Must be larger than they could do by hand
- Real-world data is more interesting
  - Database search for their friends instead of "A. Random Student" or "student 12"

Data sources

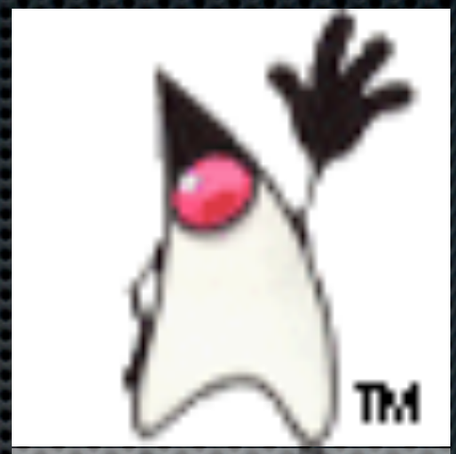
- Student directory information
- Library/card catalog
- Project Gutenberg
- Wikipedia
- IMDB
- CIA World Fact Book
- US Census data
- FreeDB
- Celestia/Astro DB
- JDK
- Thinking in Java
- WWW/Web crawling
- Swivel.com

Student directory information



- Ask for it from registrar(?)
- Often online, searchable database
  - Usually can generate URLs & download
- BlackBoard has passwd like users file
- Library often has searchable catalog

Sun's JavaDoc

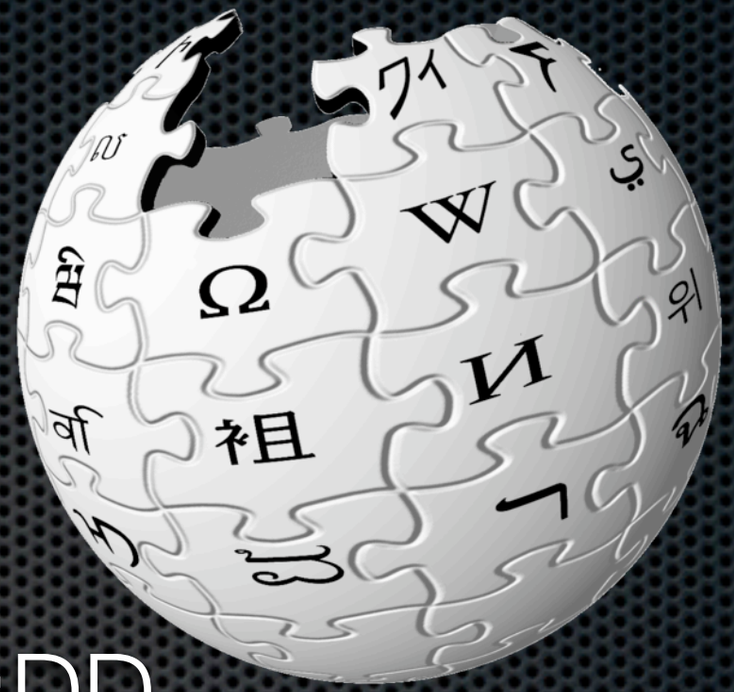


- Full copy of Java documentation in HTML
  - http://java.sun.com/javase/downloads/index.jsp
- Local mirror copy (284MB)
  - Do "file://" URL processing instead of http

IMDB.com

- Full database available for download
  - http://www.imdb.com/interfaces#plain
- You want “actors.list.gz” & “actresses.list.gz”
- 193MB processed, 4,799,462 entries

Wikipedia



- http://en.wikipedia.org/wiki/WP:DD
- ~8GB of raw data
  - HTML, XML, SQL
- perl parser for XML
- Static dump, meta info dump