

## The compressed-core program

### Why compressed?

Knox College, like many liberal arts colleges, imposes a limit as to how many courses can be required in order to complete a major. Though we want them to have to take advanced electives, we don't want to graduate any CS majors without at least some exposure to all the different subfields in CS. As a result, the compressed-core approach outlined in CC2001<sup>1</sup> has proven to be an excellent fit.

### The intermediate core

**CS262: Information and Knowledge Management**

**CS201: Computer Organization and Assembly Language**

**CS226: Operating Systems and Networking**

**CS205: Algorithm Design and Analysis**

**CS292: Software Development and Professional Practice**

### Requirements for the CS Major

- CS1, CS2
- Discrete math
- All 5 intermediate core courses —including CS 262
- Any 3 advanced CS electives
- 2 additional math electives

### Requirements for the CS Minor

- CS1, CS2
- Discrete math
- 4 intermediate or advanced courses —at least one advanced

# A Compressed, Breadth-second Approach

## Implementing CS262c “Information and Knowledge Management”

Don Blaheta  
Knox College

### Strengths of CS262c

**As prerequisite.** In addition to being a mandatory core course, CS262 is a prerequisite for several upper-level classes, including Artificial Intelligence and Database Systems. Since students already have the basic concepts and skills in these areas, we are able to hit the ground running in these electives and go further; in the advanced-level courses, students are now implementing more sophisticated projects and reading and presenting recently-published research papers on the cutting edge.

**Project synergy.** In some cases, projects will naturally have components that correspond with multiple different units of material. For instance, any realistic database project will need a user interface of some sort—and thus the project can explicitly include both a DB design portion as well as an HCI portion.

**Language exposure.** Though many acknowledge the value of exposing students to multiple programming languages, departments whose CS1/CS2 sequence is entirely in one language often don't find time to introduce students to other languages until much later. Our CS262 course spends time in both Prolog (in the Logic and Reasoning unit) and SQL (in the Databases unit).

**Ethics and social issues discussion.** The concept of information as value is a natural fit for the theme of the course, and from there we find a natural progression to copyright and intellectual property, open source software, and privacy issues. In the most recent iteration of the course, we had the students themselves research and present these topics, and lead a discussion afterwards; this technique elicited rather more engagement and discussion, and we plan to continue it.

### Caveat Professor

**Assignments are tricky.** The first homework in nearly any course is hard to write, because the students don't know very much about the topic yet; here, nearly every assignment is one of these “first assignments”.

**Don't try to segue everything.** The course is thematically driven by issues of representing and processing information, but at its core a ‘breadth-second’<sup>2</sup> course like this is still a sort of survey course—not quite like the ‘breadth-first’ CS1 courses, but in a similar spirit—and sometimes it's appropriate to just make a clean break and move on to the next topic.

**Student experience is minimal.** This depends to some extent on where in your sequence CS262 falls, but if CS2 is the only prerequisite, students may not have had important background in graph theory, algorithms, and other concepts. Particularly if one is accustomed to teaching upper-level courses in AI, NLP, and the like, it is important to note that a unit of CS262 is not quite the same as the first two to three weeks of an advanced course: the students don't have as much background.

**Lack of textbooks.** There are several textbooks whose first three to six chapters would make an appropriate text for one unit of the course. At \$100 a book, it is obviously infeasible to require three or four books for the course just for their introductory chapters. Knox's CS262 course has instead gone without a required textbook, putting several textbooks on reserve at the library and relying on a variety of web resources including Wikipedia—a decent stopgap, but not really satisfactory in the long term. This is probably the #1 stopper on a class like this, and mindful of that fact, I'm thinking of assembling my notes into a textbook; please let me know if you would be interested in hearing more about that!

## CS262 Topics

### Logic & Reasoning

- Propositional logic
- Predicate logic
- Prolog

### Discrete probability

- Joint and conditional probability
- Bayes' Law
- Inducing probability models from data

### Legal and ethical ideas

- Copyright, trademark, and other IP
- Free software/open source
- Privacy and security

### Artificial Intelligence

- The Turing test
- Problem spaces and minimax
- Heuristic search (A\*)

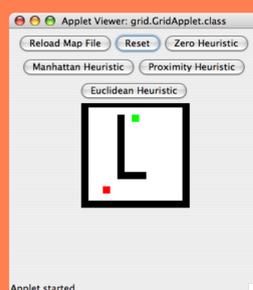
### Human-Computer Interaction

- Affordances and cultural inputs
- Usability design principles
- Prototyping and user testing

### Database systems

- DBMS structure and correctness
- Relational algebra
- Basics of SQL

### A\*: Problem spaces and heuristic search

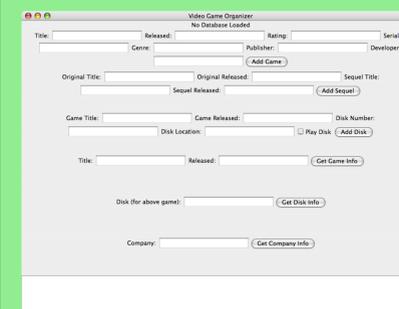


In this project, students implement the A\* shortest-path algorithm, with a variety of admissible heuristics to shape the search process. By dropping it into a GUI, they can visualise the real effects of different choices in the search policy. To the left is a screenshot of the GUI and an initial map; below are that map's exploration according to four different heuristics.

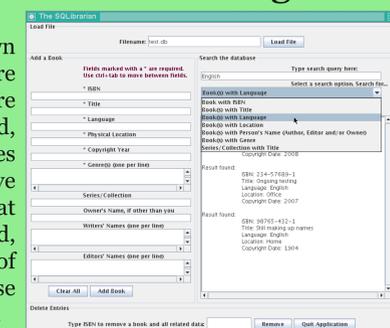


Screenshots used with permission.

### Catalogue your stuff: Database development and interface design



This project makes each student their own customer, building a database that can store information in a domain they care about. There are two phases: first, their general design is approved, and they implement some very simple SQL queries in a minimalist GUI. Then, they choose to improve either the back-end, with complex SQL queries that do significant data manipulation, or the front-end, with a better UI designed according to principles of HCI. To the left, a game catalogue on the database track; to the right, a book catalogue on the UI track.



Screenshots used with permission.

<sup>1</sup> ACM/IEEE Joint Task Force on Computing Curricula. Computing Curricula 2001: Computer Science. 2001.

<sup>2</sup>The ‘breadth-second’ idea is not new; it is mentioned in CC2001 as an alternative to breadth-first, and see also Gray and Frazier, “Introducing computer science after programming”, *Journal of Computing Sciences in Colleges* 18:1, October 2002; and Brazier, Grabowski, and Dietrich “Closing the CS I-II gap: a breadth-second approach”, *33rd ASEE/IEEE Frontiers in Education Conference*, Boulder, 2003.