Curriculum Reform

Department of Computer Sciences
University of Texas at Austin
7/10/08

CS Degrees

- BA with a major in CS
- BS, Option I: Computer Sciences
- BS, Option II: Turing Scholars Honors
- BS, Option III: Computer Sciences Honors (Dean's Scholars Program)
- BS, Option IV: Integrated Program

  (5 year program combined with MS CS)

Reasons for Reform

- CS has become too large to cover in 4 years, especially in application areas
- Consequently, students learn too little about too much
- Employers and graduate schools want many of our graduates to have deeper knowledge and skills

Summary

- Core
  - Shrink the set of required courses
  - Shift the math requirements
    - Less calculus
    - More probability, statistics, linear algebra
- Concentrations
  - Each one is a coherent cluster of electives
  - Most students complete two concentrations
  - Every student completes a research project or software development project

Math Requirements

- BA requirement:
  - Old: Calculus 1 and 2, elective.
  - Proposed: Calculus 1, Matrices, Probability.
- BS requirement:
  - Old: Calculus 1 and 2, Matrices, elective.
  - Proposed: Calculus 1, Matrices, Probability, elective.
- Rationale:
  - Linear algebra, probability more useful in CS than multivariable calculus.
  - Larger variety of mathematics.

Proposed Core

- Reduce the core to 9 classes (down from 11)
- Include a one hour “immigration course”
- Design the courses to fit together coherently

- Core courses in three areas:
  - Programming
  - Systems
  - Theory

Programming Core

Core Programming Classes – ACM Recommendations

- Three introductory courses
  - Plus a discrete math course in parallel
- Six recommended approaches
  - imperative first, objects first, functional first, breadth first, algorithms first, hardware first
- Introductory courses should not be programming fundamentals only.
  - Recommends incorporating topics from discrete math, software engineering, and other areas.

Core Programming Classes
Example of Variation – Languages

- Brown – Java
- Berkeley – Scheme, Scheme, Java
- Caltech – Scheme, multiple languages in second course
- CMU – Java
- Columbia – Java, third course with multiple languages
- Cornell – Java/Matlab, Java/C++, ML/C++/Assembly
- Duke – Java
- Georgia Tech – Python, Java
- Harvard – C/Ruby (some Java, C++)
- Illinois – Java, C++, C
- Maryland – Java, Java, C
- Michigan – C++
- Penn – Java, C/Java/Python
- Princeton – Java
- Purdue – Java
- Rice – Scheme, Java
- Rutgers - Java, Java, Java
- Stanford – Java, C++
- UC Irvine – Java, Scheme
- UCLA – C++
- UCSD – Java, Java/C++
- UMass – Java, C/C++
- USC Java, C++
- Washington – Java, Java
- Wisconsin – Java, Java/C++
- Yale – Scheme, C

Core Programming Classes – Programming 0

- For students with no high school experience in CS
- Programming Fundamentals
- Use a small subset of Java
- Focus on fundamentals of computation, algorithms, and programming
- Ideas stressed through out course:
  - algorithm and program development, testing, predicting program behavior, debugging, reasoning about programs.
- By the end of the course students should be able to design and implement a program that involves an array of objects/structures/records.
  - 200 - 250 lines of code for a good solution.

Core Programming Classes – Programming 0 Topics List

- variables and expressions
- control structures
  - looping, decision making
- Boolean logic
- simple i/o and file i/o,
- structural decomposition
- procedures / methods / functions, parameters, problem generalization
- simple testing and debugging
- arrays
- using, library functions (strings, math functions)
- creating user defined data types
- program and algorithm development
- understand and modify existing code
- quadratic sorting algorithms

Core Programming Classes – Programming 1

- First class for students with some CS from high school
- Data structures and algorithms.
- Ideas stressed throughout course:
  - algorithm and program development, thinking about efficiency of algorithms, experiments on data structures and algorithms to verify performance, data structure algorithms, time and space trade offs, testing, abstraction, debugging, reasoning about programs.
- Continue to use Java

Core Programming Classes – Programming 1 Topics List

- Language review, pointers
- introduction to algorithm analysis
- sorting and searching
- recursion
- data structures
  - array based lists, linked lists, iterators, stacks, queues, trees, hash tables, maps/dictionaries

Core Programming Classes – Programming 2

- Under construction …
- Potential topics and goals:
  - learn a different language or two
  - advanced data structures
    - graphs, heaps, others
  - memory management
  - open ended projects
  - design, implementation, and testing of larger programs
- other possibilities
  - modeling, parallel programming / threads, networked applications, systems programming, programming tools

Computer Systems Core

Goals

- Knowledge
  - How a computer works at a basic level
    - From bits and gates to OS and programming languages
  - Systems principles: abstraction, pipelining, caching, virtualization
- Skills
  - Systems programming (C and assembly)
  - Thinking sequentially and concurrently
  - Reasoning about system performance
  - Practice with real-world systems
- Some inspiration drawn from "Computer Systems, A Programmer's Perspective," by Bryant and O'Hallaron

Systems I Overview (39 lecture hrs)

- Digital logic and digital systems (8)
- Data representation (2)
- ISA/assembly language (5)
  - Encoding, addresses, program/data interpretation
- Programming language primitives (7)
  - Compound data structures, procedure calling, control (loops, if/then/else)
- Machine organization (8)
  - Datapath design, control logic, instruction execution
- Performance principles (2)
  - Clock rate, parallelism, Amdahl's law
- Pipelining principles (4)
  - Control/data hazards, simple branch speculation
- Principles of caching (3)
  - Locality, cache organization

Systems II Overview (35 lecture hrs)

- Concurrency and concurrent programming (12)
  - Threads/locks/monitors
  - Mutual exclusion, safetly/liveness
  - Practice in concurrent programming
- Memory organization and management (4)
  - Memory layout, pointers, heap mgt.
- Virtualization (6)
  - Process organization, isolation
  - Address translation, page tables, TLBs
- File systems and I/O (4)
  - What's a file, what's a directory, transactions
- Security (3)
  - Basic overview, public key encryption, authentication
- Networking (3)
  - Data transmission, internet protocol, reliability
- Systems performance evaluation (3)
  - Throughput/latency, performance models (LogP)

Theory Core

Theory 1: Logic, Sets and Functions

1. Logic (4)
   - Propositional logic (truth tables, inference rules)
   - Predicate logic (inference rules)
   - Formal expression of statements using logical notation
2. Proof Techniques (direct, contradiction, induction) (4)
3. Set Theory (operations) (2)
4. Relations (2)
   - Betweens sets, On sets
   - Representations by matrices and graphs
   - Properties of relations on sets
   - Equivalence relations and partitions
5. Functions (2)
   - Properties
   - Pigeonhole principle

Theory 2

- Under construction …

Theory 3: Algorithms

- Topics
  - Basic graph algorithms
  - Divide and conquer
  - Greedy algorithms
  - Dynamic programming
  - Graph algorithms: MST and shortest paths
  - Data structures
  - NP-completeness
  - Randomized algorithms
  - Approximation algorithms
  - Number theory algorithms/RSA
- Prerequisites: Theory 2, Linear Algebra, and Probability.

Concentration Areas

Degree Requirements

- Students select 7 concentration courses
  - For "specialists"
    - 3 in a "major" concentration area, 2 in a "minor" area
    - 2 electives - can be in major area, minor area, other area
  - For "generalists"
    - 3 "header" courses from 3 different concentration areas
    - 4 electives - from any concentration area or non-area

Concentration Areas

- Theory
- Intelligent Systems
- Computer Systems
- Networking and Security
- SW Development
- Programming Languages
- Scientific and Data Driven Computing
- "Interdisciplinary"

Example Concentration Area

- Networking and Security
  - Computer Networks (header)
  - Computer Security (header)
  - Cryptography
  - Wireless Networks
  - Advanced Computer Networks
  - Applications of Networks and Security (Project)

Example Concentration Area

- Intelligent Systems
  - Overview of Artificial Intelligence (header)
  - Knowledge Representation and Reasoning
  - Machine Learning and Data Mining
  - Robotics
  - Computer Vision
  - Natural Language Processing

Project Class ("Capstone")

- Research project or software development project
- Junior year
- Give students something to brag about