Building a more attractive CS curriculum

Joe Warren, Chair
Department of Computer Science
Rice University
In the late 1990’s, CS was one of the most popular degrees at many universities

Dramatic decline in # of degrees over last decade
Causes of the decline

Three factors contributed to dramatic decrease

- Dot.com crash – many majors were attracted by the lure of quick wealth via startups
- Outsourcing of IT jobs overseas – perception was (and still is) stronger than reality
- Growth of bio-sciences – attractive to students wishing to “solve world’s problems”
Goal – Increase number of CS majors

- Improve PR with potential majors
  - “Why should I major in CS?”
- Build better community for existing majors
  - Undergraduate lunch series – alumni
  - CS lounge on 3rd floor
  - Graduates – exit interview, Facebook
- Revise curriculum to attract more majors
Previous curriculum - programming

- Functional prog (210) - recursion, Scheme
- Imperative prog (212) - object-oriented, Java
- Applied alg and data (314) – team prog, Java
- Senior design - (one of 402, 410, 460)

Related course shared with ECE

- Intro to comp systems (221) – pointers, memory allocation and management, C
Core idea of 210 – Program templates

- Start with recursive data definition
  
  \[
  \text{List} = \text{Null} || \text{Prepend}[\text{elem}, \text{List}] 
  \]

- Build template systematically from definition
  
  \[
  \text{listProc}[l] = \text{If}[\ \text{Empty}[l], \\
  \quad \text{process empty list,} \\
  \quad \text{process and merge } \text{First}[l] \text{ and } \text{Rest}[l]]
  \]

- Fill in yellow areas based on desired action
  
  \[
  \text{reverse}[l] = \text{If}[\ \text{Empty}[l], \text{Null}, \text{Append}[\text{First}[l], \text{Rest}[l]]]
  \]
Strength and weaknesses of 210

- Teaches systematic process for design of small, recursive programs
- Reinforces abstract thinking
- Balances AP curriculum’s emphasis on loop-based imperative prog with recursion

Many potential CS majors decided to major in ECE after taking Comp 210
Ideas for improved CS curricula

- Georgia Tech model - “Threads” curriculum
- 8 diverse themes from CS – intelligence, media, computational modeling, etc.
- Majors focus on 2 threads plus core programming and architecture classes

Increased enrollment at Georgia Tech by 33%
Revised Rice curriculum

Intro course should focus on solving interesting problems, not programming for it’s own sake

Comp 140 – Computational thinking, Python
Comp 160 – Intro to game design, C#
Comp 170 – Intro to bioinformatics, Matlab

Goal – Build bad programmers who are excited to take more courses and major in CS
Effect on mid-level courses

- Fuse old intro courses (210 and 212) into single, second course - Comp 211

- Modify third course (314) to include material not included in 211

Challenge - keep student enthusiastic while not weakening content of previous curriculum
Upper level courses

Senior-design project (now required for BS)
- Comp 402  - Production programming
- Comp 410 – Software engineering
- Comp 460 – Game design for the Xbox 360

New offering – Senior-level open source project in systems
Innovation at the upper level

- Advanced training in parallel programming - mathematical and practical
- Presentation and writing skills – student seminars and colloquia
- Inter-disciplinary teams – projects spanning multiple departments
- More undergrad participation in research
Audience questions

- What skills does industry value?

- What skills should Rice CS be teaching its students?

- What skills should we not bother teaching our students?