CS 4341/6341: Continuous Algorithms

Course overview, policies

- **Instructor**: Prof. Rahul Simha

  `Email: simha@gwu.edu`

- **Time/place**:
  - Class: Fridays, 3.30-6.00pm, Tompkins 201

- **Office Hours**: Wednesdays 4-6pm, SEH 4560 or by appointment.

- **TA:** Shabnam Tafreshi (`shabnamt AT gwmail.gwu...` etc)

- **TA office hours:** TBA

- **Prerequisites**: CS 2113 or equivalent (See undergraduate curriculum).

- **Official course description:** Overview of structures in continuous mathematics from a computational viewpoint. Main topics include simulation, computational modeling, machine learning, neural networks, text classification, statistical language processing, robot control algorithms.

- **My description:** To understand what this course is about and why you should consider taking it, let's first examine the standard CS algorithms course. What do we learn in *that* course? The standard course features data structures (trees, tries, hashing), graph algorithms (DFS, BFS, Dijkstra), discrete optimization and NP-completeness. Almost any textbook on algorithms has these topics at its core. The problem-solving paradigms we learn have to do with divide-and-conquer, dynamic programming, recursion and the use of appropriate data structures.

  Yet, there is a whole world of algorithms that are both important and are fundamentally different. These algorithms include:

  - Algorithms that simulate physical reality, for example the motion of objects or interactions between chemicals, or complex systems such as transportation networks.
  - Algorithms that reason about probability, or that work with probabilistic quantities (We call them simulations).
  - Algorithms that control robots, both at the low level (controlling movement) as well as at the high level (planning a path).

    - Algorithms that involve learning or recognition, for example, face-recognition or handwriting recognition.
    - Algorithms that involve understanding natural languages.

  What is common to all these algorithms, and what differentiates them from algorithms in the standard course, is that they are all based on *continuous* structures or continuous mathematics. Thus, while graph algorithms are based on discrete structures (graphs), a simulation in contrast involves *real numbers*. The purpose of this course is to give you an overview of this area, introduce you to key ideas, and along the way, stimulate interest in the very different types of problems you can solve with this approach.

- **Learning outcomes**: By the end of the course, you should be able to:
    - Develop an understanding of key concepts in areas of continuous mathematics, such as calculus/ODEs, probability, and optimization, useful to computer science.
    - Understand how these concepts are applied to computer science problems.
    - Apply techniques and algorithms learned to computer science problems, especially to the three motivating problem areas of the course.

- **Textbook**: There is no required textbook. All the material is on the course website. For supplemental reading, there's no book that covers all the material we need. Here are some suggestions:
    - *Introduction to Probability Models*. S.Ross. Ross is easily one of the best textbook writers in mathematics, especially in probability. The writing is concise and the examples outstanding.
    - *An Introduction to Computer Simulation Methods: Applications to Physical Systems*. H.Gould, J.Tobochnik and W.Christian. An accessible exposition of simulating physical systems.
    - *Discrete Event Simulation: A First Course*. L.Leemis and S.Park. A nice introduction to discrete-event simulation. (Full disclosure: Larry and Steve were former colleagues.)
    - *Machine Learning*. S.Marsland. This is an entry-level textbook in machine learning, not too detailed yet covering most important topics. For more details there are the well-known books by Bishop and Duda et al.
    - *Neural Networks: Algorithms, Applications and Programming Techniques.* . J.A.Freeman and D.M.Skapura. I found this explanation and development of early neural networks to be easy to follow.

- **Programming load:** The course is not intended to be a heavy load. There will be programming assignments and group projects, but no exams. The idea is to see if we can apply the concepts learned to some practical problems. Students may potentially propose projects.

- **Grads vs. undergrads:** Undergrads will be graded on a different scale, and will possibly work in teams. Yes, we recognize that you are taking an elective course.

- **Coursework and grading:** TBA

- **Assignment submission and late work policy:** Since the courseload is modest, and since much work is team-based, no late submissions will be accepted except for medical emergencies that are substantiated.

- **Email policy:** No debugging by email. Stop by in person - it's much easier that way.

- **Academic Integrity policy:**
    - In this course, you will be expected to work on **all** assigned coursework by yourself, unless otherwise specified by me or circumscribed by teamwork. If you have any questions whatsoever regarding these policies, see me during office hours.
    - You may not, without my permission, exchange course-related code with anyone (including anyone not registered in the course), or download code for use in your coursework, or use material from books other than the textbook. Likewise, you may not look at anyone else's code or show your code to anyone else. Protect your work: for example, be careful not to leave your printouts around.
    - I can't imagine you'll be using a tutor in this course, but if you do, *All tutors for this class need to first register with me, by meeting me during office hours*.
    - If you use material in your assignments that are from outside the course material, then you should be prepared to explain that material. The instructor and TA's reserve the right to question you on your use of extraneous material. Failure to answer such questions might be viewed as grounds for an integrity violation.
    - The Academic Integrity Code will apply to this course. Please read through the code carefully.
    - Penalties for violating the code or the policies described here include failing this course, and are elaborated in the Academic Integrity Code.

- If you have a disability that may effect your participation in this course and wish to discuss academic acommodations, please contact me as soon as possible.

- **Teams:**
    - When pairs of students may work as a team, team members can share code (this is the only exception to the integrity policy above), but will submit individual work.
    - More will be expected of grad teams than individual grads.

- **Attitude:**
  We need to touch upon two attitude-related issues. One, since this is an elective, project-oriented course, you ought to take this opportunity to learn by plunging into the material and by taking on an interesting project that could become a research project, or your senior design project. Second, if you are a math-phobe, you should not let that get in the way of appreciating the material, much of which is based on continuous mathematics but much of which we will cover computationally. More about this in class.

- **Minimum course load:** In a 15-week semester, including exam week, students are expected to spend a minimum of 100 minutes of out-of-class work for every 50 minutes of direct instruction, for a minimum total of 2.5 hours a week. A 3-credit course includes 2.5 hours of direct instruction and a minimum of 5 hours of independent learning, or a minimum of 7.5 hours per week. More information about GWâ€™s credit hour policy can be found at: provost.gwu.edu/policies-forms