Pre-registration Info Session

Computer Science Department
Academic Year 2020-2021

Fall 2020

Core Courses

- **CS134 :: Intro to Computer Science** (72)
  - Duane Bailey and Daniel Aalberts
- **CS136 :: Data Structures & Adv Prog** (60)
  - Bill Lenhart, Aaron Williams
- **CS237 :: Computer Organization** (30)
  - Kelly Shaw
- **CS256 :: Algorithm Design & Analysis** (30)
  - Sam McCauley
- **CS334 :: Principles of Programming Lang** (30)
  - Steve Freund
- **CS361 :: Theory of Computation** (30)
  - Aaron Williams

Elective Courses

- **CS338 :: Parallel Processing** (24)
  - Kelly Shaw
- **CS356T :: Advanced Algorithms** (10)
  - Bill Lenhart
- **CS357 :: Algorithmic Game Theory** (24)
  - Shikha Singh
- **CS373 :: Artificial Intelligence** (24)
  - Andrea Danyluk
- **CS377 :: Human Work in Computational Systems** (24)
  - Molly Feldman

Spring 2021

Core Courses

- **CS134 :: Intro to Computer Science** (75)
  - Duane Bailey, Shikha Singh, Molly Feldman
- **CS136 :: Data Structures & Adv Prog** (60)
  - Bill Lenhart, Sam McCauley
- **CS237 :: Computer Organization** (30)
  - Kelly Shaw
- **CS256 :: Algorithm Design & Analysis** (30)
  - Shikha Singh
- **CS334 :: Principles of Programming Lang** (30)
  - Molly Feldman
- **CS361 :: Theory of Computation** (30)
  - Aaron Williams

Elective Courses

- **CS326 :: Software Methods** (24)
  - Steve Freund
- **CS358 :: Applied Algorithms** (24)
  - Sam McCauley
- **CS374T :: Machine Learning** (10)
  - Andrea Danyluk
- **CS432 :: Operating Systems** (24)
  - Duane Bailey

Winter Study Courses

- **CS10 :: C, Unix and Software Tools**
  - Lida Doret
- **CS11 :: Video Game Appreciation (1972-1992)**
  - Aaron Williams

"Special" Courses

- **CS23 :: R&D Computing**
  - Kelly Shaw
- **CS31 :: Senior Thesis**
  - Steve Freund
- **CS99 :: Independent Study**
  - Steve Freund

"Core"

Required Courses to Graduate (with a CS major)

CS134 :: Introduction to Computer Science

This course introduces students to the science of computation by exploring the representation and manipulation of data and algorithms. We organize and transform information in order to solve problems using algorithms written in a modern object-oriented language. Topics include organization of data using objects and classes, and the description of processes using conditional control, iteration, methods and classes. We also begin the study of abstraction, self-reference, reuse, and performance analysis. While the choice of programming language and application area will vary in different offerings, the skills students develop will transfer equally well to more advanced study in many areas. In particular, this course is designed to provide the programming skills needed for further study in computer science and is expected to satisfy introductory programming requirements in other departments.

CS136 :: Data Structures & Advanced Prog

This course builds on the programming skills acquired in Computer Science 134. It couples work on program design, analysis, and verification with an introduction to the study of data structures. Data structures capture common ways in which to store and manipulate data, and they are important in the construction of sophisticated computer programs. Students are introduced to some of the most important and frequently used data structures: lists, stacks, queues, trees, hash tables, graphs, and files. Students will be expected to write several programs, ranging from very short programs to more elaborate systems. Emphasis will be placed on the development of clear, modular programs that are easy to read, debug, verify, analyze, and modify.

CS237 :: Computer Organization

This course studies the basic instruction set architecture and organization of a modern computer. It provides a programmer's view of how computer systems execute programs, store information, and communicate. Over the semester the student learns the fundamentals of translating higher level languages into assembly language, and the interpretation of machine languages by hardware. At the same time, a model of computer hardware organization is developed from the gate level upward.

CS256 :: Algorithm Design & Analysis

This course investigates methods for designing efficient and reliable algorithms. By carefully analyzing the structure of a problem within a mathematical framework, it is often possible to dramatically decrease the computational resources needed to find a solution. In addition, analysis provides a method for verifying the correctness of an algorithm and accurately estimating its running time and space requirements. We will study several algorithm design strategies that build on data structures and programming techniques introduced in Computer Science 136. These include induction, divide-and-conquer, dynamic programming, and greedy algorithms. Additional topics of study include algorithms on graphs and strategies for handling potentially intractable problems.

CS334 :: Principles of Programming Languages

This course examines the concepts and structures governing the design and implementation of programming languages. It presents an introduction to the concepts behind compilers and run-time representations of programming languages; features of programming languages supporting abstraction and polymorphism; and the procedural, functional, object-oriented, and concurrent programming paradigms. Programs will be required in languages illustrating each of these paradigms.

CS 361 :: Theory of Computation

This course introduces a formal framework for investigating both the computability and complexity of problems. We study several models of computation including finite automata, regular languages, context-free grammars, and Turing machines. These models provide a mathematical basis for the study of computability theory–the examination of what problems can be solved and what problems cannot be solved–and the study of complexity theory–the examination of how efficiently problems can be solved. Topics include the halting problem and the P versus NP problem.

Electives

Fall 2020

CS338 :: Parallel Processing

This course explores different parallel programming paradigms used for writing applications on today's parallel computer systems. The course will introduce concurrency (i.e. multiple simultaneous computations) and the synchronization primitives that allow for the creation of correct concurrent applications. It will examine how a variety of systems organize parallel processing resources and enable users to write parallel programs for these systems. Covered programming paradigms will include multiprogramming with processes, message passing, threading in shared memory multiprocessors, vector processing, graphics processor programming, transactions, MapReduce, and other forms of programming for the cloud. Class discussion is based on assigned readings. Assignments provide students the opportunity to develop proficiency in writing software using different parallel programming paradigms.

CS356T :: Advanced Algorithms

This course explores advanced concepts in algorithm design, algorithm analysis and data structures. Areas of focus will include algorithmic complexity, randomized and approximation algorithms, geometric algorithms, and advanced data structures. Topics will include combinatorial algorithms for packing, and covering problems, algorithms for proximity and visibility problems , linear programming algorithms, approximation schemes, hardness of approximation, search, and hashing.

CS357 :: Algorithmic Game Theory

This course focuses on topics in game theory and mechanism design from a computational perspective. We will explore questions such as: how to design algorithms that incentivize truthful behavior, that is, where the participants have no incentive to cheat? Should we let drivers selfishly minimize their commute time or let a central algorithm direct traffic? Does Arrow's impossibility result mean that all voting protocols are doomed? The overarching goal of these questions is to understand and analyze selfish behavior and whether it can or should influence system design. Students will learn how to model and reason about incentives in computational systems both theoretically and empirically. Topics include auction design, efficiency of equilibria, network games, two-sided markets, crowdsourcing markets, incentives in computational applications such as file sharing and cryptocurrencies, and computational social choice.

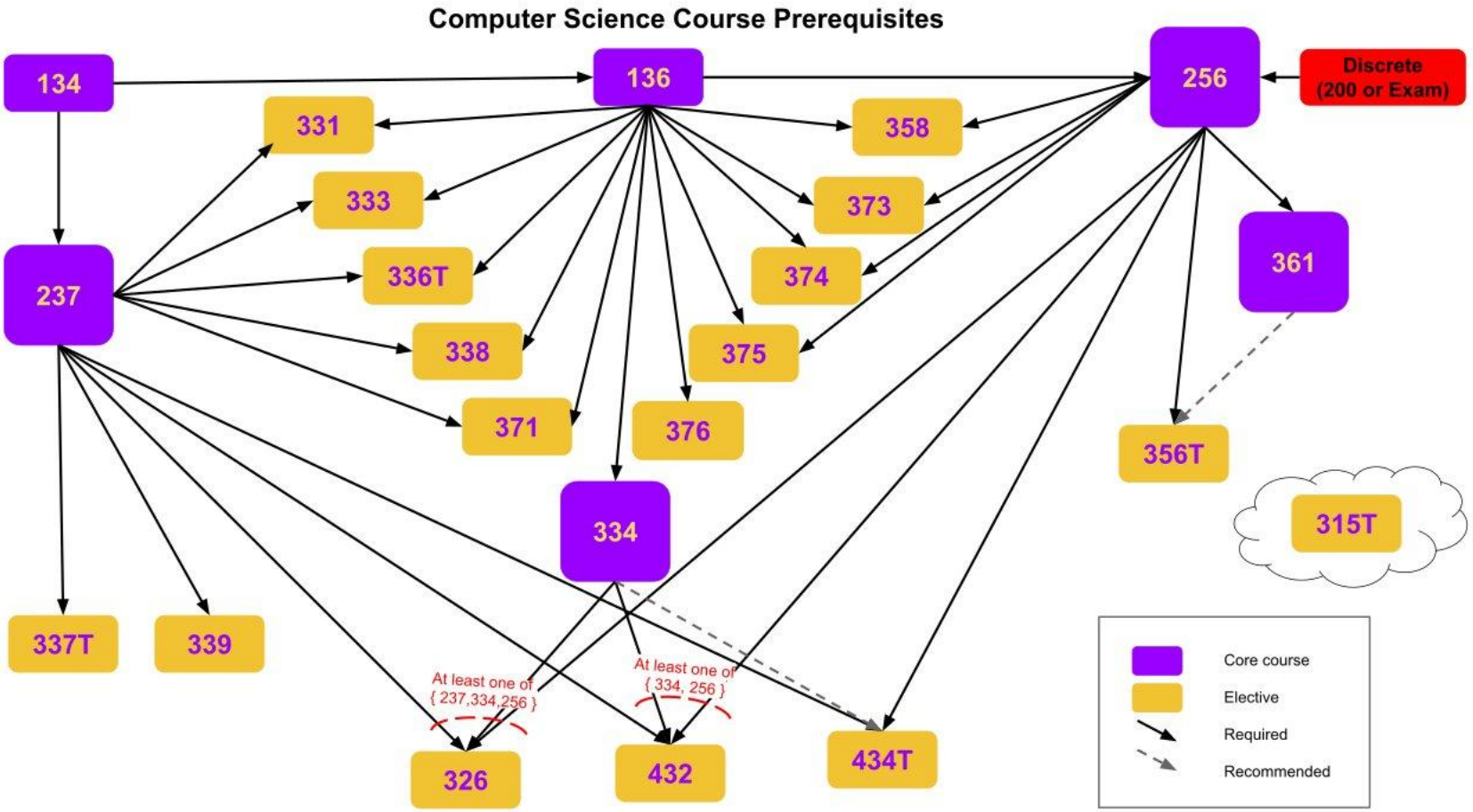
CS373 :: Artificial Intelligence

Artificial Intelligence (AI) has become part of everyday life, but what is it, and how does it work? This course introduces theories and computational techniques that serve as a foundation for the study of artificial intelligence. Potential topics include the following: Problem solving by search, Logic, Planning, Constraint satisfaction problems, Uncertainty and probabilistic reasoning, Bayesian networks, and Automated Learning.

CS377 :: Human Work in Computational Systems

As far as we know, the technological singularity has not yet arrived. Therefore, humans remain a part of our current computation pipeline. However, the role humans play varies greatly: self-driving cars aim to have human involvement only in development and emergencies, whereas educational tools are built for constant human involvement. In this course, we broadly explore human work within computational systems through topics such as crowdsourcing, educational technology, citizen science, human computation, open-source software, artificial intelligence, human-in-the-loop machine learning, micro-labor markets, and online gaming. Students should expect broad exposure to a wide variety of human computing topics.

http://www.cs.cornell.edu/~molly/



http://www.cs.williams.edu/~jannen/teaching/cs-prereqs.svg

Majoring

Planning and Declaring

Declaring the Major

- Must have completed **two CS courses** by end of sophomore year
- Must have satisfied the **Discrete Math requirement**
  - Take (and pass) Math 200 **for credit**, or
  - Pass the Discrete Math Proficiency Exam
- Only the first course in the major may be taken Pass/Fail (except Spring 2020)
- Courses with grades below C- may not be used toward the major (at time of declaration)
- Schedule a meeting with a CS faculty member to discuss your plans
  - Bring an internal transcript, CS major planning sheet, and registrar's major declaration form
- Any CS faculty member can sign your form! (advisors are assigned later)
- After submitting major form, pick up a sticker from Lauren! :-)

Completing the Major

- Intro: CS134, CS136 (or “replacements” if you skipped due to advanced placement)
- Core: CS237, CS256, CS334, CS361
- Electives: ***two*** (or more) CS electives numbered 300+
- Math: Discrete math proficiency, ***one*** (or more) Math/Stat course labeled 200+
- Colloquium: attendance at 20 (or more) colloquia (start now!!!)
- No double counting! If you are majoring in Stat or Math, you cannot count any classes towards both majors

In total:

- 8 computer science courses (core + eligible electives)
- ~2 math courses (200/DPME + 200+)

Useful Forms and Pages

- Major declaration form: https://williamscollege.formstack.com/forms/declare_major
- CS major planning form: http://sysnet.cs.williams.edu/~jeannie/advising/cs-planning-sheet.pdf
- Sample (partially completed) planning form: http://sysnet.cs.williams.edu/~jeannie/advising/cs-planning-sheet-sample.pdf
- CS/Math double major planning form: http://sysnet.cs.williams.edu/~jeannie/advising/cs-math-planning-sheet.pdf
- CS/Stat double major planning form: http://sysnet.cs.williams.edu/~jeannie/advising/cs-stat-planning-sheet.pdf
- Official CSCI catalog: https://catalog.williams.edu/pdf/csci.pdf

Study Away

Study Away Guidelines

- Study Away credit for CS courses is possible at *most* English speaking programs
- Credit may be obtained for non-English programs on a case-by-case basis
- *Common* programs: AIT Budapest, WEPO
- *Common* core courses taken abroad: Theory of Computation, Algorithms, Programming Languages
- Students generally find courses taken abroad to be less rigorous than our courses
- Beware of prerequisites and assumed prior knowledge (particularly for systems-oriented electives)

Study Away Guidelines

- You may receive credit for **no more than 2 CS courses** (elective or core) **and 1 Math course** toward the CS major
- You cannot receive credit for the same course twice (i.e., If you take AI abroad, you cannot take it at Williams)
- You will receive **four colloquium credits for each semester away, up to a total of eight credits**
- You must get approval for courses you plan to take for credit *before* taking them
- To get approval, email the chair or the departmental study away advisor with a syllabus and course description for each course you plan to take
- Upon returning to Williams, submit Study Away Evaluation form to receive credit
- All of this should be done in addition to the official protocol specified by the Study Away office