COMPUTER SCIENCE

Computer Science is the area of study that encompasses all of the theory and practice of computing. The mission of the Department of Computer Science at New Mexico State University is to provide formal education in the core disciplines of computer science, as well as to prepare our graduates for research, development and academic careers. For more information on the Department of Computer Science, please visit the web site https://computerscience.nmsu.edu.

Undergraduate Program Information

The undergraduate computer science programs prepare students for graduate study in computer science and for employment in positions involving the design, construction and application of computer systems. Undergraduate degree programs include a Bachelor of Science (**ABET accredited**), Bachelor of Arts, Bachelor of Science in Cybersecurity, four minor degree tracks, and seven concentrations. The B.S. degree is the traditional computer science degree program, while the B.A. degree offers a more open, flexible degree plan that is easier to combine with studies in other disciplines. The minors are for non-Computer Science majors and offer specialized tracks in algorithm theory, bioinformatics, computer systems and software development. The concentrations are for Computer Science majors and provide a focus on specialized areas such as algorithm theory, artificial intelligence, big data and data science, networking, cybersecurity, human computer interaction, and software development. With technology underpinning almost every area of human endeavor today, students across NMSU should consider pursuing a minor or at least taking some computer science courses. Computer science majors should review their programs of study in consultation with their advisors each semester, preferably using the most recent Undergraduate Catalog.

Graduate Program Information

The department offers both Master of Science and Doctor of Philosophy graduate degrees in computer science, along with a Master of Science in Bioinformatics. We also encourage students in other disciplines to do a graduate minor in computer science. Graduate students typically work closely with a faculty member in a specific area of research. The department offers expertise in several research areas, such as: artificial intelligence and knowledge representation; computer and wireless networks; data mining and machine learning; game design and human-computer interaction; bioinformatics; high performance computing; software engineering and programming languages; theory of computing; and assistive technologies.

A number of laboratories have been established to coordinate research activities, including

- the Knowledge representation, Logic and Advanced Programming (KLAP) lab;
- the Play and Interactive Experiences for Learning (PIxL) lab;
- the Knowledge Discovery and Data Mining (KDD) lab;
- the Programming Languages, Environments, and Automated Software Engineering (PLEASE) lab;
- the Bioinformatics Research lab;
- the Multimedia Management lab;
- the Cryptography, Privacy and Security Research (CrySPR) lab; and
- the Network and Systems Optimization Lab (NSOL).

Department members are also directing the iCREDITS interdisciplinary Center of Research Excellence in Design of Intelligent Technologies for Smartgrids, offering educational and research opportunities in smartgrids.

Entrance Requirements for Graduate Study in Computer Science

The Graduate Record Exam (GRE) General Test is not required for admission; however, high GRE scores will strengthen a candidate's application and are highly regarded in the awarding of Graduate Assistantships. To be admitted without undergraduate deficiencies, an entering student must have completed undergraduate preparation substantially equivalent to that required for the Bachelor of Science degree in Computer Science at New Mexico State University; in particular, this includes courses equivalent to

Prefix	Title	Credits
C S 172	Computer Science I	4
C S 271	Object Oriented Programming	4
C S 272	Introduction to Data Structures	4
C S 273	Machine Programming and Organization	4
C S 278	Discrete Mathematics for Computer Science	4
C S 370	Compilers and Automata Theory	4
C S 371	Software Development	4
C S 471	Programming Language Structure I	3
C S 474	Operating Systems I	3

Deficiencies should be satisfied as early in the student graduate program as possible, through the regular undergraduate courses, the C S 460 - C S 468 transition courses, or through tests administered by faculty members in the relevant areas. Students should consult with their Graduate Advisor to address issues related to deficiencies. Deficiencies are also assigned to applicants whose transcripts denote low grades in selected areas. Admission is often denied to candidates with little background in Computer Science. Instructions for prospective applicants can be found at https://computerscience.nmsu.edu.

Entrance Requirements for Graduate Study in Bioinformatics

The Graduate Record Exam (GRE) General Test is not required for admission; however, high GRE scores will strengthen a candidate's application and are highly regarded in the awarding of Graduate Assistantships. Students wishing to enroll in the Master program in Bioinformatics must meet the following criteria:

1. Hold a BS degree, from an accredited institution of higher learning, in either a computational field (e.g., Computer Science) or in life sciences (preferably Biology, Biochemistry, or Environmental Sciences)
2. Hold a minimum grade point average of 3.2

Applicants will be expected to provide a Career statement, motivating the interest in bioinformatics and a minimum of three letters of reference.

Graduate Assistantships

Graduate assistantships (in the form of Teaching and Research assistantships) are expected to be available during the academic year. Inquiries should be addressed to the departmental Graduate Committee. Research assistantships are available at the discretion of individual research project leaders in the Department or elsewhere on campus.

Submitting detailed vitae, letters of reference, and GRE test scores are encouraged when applying for any assistantship.

Degrees for the Department

Bachelor Degree(s) & Dual Degree(s)

- Computer Science (Algorithm Theory) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-algorithm-theory-bachelor-science/)
- Computer Science (Artificial Intelligence) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-artifical-intelligence-bachelor-science/)
- Computer Science (Big Data and Data Science) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-big-data-science-bachelor-science/)
- Computer Science (Computer Networking) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-computer-networking-bachelor-science/)
- Computer Science (Cybersecurity) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-cybersecurity-bachelor-science/)
- Computer Science (Human Computer Interaction) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-human-computer-interaction-bachelor-science/)
- Computer Science (Secondary Education) - Bachelor of Arts (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-secondary-education-bachelor-arts/)
- Computer Science (Software Development) - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-software-development-bachelor-science/)
- Computer Science - Bachelor of Arts (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-bachelor-arts/)
- Computer Science - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-bachelor-science/)
- Computer Science - Bachelor of Science/Master of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-bachelor-science-master-science/)
- Cybersecurity - Bachelor of Science (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/cybersecurity-bachelor-science/)

Master Degree(s)

- Bioinformatics - Master of Science (https://catalogs.nmsu.edu/nmsu/graduate-school/bioinformatics-master-science/)
- Computer Science - Master of Science (https://catalogs.nmsu.edu/nmsu/graduate-school/computer-science-master-science/)
- Data Analytics (Digital Agriculture) - Master of Data Analytics (Online) (https://catalogs.nmsu.edu/global/nmsu-global/data-analytics-digital-agriculture-mda-online/)
- Data Analytics - Master of Data Analytics (https://catalogs.nmsu.edu/nmsu/graduate-school/data-analytics-master-data-analytics/)
- Data Analytics - Master of Data Analytics (Online) (https://catalogs.nmsu.edu/global/nmsu-global/data-analytics-mda-online/)

Doctoral Degree(s)

- Computer Science - Doctor of Philosophy (https://catalogs.nmsu.edu/nmsu/graduate-school/computer-science-doctor-philosophy/)

Minors for the Department

A student cannot earn more than one of the undergraduate minors unless they pass at least 6 credits in the second minor beyond the requirements of the first minor. The maximum number of undergraduate minors that a student may earn is two. Most courses for the minors listed below have prerequisites. Please check the undergraduate catalog for individual course prerequisites. Students interested in pursuing a computer science minor are encouraged to pick up more information at the departmental office.

- Algorithm Theory - Undergraduate Minor (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/algorithm-theory-undergraduate-minor/)
- Bioinformatics (with Computer Science) - Graduate Minor (https://catalogs.nmsu.edu/nmsu/graduate-school/bioinformatics-computer-science-graduate-minor/)
- Bioinformatics - Undergraduate Minor (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/bioinformatics-undergraduate-minor/)
- Computer Science (Secondary Education) - Bachelor of Arts (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-science-secondary-education-bachelor-arts/)
- Computer Science - Graduate Minor (https://catalogs.nmsu.edu/nmsu/graduate-school/computer-science-graduate-minor/)
- Computer Systems - Undergraduate Minor (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/computer-systems-undergraduate-minor/)
- Software Development - Undergraduate Minor (https://catalogs.nmsu.edu/nmsu/arts-sciences/computer-science/software-development-undergraduate-minor/)

Faculty

Professor Son Tran, Department Head

Professors Cook, Cao, Pontelli, Misra, Song, Tran; ***Associate Professors*** Pivkina, Toups; ***Assistant Professors*** Hamilton, Le, Nagarkar, Reynolds, Vishwanathan, Wang; ***College Associate Professor*** Cooper

H. Cao, Ph.D. (Hong-Kong)– data mining, databases, data integration, applied machine learning; J. Cook, Ph.D. (Colorado)– software engineering, high performance computing; W. Hamilton, Ph.D. (Texas A&M)–media design, online communities, online education, and video game design/culture; T. Le, Ph.D. (Singapore Management University, Singapore)– data mining and machine learning, dimensionality reduction, visualization, topic models, embedding and generative models; S. Misra, Ph.D. (Arizona State)– communication networks, social networks, high performance computing, security and privacy; P. Nagarkar, Ph.D. (Arizona State)– query optimization, indexing, data analytics, big data; I. Pivkina, Ph.D. (Kentucky)– artificial intelligence, computer science education, data mining; E. Pontelli, Ph.D. (New Mexico State)– parallel processing, logic programming, knowledge representation, bioinformatics, assistive technologies; J. Reynolds (UIUC)-network and security; M. Song, Ph.D. (Washington)– statistical computing, systems biology, bioinformatics, computer vision; P. Toups Dugas, Ph.D. (Texas A&M)– digital games, human-computer interaction, mixed reality; S. Tran, Department Head, *Ph.D. (Texas-El Paso)–artificial intelligence,* *knowledge representation, planning, logic programming, non-monotonic reasoning; R. Vishwanathan, Ph.D. (North Texas)– cryptography, theoretical and applied, security, privacy; T. Wang, Ph.D. (University of South Florida)– cyber-physical system security, web security, network security.*

College Faculty:

S. Cooper, Ph.D. (New Mexico State)– computer networks

Computer Science Courses

C S 111. Computer Science Principles
4 Credits (3+2P)
This course provides a broad and exciting introduction to the field of computer science and the impact that computation has today on every aspect of life. It focuses on exploring computing as a creative activity and investigates the key foundations of computing: abstraction, data, algorithms, and programming. It looks into how connectivity and the Internet have revolutionized computing and demonstrates the global impact that computing has achieved, and it reveals how a new student in computer science might become part of the computing future. May be repeated up to 4 credits.
Prerequisite: MATH 1215 or higher.

Learning Outcomes

1. Identify and differentiate programming constructs like IF, FOR, and WHILE
2. Convert numbers between Hexadecimal, Binary and Decimal
3. Write pseudo code to manipulate a robot
4. Use an ASCII table to translate HEX strings into characters
5. Encrypt and Decrypt simple messages with a Caeser Cypher

C S 117. Introduction to Computer Animation
3 Credits (3)
Introductory course for learning to program with computer animation as well as learning basic concepts in computer science. Students create interactive animation projects such as computer games and learn to use software packages for creating animations in small virtual worlds using 3D models. Recommended for students considering a minor/major in computer science or simply interested in beginning computer animation or programming.

C S 151. C++ Programming
3 Credits (2+2P)
Introduction to object-oriented programming in the C++ language. The focus will be on preparing students to use C++ in their own areas. No prior programming experience is required. Taught with C S 451.
Prerequisite: MATH 1215 or higher.

Learning Outcomes

1. Use various data types and the corresponding operations.
2. Write C++ programs that contain expressions, program control, functions, arrays, and input/output
3. Explain basic object-oriented programming concepts.
4. Demonstrate proficiency in using classes, inheritance, pointers, streams, and recursion

C S 152. Java Programming
3 Credits (2+2P)
Programming in the Java language. May be repeated up to 3 credits.
Prerequisite(s): MATH 1215 or higher.

C S 153. Python Programming I
3 Credits (3)
This course is an introduction to programming in the Python language, covering fundamental scripts, data types and variables, functions, and simple object creation and usage. The focus will be on preparing students to use Python in their own areas. No prior programming experience is required. Taught with C S 453.
Prerequisite: MATH 1215 or higher.

Learning Outcomes

1. Develop an algorithm to solve a problem
2. Demonstrate the ability to use Python data types: int, float, strings, and lists; and the built-in functions associated with those data types
3. Edit and debug programs using the Spyder IDE for Python
4. Implement algorithms using the Python features of assignment, input, output, branches, loops, and functions
5. Explain the fundamental concepts of object-oriented programming with Python
6. Design and implement Python classes based on given attributes and behaviors
7. Work with existing Python modules such as math, random, and os
8. Write Python programs that input data from files and store results in files

C S 154. Python Programming II
3 Credits (3)
This course covers advanced Python programming, including classes, objects, and inheritance, embedded programming in domain applications, database interaction, and advanced data and text processing. The focus will be on preparing students to use Python in their own areas.
Prerequisite(s): C S 153 or C S 453.

C S 158. R Programming I
3 Credits (3)
This course is an introduction to data processing in the R language, covering fundamental script configuration, data types and data collections, R control structures, and basic creation of graphs and data visualizations. This course will not focus on the statistical capabilities of R, though some basic statistical computations will be used.
Prerequisite(s): MATH 1220G.

C S 171G. Modern Computing in Practice
4 Credits (3+2P)
This course provides a survey of practical and theoretical foundations for how computers work and how they support fundamental organizational needs. The course covers broad aspects of the hardware, software, and mathematical basis of computers. Lab assignments provide hands-on applications to use simple basic software tools to write simple programs, build and edit websites, analyze data with spreadsheets, choose an office productivity suite, and demonstrate computer literacy to potential employers. May be repeated up to 4 credits.
Prerequisite: MATH 1130G or MATH 1215 or higher.

Learning Outcomes

1. Students will create simple python programs using conditional statements and loops
2. Students will analyze data with spreadsheet formulas, charts, and tools
3. Students will create and publish a personal website using website building tools
4. Students will edit HTML and CSS to format a website manually
5. Students will practice the skill of performing software QA and providing actionable feedback to developers
6. Students will become aware of common cybersecurity risks

7. Students will learn basic vocabulary and context for broad aspects of hardware, software, and computer science theory such as Security, Privacy, Cloud Computing, the Internet, the Web, Operating Systems, Discrete Math, and Information Systems
8. Students will be exposed to various sub-fields of CS including artificial intelligence, security, data analytics, UX, web development, and QA testing
9. Students will reason about the societal impacts of technology 1
10. Students will incorporate their new knowledge and skills into their resume

C S 172. Computer Science I
4 Credits (3+2P)
Computational problem solving; problem analysis; implementation of algorithms using Java. Object-oriented concepts, arrays, searching, sorting, and recursion. Taught with C S 460. May be repeated up to 4 credits.
Prerequisite: (A C- or better in either MATH 1250G or (MATH 1430G or higher)) OR (A C- or better in MATH 1220G and a 1 or better in the CS Placement Test) OR (A C- or better in MATH 1220G and a C- or better in C S 111).
Learning Outcomes
1. Develop algorithms to solve problems
2. Implement algorithms using the fundamental programming features of sequence, selection, iteration, and recursion
3. Apply an understanding of primitive and object data types
4. Design and implement classes based on given attributes and behaviors
5. Explain the fundamental concepts of object-oriented programming,

C S 209. Special Topics.
1-3 Credits
May be repeated for a maximum of 12 credits.

C S 271. Object Oriented Programming
4 Credits (3+2P)
Introduction to problem analysis and problem solving in the object-oriented paradigm. Practical introduction to implementing solutions in the C++ language. Pointers and dynamic memory allocation. Hands-on experience with useful development tools. Taught with C S 462. May be repeated up to 4 credits.
Prerequisite: At least a C- in C S 172 or ENGR 140.
Learning Outcomes
1. Develop an algorithm to solve a problem.
2. Implement algorithms using the C and C++ languages including imperative and object-oriented language features.
3. Beyond what was learned in C S 172, E E 112, or E E 161 demonstrate a noticeable increase in understanding of problem analysis and program design.
4. Demonstrate proficiency in using control structures including if statements (single selection), switch (multiple selection), and loops (repetition).
5. Demonstrate proficiency in using arrays and functions
6. Create UML class and relationship diagrams.
7. Design a class to model a real-world person, place, thing, or event.
8. Use editing and debugging software to create, debug, and test C and C++ programs.
9. Understand the basic terminology used in object-oriented programming. 1
10. Create a make file to build an executable from a set of C or C++ source files.

C S 272. Introduction to Data Structures
4 Credits (3+2P)
Design, implementation, use of fundamental abstract data types and their algorithms: lists, stacks, queues, deques, trees; imperative and declarative programming. Internal sorting; time and space efficiency of algorithms. Taught with C S 463.
Prerequisite: At least a C- in C S 172, or placement.
Learning Outcomes
1. Be able to implement and use lists
2. Be able to implement and use stacks
3. Be able to implement and use queues
4. Be able to implement and use trees
5. Be able to perform the run time analysis of basic algorithms using Big O notation
6. Be able to implement, use, and analyze searching algorithms
7. Be able to solve a problem recursively
8. Take a problem statement from a user and convert it into a Java program that fulfills the user's needs
9. Create object oriented Java classes that effectively separate and hide implementation details from client applications

C S 273. Machine Programming and Organization
4 Credits (3+2P)
Computer structure, instruction execution, addressing techniques; programming in machine and assembly languages. Taught with C S 464. May be repeated up to 4 credits.
Prerequisite: At least a C- in C S 172 or ENGR 140.
Learning Outcomes
1. Describe the architecture of a microcontroller, the interconnections between the components, and the basic units inside the CPU
2. Use signed and unsigned numbers, the associated branching instructions, and the corresponding flags in the status register
3. Explain immediate, direct, indirect addressing modes, their opcode and operands, and their utilities
4. Map high-level programming language features to assembly instructions, including loops, conditionals, procedure calls, value and reference parameter passing, return values, and recursion
5. Interface with I/O devices including LED and sensors via digital input and output, and analog-to-digital conversion
6. Program timers/counters and interrupts to control real-time applications
7. Design an assembly program

C S 278. Discrete Mathematics for Computer Science
4 Credits (3+2P)
Discrete mathematics required for Computer Science, including the basics of logic, number theory, methods of proof, sequences, mathematical induction, set theory, counting, and functions. Taught with C S 465.
Prerequisite: At least C- in C S 172.
Learning Outcomes
1. Use logic to specify precise meaning of statements, demonstrate the equivalence of statements, and test the validity of arguments
2. Construct and recognize valid proofs using different techniques including the principle of mathematical induction

3. Use summations, formulas for the sum of arithmetic and geometric sequences
4. Explain and apply the concepts of sets and functions
5. Apply counting principles to determine the number of various combinatorial configurations

C S 343. Algorithm Design & Implementation
3 Credits (3)
Introduction to efficient data structure and algorithm design. Basic graph algorithms. Balanced search trees. Classic algorithm design paradigms: divide-and-conquer, greedy scheme, and dynamic programming. Taught with C S 493.
Prerequisite: At least a C- in C S 272, or consent of instructor.
Learning Outcomes
1. Be able to use and implement sorting algorithms
2. Be able to design and implement graph algorithms
3. Be able to design and implement algorithms using the divide-and-conquer technique
4. Be able to design and implement algorithms using the greedy technique
5. Be able to design and implement algorithms using the dynamic programming technique
6. Be able to use and implement balanced search trees
7. Be able to use and implement hashing techniques
8. Be able to perform the run time analysis of basic algorithms using Big O notation

C S 370. Compilers and Automata Theory
4 Credits (3+2P)
Methods, principles, and tools for programming language processor design; basics of formal language theory (finite automata, regular expressions, context-free grammars); development of compiler components. Taught with C S 466.
Prerequisite: At least a C- in C S 271, C S 272, and C S 273.
Learning Outcomes
1. Understand the language theory concepts of regular languages, context free languages, regular expressions, context free grammars, and formal language hierarchy
2. Use Thompson's construction to convert from regular expression to NFA, and subset construction to convert from NFA to DFA
3. Apply recursive descent parsing in programming a parser of a small grammar
4. Understand the ideas in LL and LR parsing of context-free language classes
5. Understand and use table-driven top-down (LL(1)) and bottom up (SLR) parsing to parse a sentence

C S 371. Software Development
4 Credits (3+2P)
Software specification, design, testing, maintenance, documentation; informal proof methods; team implementation of a large project. Taught with C S 468.
Prerequisite: At least a C- in C S 271 and C S 272.
Learning Outcomes
1. Understand and explain the activites and structure of different styles of software development processes, including waterfall, (spiral,) iterative, and agile methodologies
2. Apply requirements knowledge and techniques to create functional and non-functional requirements for a software system
3. Apply high and low level design ideas to create an object-oriented design of a software system
4. Use good design and programming ideas to implement individual and team software systems in compiled OOP languages
5. Apply white and black box testing techniques and tools to individual and team software development
6. Use UML class diagrams (and sequence diagrams) to capture aspects of system design and/or requirements (domain)
7. Use practical software development tools, including version control systems, automated build tools, and testing tools

C S 372. Data Structures and Algorithms
4 Credits (3+2P)
Introduction to efficient data structure and algorithm design. Order notation and asymptotic run-time of algorithms. Recurrence relations and solutions. Abstract data type dynamic set and data structures based on trees. Classic algorithm design paradigms: divide-and-conquer, dynamic programming, greedy algorithms. Taught with C S 469. May be repeated up to 4 credits.
Prerequisite: At least a C- in C S 272 and C S 278.
Learning Outcomes
1. Analyze the growth of functions via asymptotic notation
2. Evaluate the asymptotic running time of a given algorithm
3. Solve recurrence relations of the kinds encountered in algorithm analysis
4. Design algorithms using the divide-and-conquer technique
5. Design algorithms using the greedy technique
6. Design algorithms using the dynamic-programming technique
7. Use and analyze data structures based on trees
8. Analyze the design, correctness, and time complexity of basic graph algorithms

C S 380. Introduction to Cryptography
3 Credits (3)
The course covers basic cryptographic primitives, such as symmetric, public-key ciphers, digital signature schemes, and hash functions, and their mathematical underpinnings. Course helps students understand basic notions of security in a cryptographic sense: chosen plaintext and chosen ciphertext attacks, games, and reductions. Course also covers computational number theory relevant to cryptography. Consent of Instructor required. Taught with: C S 525.
Prerequisite: C S 278 (or equivalent) with a C or better.
Learning Outcomes
1. Describe basic cryptographic primitives, including symmetric ciphers, asymmetric ciphers, digital signatures, message authentication codes, and hash functions.
2. Understand the mathematical, fundamental underpinnings of cryptography, and how to reason about the security of crypto primitives: indistinguishability (IND) properties of ciphertexts, CPA/CCA games, and reductions to fundamental math assumptions;
3. Be able to discuss number theory/algebra underpinning the design of cryptographic primitives, in some depth.

C S 381. Principles of Virtual Reality
3 Credits (3)
This course is an introduction to building systems and doing research in / on virtual reality. We cover system design, development, and evaluation, with an emphasis on recent research in the space. We cover a range of methods, qualitative and quantitative, in order to develop insights into effective VR designs. Students in this class will develop a foundation in

VR development; learn about current topics in VR; and design, develop, evaluate, and report on a VR system. Consent of Instructor required.
Prerequisite: C S 485.
Learning Outcomes
1. Design and develop systems in virtual reality.
2. Understand the variety of development techniques in VR.
3. Understand the state-of-the-art in VR systems.
4. Communicate understanding of people, designs, and evaluations through presentations, demos, and/or reports.

C S 382. Modern Web Technologies
3 Credits (3)
In this course, we will take a full-stack approach to modern web application design. We will start with the fundamentals including HTML5, CSS3, Javascript, JSON, and the underlying networking concepts and protocols driving the modern web. We will then move on to more advanced topics including javascript backend development with Node.js, NoSQL database design with MongoDB, cloud computing, and re- sponsive web design. Finally, we cover advanced topics including the design and im- plementation of browser extensions and real-time web technologies like WebRTC and WebSockets. Consent of Instructor required. Taught with: C S 532.
Learning Outcomes
1. Understand the fundamental technologies and operation of the web.
2. Design and develop responsive interactive web sites.
3. Deploy web applications on Cloud Computing Platforms.
4. Leverage modern tools and packages to develop full stack web applications.
5. Be fluent in the application of emerging web technologies like browser extensions, WebSockets, and WebRTC.
6. Use existing materials and references on the web to learn new skills.

C S 383. Introduction to Deep Learning
3 Credits (3)
The course covers basic concepts of neural networks which include transition of classical machine learning to hierarchical feature learning, feedforward networks, regularization, optimization, hyperparameter tuning, deep convolutional networks and their applications in computer vision, deep sequence models, and deep generative models. Taught together with C S 533. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 272 or C S 153, and C S 278, or consent of instructor.
Learning Outcomes
1. Have significant familiarity with different state-of-the-art theories and practices of deep learning.
2. Be able to apply deep learning to a variety of tasks suitable for data science-based projects of academia and industry.
3. Understand much of the current literature on the topic, review papers, and extend their knowledge through further study.
4. Design and evaluate novel deep learning models.
5. Train and test deep learning models on real-life and benchmark datasets using Python libraries such as TensorFlow and PyTorch.

C S 384. Graph Data Mining
3 Credits (3)
The course covers graph terminology, representation, and techniques to extract patterns from large graph data. The topics include random and scale-free graph generation, link analysis (PageRank), graph representation learning, graph neural networks, deep graph generation, community detection, frequent subgraph mining, graph classification, influence maximization, and knowledge graph mining. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 272 or C S 153, and C S 278, or consent of instructor.
Learning Outcomes
1. Have significant familiarity with different state-of-the-art theories and practices of graph data mining
2. Graph representation and graph querying using graph manipulating toolbox/library
3. Use random and scale-free graph models to generate graphs and visualize complex network properties
4. Apply algorithms such as PageRank, spectral clustering, and non-negative matrix factorization
5. Implement graph representation learning algorithms and graph neural networks
6. Understand much of the current literature on the topic, review papers, extend their knowledge through further study, and present findings of the papers.

C S 409. Independent Study
1-6 Credits (1-6)
Faculty supervised investigation, to culminate in a written report. May be repeated up to 6 credits.
Prerequisite(s): Written agreement with faculty supervisor.

C S 419. Computing Ethics and Social Implications of Computing
1 Credit (1)
An overview of ethics for computing majors includes: history of computing, intellectual property, privacy, ethical frameworks, professional ethical responsibilities, and risks of computer-based systems.
Prerequisite: At least a C- in C S 371.
Learning Outcomes
1. Understand the fundamental technologies and operation of the web.
2. Design and develop responsive interactive web sites.
3. Deploy web applications on Cloud Computing Platforms.
4. Leverage modern tools and packages to develop full stack web applications.
5. Be fluent in the application of emerging web technologies like browser extensions, WebSockets, and WebRTC.
6. Use existing materials and references on the web to learn new skills.

C S 448. Senior Project
4 Credits (4)
Capstone course in which C S majors work in teams and apply computer science skills to complete a large project. Restricted to: C S majors.
Prerequisite: At least a C- in C S 370 and C S 371.
Learning Outcomes
1. Apply design and development principles in the construction of software systems of varying complexity
2. Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices
3. Design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
4. Use current techniques, skills, and tools necessary for computing practice
5. Analyze a problem, and identify and define the computing requirements appropriate to its solution

6. Function effectively as teams to accomplish a common goal
7. Communicate effectively with a range of audiences

C S 449. Senior Thesis
4 Credits (4)
Capstone course in which C S majors apply computer science skills to complete a research project, culminating in a written thesis report. Restricted to: C S majors.
Prerequisite: At least a C- in C S 370 and C S 371.
Learning Outcomes
1. Apply design and development principles in the construction of software systems of varying complexity
2. Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices
3. Design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
4. Use current techniques, skills, and tools necessary for computing practice
5. Analyze a problem, identify, and define the computing requirements appropriate to its solution
6. Communicate effectively with a range of audiences via presentations and technical reports

C S 451. C++ Programming
3 Credits (3)
Programming in the C++ language. Taught with C S 151. More advanced than C S 151. Recommended for nonmajors only. Not for CS undergraduate students. May be repeated up to 3 credits.
Learning Outcomes
1. Use various data types and the corresponding operations.
2. Write C++ programs that contain expressions, program control, functions, arrays, and input/output.
3. Explain basic object-oriented programming concepts.
4. Demonstrate proficiency in using classes, inheritance, pointers, streams, and recursion.

C S 452. Java Programming
3 Credits (2+2P)
Programming in the Java language. More advanced than C S 152. Recommended for nonmajors only. Not for CS undergraduate standing. May be repeated up to 3 credits.

C S 453. Python Programming I
3 Credits (3)
This course is an introduction to programming in the Python language, covering fundamental scripts, data types and variables, functions, and simple object creation and usage. The focus will be on preparing students to use Python in their own areas. No prior programming experience is required. Taught with C S 153. More advanced than C S 153.
Learning Outcomes
1. Develop an algorithm to solve a problem
2. Demonstrate the ability to use Python data types: int, float, strings, and lists; and the built-in functions associated with those data types
3. Edit and debug programs using the Spyder IDE for Python
4. Implement algorithms using the Python features of assignment, input, output, branches, loops, and functions
5. Explain the fundamental concepts of object-oriented programming with Python
6. Design and implement Python classes based on given attributes and behaviors
7. Work with existing Python modules such as math, random, and os
8. Write Python programs that input data from files and store results in files

C S 454. Python Programming II
3 Credits (3)
This course covers advanced Python programming, including classes, objects, and inheritance, embedded programming in domain applications, database interaction, and advanced data and text processing. The focus will be on preparing students to use Python in their own areas. For graduate students only. Has more advanced work than C S 154, and does not count towards CS major requirements. Not for CS undergraduate students. May be repeated up to 3 credits. Restricted to: exclude C S majors.
Prerequisite(s): C S 153 or C S 453.

C S 458. R Programming I
3 Credits (3)
This course is an introduction to data processing in the R language, covering fundamental script configuration, data types and data collections, R control structures, and basic creation of graphs and data visualizations. This course will not focus on the statistical capabilities of R, though some basic statistical computations will be used. For graduate students only. Has more advanced work than C S 158. Does not count towards CS major requirements. May be repeated up to 3 credits.
Prerequisite(s): Good understanding of college algebra or higher.

C S 460. Computer Science I Transition
3 Credits (3)
Computational problem solving; problem analysis; implementation of algorithms. Recursive structures and algorithms. For C S graduate students only; cannot be used to meet a C S student's program of study. Taught with C S 172.
Learning Outcomes
1. Develop algorithms to solve problems
2. Implement algorithms using the fundamental programming features of sequence, selection, iteration, and recursion
3. Apply an understanding of primitive and object data types
4. Design and implement classes based on given attributes and behaviors
5. Explain the fundamental concepts of object-oriented programming

C S 462. Object Oriented Programming Transition
3 Credits (3)
Introduction to problem analysis and problem solving in the object-oriented paradigm. Practical introduction to implementing solutions in the C++ language. Hands-on experience with useful development tools. Cannot be used in a C S student's program of study. Consent of Instructor required. Taught with C S 271.
Prerequisite: At least a C- in C S 172 or C S 460 or consent of instructor.
Learning Outcomes
1. Develop an algorithm to solve a problem.
2. Implement algorithms using the C and C++ languages including imperative and object-oriented language features.
3. Demonstrate a noticeable increase in understanding of problem analysis and program design beyond what was learned in C S 172, E E 112, or E E 161
4. Demonstrate proficiency in using control structures including if statements (single selection), switch (multiple selection), and loops (repetition).

5. Demonstrate proficiency in using arrays and functions.
6. Create UML class and relationship diagrams.
7. Design a class to model a real-world person, place, thing, or event.
8. Use editing and debugging software to create, debug, and test C and C++ programs.
9. Understand the basic terminology used in object-oriented programming. 1
10. Create a make file to build an executable from a set of C or C++ source files.

C S 463. Introduction to Data Structures Transition
3 Credits (3)
Design, implementation, use of fundamental abstract data types and their algorithms: lists, stacks, queues, deques, trees; imperative and declarative programming. Internal sorting; time and space efficiency of algorithms. Cannot be used in a C S student's program of study. Consent of Instructor required. Taught with C S 272.
Prerequisite: At least a C- in C S 172 or C S 460 or consent of instructor.
Learning Outcomes
1. Be able to implement and use lists
2. Be able to implement and use stacks
3. Be able to implement and use queues
4. Be able to implement and use trees
5. Be able to perform the run time analysis of basic algorithms using Big O notation
6. Be able to implement, use, and analyze searching algorithms
7. Be able to solve a problem recursively
8. Take a problem statement from a user and convert it into a Java program that fulfills the user's needs
9. Create object oriented Java classes that effectively separate and hide implementation details from client applications

C S 464. Machine Programming and Organization Transition
3 Credits (3)
Computer structure, instruction execution, addressing techniques; programming in machine and assembly languages. Cannot be used in a C S student's program of study. Consent of Instructor required. Taught with C S 273.
Prerequisite: At least a C- in C S 172 or C S 460 or consent of instructor.
Learning Outcomes
1. Describe the architecture of a microcontroller, the interconnections between the components, and the basic units inside the CPU
2. Use signed and unsigned numbers, the associated branching instructions, and the corresponding flags in the status register
3. Explain immediate, direct, indirect addressing modes, their opcode and operands, and their utilities
4. Map high-level programming language features to assembly instructions, including loops, conditionals, procedure calls, value and reference parameter passing, return values, and recursion
5. Interface with I/O devices including LED and sensors via digital input and output, and analog-to-digital conversion
6. Program timers/counters and interrupts to control real-time applications
7. Design an assembly program

C S 465. Discrete Math for Computer Science Transition
3 Credits (3)
Logical connectives, sets, functions, relations, graphics, trees, proofs, induction, and application to computer science. For C S graduate students only. Cannot be used in a C S student's program of study. Consent of Instructor required. Taught with C S 278.
Prerequisite: At least a C- in C S 172 or C S 460 or consent of instructor.
Learning Outcomes
1. Use logic to specify precise meaning of statements, demonstrate the equivalence of statements, and test the validity of arguments
2. Construct and recognize valid proofs using different techniques including the principle of mathematical induction
3. Use summations, formulas for the sum of arithmetic and geometric sequences
4. Explain and apply the concepts of sets and functions
5. Apply counting principles to determine the number of various combinatorial configurations

C S 466. Compilers and Automata Transition
3 Credits (3)
Methods, principles, and tools for programming language processor design; basics of formal language theory (finite automata, regular expressions, context-free grammars); development of compiler components. For C S graduate students only; cannot be used in a students program of study. Taught with C S 370.
Prerequisite: At least a C in (C S 271 or C S 462), in (C S 272 or C S 463), in (C S 273 or C S 464), or consent of instructor.
Learning Outcomes
1. Understand the language theory concepts of regular languages, context free languages, regular expressions, context free grammars, and formal language hierarchy
2. Use Thompson's construction to convert from regular expression to NFA, and subset construction to convert from NFA to DFA
3. Apply recursive descent parsing in programming a parser of a small grammar
4. Understand the ideas in LL and LR parsing of context-free language classes
5. Understand and use table-driven top-down (LL(1)) and bottom up (SLR) parsing to parse a sentence

C S 468. Software Development Transition
3 Credits (3)
Software specification, design, testing, maintenance, documentation; informal proof methods; team implementation of a large project. For C S graduate students only. Cannot be used in a C S student's program of study. Consent of Instructor required. Taught with C S 371.
Prerequisite: At least a C- in C S 271 or C S 462, in C S 272 or C S 463, or consent of instructor.
Learning Outcomes
1. Understand and explain the activites and structure of different styles of software development processes, including waterfall, (spiral,) iterative, and agile methodologies
2. Apply requirements knowledge and techniques to create functional and non-functional requirements for a software system
3. Apply high and low level design ideas to create an object-oriented design of a software system
4. Use good design and programming ideas to implement individual and team software systems in compiled OOP languages
5. Apply white and black box testing techniques and tools to individual and team software development
6. Use UML class diagrams (and sequence diagrams) to capture aspects of system design and/or requirements (domain)

7. Use practical software development tools, including version control systems, automated build tools, and testing tools

C S 469. Data Structure and Algorithms Transition
3 Credits (3)
Introduction to efficient data structure and algorithm design. Order notation and asymptotic run-time of algorithms. Recurrence relations and solutions. Abstract data type dynamic set and data structures based on trees. Classic algorithm design paradigms: divide-and-conquer, dynamic programming, greedy algorithms. For CS graduate students only. Taught with CS 372. May be repeated up to 24 credits.
Prerequisite: At least a C- in (C S 272 or C S 463) and a C- in (C S 278 or C S 465), or consent of instructor.
Learning Outcomes
1. Analyze the growth of functions via asymptotic notation
2. Evaluate the asymptotic running time of a given algorithm
3. Solve recurrence relations of the kinds encountered in algorithm analysis
4. Design algorithms using the divide-and-conquer technique
5. Design algorithms using the greedy technique
6. Design algorithms using the dynamic-programming technique
7. Use and analyze data structures based on trees
8. Analyze the design, correctness, and time complexity of basic graph algorithms

C S 471. Programming Language Structure I
3 Credits (3)
Syntax, semantics, implementation, and application of programming languages; abstract data types; concurrency. Not for C S graduate students.
Prerequisite: At least a C- in C S 370 and C S 371.
Learning Outcomes
1. Improve the background for choosing appropriate programming languages for certain classes of programming problems
2. Increase the ability to learn new programming languages
3. Critically evaluate what paradigm and language are best suited for a new problem
4. Demonstrate the use of the primary segments for a running program
5. Apply the principles of functional programming
6. Apply the principles of logic programming
7. Program a simple parallel program with threads
8. Program in at least five different programming languages
9. Program in C to demonstrate architecture details

C S 473. Architectural Concepts I
3 Credits (3)
Comparison of architectures to illustrate concepts of computer organization; relationships between architectural and software features. Not for C S graduate students.
Prerequisite: At least a C- in C S 273 and C S 370.
Learning Outcomes
1. Explain the concepts in instruction set architecture
2. Analyze the behavior of pipelined CPU data path and control
3. Analyze behavior and performance of memory hierarchies with different cache designs
4. Describe the implementation of binary integer and floating point representation and arithmetic
5. Identify and analyze performance of instruction level parallelism and multi-core parallelism
6. Describe virtual memory and architectural support for operating systems
7. Understand the organization of various kinds of secondary storage devices, and their performance and tradeoffs
8. Create software that demonstrates performance of architectural features and evaluate the effects of software change

C S 474. Operating Systems I
3 Credits (3)
Operating system principles and structures, and interactions with architectures. Not for C S graduate students.
Prerequisite: At least a C- in C S 273, C S 371, and C S 372.
Learning Outcomes
1. Explain OS control and management of hardware resources
2. Explain OS management and execution of processes
3. Explain OS control and management of real and virtual memory
4. Explain classical concurrency issues and their solutions
5. Analyze and implement threads
6. Analyze OS interaction with networks and architecture

C S 475. Artificial Intelligence I
3 Credits (3)
Fundamental principles and techniques in artificial intelligence systems. Intelligent Agents; solving problems by searching; local search techniques; game-playing agents; constraint satisfaction problems; knowledge representation and reasoning. Further selected topics may also be covered. Not for C S graduate students. Taught with C S 505.
Prerequisite: At least a C- in C S 272 and C S 278.
Learning Outcomes
1. Use various search algorithms commonly used in problem-solving
2. Use methods for solving constraint satisfaction problems
3. Use propositional and first-order logic to represent knowledge
4. Use logical inference methods to derive conclusions from a knowledge base
5. Use adversarial search for game-playing agents
6. Analyze the different search strategies
7. Design and Implement heuristic search for problem-solving

C S 476. Computer Graphics I
3 Credits (3)
Languages, programming, devices, and data structures for representation and interactive display of complex objects. Not for C S graduate students. Taught with C S 506.
Prerequisite: At least C- in C S 370 or C S 371.
Learning Outcomes
1. Techniques used in three-dimensional graphics
2. Computer Graphics lightning and shading
3. Client-server graphics using WebGL
4. Geometric and Solid modeling
5. Computer Graphics implementation algorithms

C S 477. Digital Game Design
3 Credits (3)
An introduction to digital game design. Topics include design, development, and playtesting of games. The course is structured to use team-based learning. Not for C S graduate students. Taught with C S 517.
Prerequisite/Corequisite: C S 371.

Learning Outcomes

1. Describe, analyze, and/or critique games with a consistent vocabulary
2. Design, develop, and playtest games
3. Understand the formal systems of games
4. Communicate game designs through demonstrations and presentations

C S 478. Computer Security
3 Credits (3)
Introduction to the art and science of computer security. Fundamentals of computer security including elementary cryptography, authentication and access control, security threats, attacks, detection and prevention in application software, operating systems, networks and databases. Not for C S graduate students. Taught with C S 513.
Prerequisite: At least a C- in C S 272, C S 273 or consent of instructor.
Learning Outcomes

1. Describe fundamental concepts in security and privacy
2. Understand requirements of security in different contexts
3. Describe practical implementation challenges in security/privacy system design
4. Explain at a high-level symmetric and public key cryptography
5. Explain various access control mechanisms such as authnetication, authorization
6. Understand aspects of secure system design that a computer programmer/engineer needs to account for

C S 479. Special Topics
1-12 Credits
Topics announced in the Schedule of Classes. May be repeated under different subtitles. Not for C S graduate students. May be repeated up to 12 credits.

C S 480. Linux System Administration
3 Credits (3)
Basic system administration for Linux environments. Topics include user managements, file systems, security, backups, system monitoring, kernel configuration and other relevant aspects of system administration. Not for Computer Science graduate students.
Learning Outcomes

1. Be able to properly set up, configure, and maintain a Linux-based set of networked computers with shared resources
2. Understand the significance of proper administration of systems and its impact on users, their data and computational resources, and the security of the overall installation

C S 481. Visual Programming
3 Credits (3)
Design and implementation of programs using visual (i.e. dataflow or diagrammatic) programming techniques, with an emphasis on real-time data processing. Students will learn how to design visual programs, including how to handle cycles and state maintenance, and will learn to process audio, video, and other data using visual programs. Not for C S graduate students. Taught with C S 518.
Prerequisite: At least a C- in C S 272 and C S 278.
Learning Outcomes

1. Develop software in graph-based visual environments
2. Understand flows of control in visual programming environments
3. Use signals, digital and analog, to drive software
4. Communicate software design and evaluation with presentations, demos, and reports

C S 482. Database Management Systems I
3 Credits (3)
Database design and implementation; models of database management systems; privacy, security, protection, recovery. Not for C S graduate students. Taught with C S 502.
Prerequisite: At least a C- in C S 272 and C S 278.
Learning Outcomes

1. Utilize the basic concepts of relational database model
2. Utilize database query languages (e.g. SQL)
3. Identify data integrity and security requirements
4. Analyze, capture, and model user requirements for building database systems using conceptual models
5. Design and normalize relational schemas
6. Apply application development methods to implement a database system

C S 484. Computer Networks I
3 Credits (3)
Fundamental concepts of computer communication networks: layered network architecture, network components, protocol stack and service. Example of application, transport, network and data link layers, protocols primarily drawn from the Internet (TCP, UDP, and IP) protocol multimedia networks; network management and security. Not for C S graduate students. Taught with C S 504.
Prerequisite: At least a C- in C S 272 and CS 273.
Learning Outcomes

1. Explain the layered model of networking using the OSI and TCP/IP models
2. Describe the purpose and concepts of each layer in the OSI and TCP/IP models
3. Describe IP as a particular network layer protocol
4. Describe TCP and UDP as particular transport layer protocols
5. Describe Ethernet (11) and WiFi (15) as particular data link layer protocols
6. Describe and analyze routing and routing issues
7. Describe and analyze data link layer switching
8. Describe the need for application protocols such as HTTP
9. Explain other network issues such as multicasting and audio/video data streaming 1
10. Implement socket-based network programs

C S 485. Human-Centered Computing
3 Credits (3)
Covers iterative, human-centered interface design, including prototyping and evaluation. Basics of graphic design and visualization. Not for C S graduate students. Taught with C S 515.
Prerequisite: At least C- in C S 371.
Learning Outcomes

1. Describe, analyze, and/or critique a device interface using a design vocabulary
2. Enact a human-centered process of interaction design: gather data; develop a data-driven design; iterate design through testing; and evaluate results
3. Conduct human-computer interaction research by proposing, developing, and conducting experiments; analyzing data; and developing synthesized results

4. Communicate design and evaluation with presentations, demos, and reports
5. Implement a variety of interaction techniques

C S 486. Bioinformatics
3 Credits (3)
Introduction to bioinformatics and computational biology. Computational approaches to sequences analysis, protein structure prediction and analysis, and selected topics from current advances in bioinformatics. Not for C S graduate students. Taught with C S 516.
Prerequisite: At least a C- in C S 272 and C S 278.
Learning Outcomes
1. Explain the biology motivation of a bioinformatics question
2. Formulate a computational problem and its solution to address a molecular biology question
3. Implement basic bioinformatics algorithms such as sequence alignment, pattern matching, and genome assembly
4. Evaluate the performance of a bioinformatics algorithm on real data sets
5. Argue the correctness of a bioinformatics algorithm
6. Analyze the complexity of a bioinformatics algorithm

C S 487. Applied Machine Learning I
3 Credits (3)
An introductory course on practical machine learning. An overview of concepts for both unsupervised and supervised learning. Topics include classification, regression, clustering, and dimension reduction. Classical methods and algorithms such as linear regression, neural networks, support vector machines, and ensemble approaches. Recent techniques such as deep learning. Focused on applying of machine learning techniques in application domains. Not for Graduate Majors. Taught with: C S 519.
Prerequisite: At least a C- in C S 272, MATH 1511G; or consent of instructor.
Learning Outcomes
1. Implement and utilize different data processing techniques
2. Differentiate and assess several dimension reduction techniques
3. Utilize several classifiers (SVM, Decision tree, k-Nearest Neighbor, and logistic regression) and differentiate their advantages and disadvantages
4. Explain and demonstrate regression analysis
5. Describe and illustrate clustering approaches
6. Apply ensemble learning approaches
7. Implement several neural network classifiers, including deep learning models

C S 488. Introduction to Data Mining
3 Credits (3)
Techniques for exploring large data sets and discovering patterns in them. Data mining concepts, metrics to measure its effectiveness. Methods in classification, clustering, frequent pattern analysis. Selected topics from current advances in data mining. Taught with C S 508.
Prerequisite: At least a C- in C S 272 and C S 278.
Learning Outcomes
1. Explain and recognize different data mining tasks such as data pre-processing, visualization, classification, regression, clustering, association rules, and anomaly detection
2. Apply classical data mining / machine learning algorithms for classification, clustering, association rules, and anomaly detection
3. Evaluate and compare the performance of different data mining / machine learning algorithms
4. Utilize data mining algorithms to analyze data in real applications using a data mining tool

C S 489. Bioinformatics Programming
3 Credits (3)
Computer programming to analyze high-throughput molecular biology data including genomic sequences, bulk and single-cell transcriptome, epigenome, and other omics data. Quality control, library size normalization, confounding effect removal, clustering, statistical modeling, trajectory inference, and visualization. Taught with C S 509. May be repeated up to 3 credits.
Learning Outcomes
1. Write R scripts and functions to manipulate biological sequences, genome annotation, and gene expression data
2. Perform high-throughput data analysis with established R packages
3. Detect differential gene expression on RNA sequencing data
4. Perform single-cell RNA sequencing data analysis (quality control, library size normalization, confounding effect removal, modeling)
5. Assess statistical significance of analytical results
6. Create automatic data analysis pipeline to link multiple software packages

C S 491. Parallel Programming
3 Credits (3)
Programming of shared memory and distributed memory machines; tools and languages for parallel programming; techniques for parallel programming; parallel programming environments. Not for C S graduate students. Taught with C S 521.
Prerequisite: At least a C- in C S 370 or consent of instructor.
Learning Outcomes
1. Describe existing parallel architectures including shared memory versus distributed memory platforms
2. Apply basic techniques for organizing parallel computations
3. Apply basic techniques for performance measurement and theoretical limitations of parallelism
4. Explain alternative parallel techniques and hardware
5. Perform performance Analysis of different parallel programming technices
6. Program shared memory machines using threads, processes, and the OpenMP library
7. Program using a message passing paradigm and obtain working knowledge of the Message Passing Interface (MPI)

C S 493. Algorithm Design and Implementation
3 Credits (3)
This course introduces the basic knowledge of designing classical algorithms and implementing these algorithms using a programming language. In particular, the course teaches various data structures, including graphs and balanced binary search trees, and efficient schemes to implement these data structures. This course also teaches basic algorithm design techniques including divide-and-conquer, greedy scheme, and dynamic programming. This course covers graph algorithms, including graph traversals (depth-first search and breadth-first search), connectivity, shortest paths, and minimum spanning trees. Graduate standing. Not for CS students. Taught with C S 343.
Prerequisite: At least a C- in C S 272, or Consent of Instructor.

Learning Outcomes

1. Be able to use and implement sorting algorithms
2. Be able to design and implement graph algorithms
3. Be able to design and implement algorithms using the divide-and-conquer technique
4. Be able to design and implement algorithms using the greedy technique
5. Be able to design and implement algorithms using the dynamic programming technique
6. Be able to use and implement balanced search trees
7. Be able to use and implement hashing techniques
8. Be able to perform the run time analysis of basic algorithms using Big O notation

C S 494. Introduction to Smart Grids
3 Credits (3)
This course is an introduction to the technologies and design strategies associated with the Smart Grid. The emphasis will be on the development of communications, energy delivery, coordination mechanisms, and management tools to monitor transmission and distribution networks. Topics include: Smart grid introduction and evolution; Power systems; Networking and transport control; Artificial intelligence & agent coordination; Data mining for smart grids. Taught with C S 514. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 272 and a C- in ENGR 230; or Consent of instructor.
Learning Outcomes

1. Get basic understanding of how conventional power system is operated and protected
2. Understand and use basic knowledge of communication techniques in smart grids
3. Understand and use basic knowledge for the coordination of the different units in smart grids
4. Understand and apply data mining techniques for protecting smart grids

C S 496. Cloud and Edge Computing
3 Credits (3)
The course presents a top-down view of cloud computing, from applications and administration to programming and infrastructure. Its main focus is on the concepts of networking and parallel programming for cloud computing and large scale distributed systems which form the cloud infrastructure. The topics include: overview of cloud computing, cloud systems, parallel processing in the cloud, distributed storage systems, virtualization, security in the cloud, and multicore operating systems. Students will study state-of-the-art approaches to cloud computing followed by large cloud corporations, namely Google, Amazon, Microsoft, and Yahoo. Students will also apply what they learn through project developments using Amazon Web Services. Not for graduate CS majors. Taught with: C S 522.
Prerequisite: At least a C- in C S 372; background in C S 484/C S 504 is preferred or consent of instructor.
Learning Outcomes

1. The emphasis of the course will be on the understanding the concepts and the engineering trade-offs involved in the design of cloud computing systems
2. Cloud deployment models, cloud service models (software-as-a-service, infrastructure- as-a-service, protocol-as-a-service), cloud architecture, cloud-edge security, service level agreements, and load balancing in cloud and datacenters
3. Learn about cloud computing, especially what are their fundamental components, how these components interact, and how the technology is evolving for the future (edge computing, cloudlets, mobile edge computing, etc.).

C S 502. Database Management Systems I
3 Credits (3)
Database design and implementation; models of database management systems; privacy, security, protection, recovery; taught with C S 482; requires more advanced graduate work than C S 482. Students are expected to have solid knowledge of data structures and discrete mathematics.
Learning Outcomes

1. Utilize the basic concepts of relational database model
2. Utilize database query languages (e.g. SQL)
3. Identify data integrity and security requirements
4. Analyze, capture, and model user requirements for building database systems using conceptual models
5. Design and normalize relational schemas
6. Apply application development methods to implement a database system

C S 504. Computer Networks I
3 Credits (3)
Fundamental concepts of computer communication networks: layered network architecture, network components, protocol stack and service. Example of application, transport, network and data link layers, protocols primarily drawn from the Internet (TCP, UDP, and IP) protocol suite; local and wide area networks, wireless and mobile networks, multimedia networks; network management and security; taught with C S 484; requires more advanced graduate work than C S 484. Students are expected to have solid knowledge of data structures, machine-level programming. Knowledge of statistics (at the level of MATH 371 or MATH 470) is recommended.
Learning Outcomes

1. Explain the layered model of networking using the OSI and TCP/IP models
2. Describe the purpose and concepts of each layer in the OSI and TCP/IP models
3. Describe IP as a particular network layer protocol
4. Describe TCP and UDP as particular transport layer protocols
5. Describe Ethernet (802-11) and WiFi (802-15) as particular data link layer protocols
6. Describe and analyze routing and routing issues
7. Describe and analyze data link layer switching
8. Describe the need for application protocols such as HTTP
9. Explain other network issues such as multicasting and audio/video data streaming 1
10. Implement socket-based network programs

C S 505. Artificial Intelligence I
3 Credits (3)
Fundamental principles and techniques in artificial intelligence systems. Knowledge representation formalisms; heuristic problem solving techniques; automated logical deduction; robot planning methods; algorithmic techniques for natural language understanding, vision and learning; taught with C S 475; requires more advanced graduate work than C S 475. Students are expected to have strong knowledge of algorithms and data structures (at the level of C S 372).

Learning Outcomes

1. Use various search algorithms commonly used in problem-solving
2. Use methods for solving constraint satisfaction problems
3. Use propositional and first-order logic to represent knowledge
4. Use logical inference methods to derive conclusions from a knowledge base
5. Use adversarial search for game-playing agents
6. Analyze the different search strategies
7. Design and Implement heuristic search for problem-solving

C S 506. Computer Graphics I
3 Credits (3)
Languages, programming, devices, and data structures for representation and interactive display of complex objects. Taught with C S 476. Requires more advanced graduate work than C S 476. Students are expected to have knowledge of compilers design and software engineering equivalent to C S 370 and C S 371.

Learning Outcomes

1. Techniques used in three-dimensional graphics
2. Computer Graphics lightning and shading
3. Client-server graphics using WebGL
4. Geometric and Solid modeling
5. Computer Graphics implementation algorithms

C S 508. Introduction to Data Mining
3 Credits (3)
Techniques for exploring large data sets and discovering patterns in them. Data mining concepts, metrics to measure its effectiveness. Methods in classification, clustering, frequent pattern analysis. Selected topics from current advances in data mining. Students are expected to have a preparation in Discrete Mathematics and Data Structures equivalent to C S 272 and C S 278. Requires more advanced graduate work than C S 488. Taught with: C S 488.

Learning Outcomes

1. Explain and recognize different data mining tasks such as data pre-processing, visualization, classification, regression, clustering, association rules, and anomaly detection
2. Apply classical data mining / machine learning algorithms for classification, clustering, association rules, and anomaly detection
3. Evaluate and compare the performance of different data mining / machine learning algorithms
4. Utilize data mining algorithms to analyze data in real applications using a data mining tool

C S 509. Bioinformatics Programming
3 Credits (3)
Computer programming to analyze high-throughput molecular biology data including genomic sequences, bulk and single-cell transcriptome, epigenome, and other omics data. Quality control, library size normalization, confounding effect removal, clustering, statistical modeling, trajectory inference, and visualization. Taught with C S 489. Requires more advanced graduate work than C S 489.

Learning Outcomes

1. Write R scripts and functions to manipulate biological sequences, genome annotation, and gene expression data
2. Perform high-throughput data analysis with established R packages
3. Detect differential gene expression on RNA sequencing data
4. Perform single-cell RNA sequencing data analysis (quality control, library size normalization, confounding effect removal, modeling)
5. Assess statistical significance of analytical results
6. Create automatic data analysis pipeline to link multiple software packages

C S 510. Automata, Languages, Computability
3 Credits (3)
Regular and context-free languages, pushdown and finite-state automata, Turing machines, models of computation, halting problems. Students are expected to have knowledge of algorithms equivalent to C S 372. May be repeated up to 3 credits.

Learning Outcomes

1. Describe the language accepted by an automaton or generated by a regular expression or a context-free grammar
2. Design automata, regular expressions and context-free grammars accepting or generating a certain language
3. Prove properties of languages, grammars, and automata with formal mathematical methods
4. Convert between equivalent deterministic and non-deterministic finite automata, and regular expressions
5. Convert between equivalent context-free grammars and pushdown automata
6. Define Turing machines performing simple tasks

C S 513. Computer Security
3 Credits (3)
Introduction to the art and science of computer security.Fundamentals of computer security including elementary cryptography, authentication and access control, security threats, attacks, detection and prevention in application software, operating systems, networks and databases. Taught with C S 478. Requires more advanced graduate work than C S 478. Recommended knowledge of materials in C S 272. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 273 or consent of instructor.

Learning Outcomes

1. Describe fundamental concepts in security and privacy
2. Understand requirements of security in different contexts
3. Describe practical implementation challenges in security/privacy system design
4. Explain at a high-level symmetric and public key cryptography
5. Explain various access control mechanisms such as authentication, authorization
6. Understand aspects of secure system design that a computer programmer/engineer needs to account for

C S 514. Introduction to Smart Grids
3 Credits (3)
This course is an introduction to the technologies and design strategies associated with the Smart Grid. The emphasis will be on the development of communications, energy delivery, coordination mechanisms, and management tools to monitor transmission and distribution networks. Topics include: Smart grid introduction and evolution; Power systems; Networking and transport control; Artificial intelligence & agent coordination; Data mining for smart grids. Taught with C S 494. Requires more advanced work than C S 494. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 272 and a C- in ENGR 230; or Consent of instructor.

Learning Outcomes

1. Get basic understanding of how conventional power system is operated and protected

2. Understand and use basic knowledge of communication techniques in smart grids
3. Understand and use basic knowledge for the coordination of the different units in smart grids
4. Understand and apply data mining techniques for protecting smart grids

C S 515. Human-Centered Computing
3 Credits (3)
Covers iterative, human-centered interface design, including prototyping and evaluation. Basics of graphic design and visualization. Taught with C S 485. Requires more advanced graduate work than C S 485 with an emphasis on studying recent research in human-computer interaction. Students are expected to have knowledge of software engineering equivalent to C S 371.
Learning Outcomes
1. Describe, analyze, and/or critique a device interface using a design vocabulary
2. Enact a human-centered process of interaction design: gather data; develop a data-driven design; iterate design through testing; and evaluate results
3. Conduct human-computer interaction research by proposing, developing, and conducting experiments; analyzing data; and developing synthesized results
4. Communicate design and evaluation with presentations, demos, and reports
5. Implement a variety of interaction techniques

C S 516. Bioinformatics
3 Credits (3)
Introduction to bioinformatics and computational biology. Computational approaches to sequences analysis, protein structure prediction and analysis, and selected topics from current advances in bioinformatics; taught with C S 486; requires more advanced graduate work than C S 486. Students are expected to have a knowledge of algorithms and data structures equivalent to C S 372 or exposure to Biology (equivalent to BIOL 2310 or BIOL 311).
Learning Outcomes
1. Explain the biology motivation of a bioinformatics question
2. Formulate a computational problem and its solution to address a molecular biology question
3. Implement basic bioinformatics algorithms such as sequence alignment, pattern matching, and genome assembly
4. Evaluate the performance of a bioinformatics algorithm on real data sets
5. Argue the correctness of a bioinformatics algorithm
6. Analyze the complexity of a bioinformatics algorithm

C S 517. Digital Game Design
3 Credits (3)
An introduction to digital game design. Topics include design, development, and playtesting of games. The course is structured to use team-based learning. Taught with C S 477. Requires more advanced graduate work than C S 477 with deeper attention to a team game project.
Learning Outcomes
1. Describe, analyze, and/or critique games with a consistent vocabulary
2. Design, develop, and playtest games
3. Understand the formal systems of games
4. Communicate game designs through demonstrations and presentations

C S 518. Visual Programming
3 Credits (3)
Design and implementation of programs using visual (i.e. dataflow or diagrammatic) programming techniques, with an emphasis on real-time data processing. Students will learn how to design visual programs, including how to handle cycles and state maintenance, and will learn to process audio, video, and other data using visual programs. Students must be in graduate standing to enroll. Taught with C S 481. Requires more advanced graduate work than C S 481.
Learning Outcomes
1. Develop software in graph-based visual environments
2. Understand flows of control in visual programming environments
3. Use signals, digital and analog, to drive software
4. Communicate software design and evaluation with presentations, demos, and reports

C S 519. Applied Machine Learning I
3 Credits (3)
An introductory course on practical machine learning. An overview of concepts for both unsupervised and supervised learning. Topics include classification, regression, clustering, and dimension reduction. Classical methods and algorithms such as linear regression, neural networks, support vector machines, and ensemble approaches. Recent techniques such as deep learning. Focused on applying of machine learning techniques in application domains. Taught with: C S 487. Requires more advanced graduate work than C S 487.
Learning Outcomes
1. Implement and utilize different data processing techniques
2. Differentiate and assess several dimension reduction techniques
3. Utilize several classifiers (SVM, Decision tree, k-Nearest Neighbor, and logistic regression) and differentiate their advantages and disadvantages
4. Explain and demonstrate regression analysis
5. Describe and illustrate clustering approaches
6. Apply ensemble learning approaches
7. Implement several neural network classifiers, including deep learning models

C S 521. Parallel Programming
3 Credits (3)
Programming of shared memory and distributed memory machines; tools and languages for parallel programming; parallelizing compilers; parallel programming environments; taught with C S 491; requires more advanced graduate work than C S 491. Students are expected to have knowledge of programming and machine organization equivalent to C S 271 and C S 273.
Learning Outcomes
1. Describe existing parallel architectures including shared memory versus distributed memory platforms
2. Apply basic techniques for organizing parallel computations
3. Apply basic techniques for performance measurement and theoretical limitations of parallelism
4. Explain alternative parallel techniques and hardware
5. Perform performance Analysis of different parallel programming technices
6. Program shared memory machines using threads, processes, and the OpenMP library

7. Program using a message passing paradigm and obtain working knowledge of the Message Passing Interface (MPI)

C S 522. Cloud and Edge Computing
3 Credits (3)
The course presents a top-down view of cloud computing, from applications and administration to programming and infrastructure. Its main focus is on the concepts of networking and parallel programming for cloud computing and large scale distributed systems which form the cloud infrastructure. The topics include: overview of cloud computing, cloud systems, parallel processing in the cloud, distributed storage systems, virtualization, security in the cloud, and multicore operating systems. Students will study state-of-the-art approaches to cloud computing followed by large cloud corporations, namely Google, Amazon, Microsoft, and Yahoo. Students will also apply what they learn through project developments using Amazon Web Services. Might have additional requirements for graduate students. To enroll in this course a background in C S 484/C S 504 is preferred or have consent from the instructor. Taught with: C S 496. Requires more advanced graduate work than C S 496.
Learning Outcomes
1. The emphasis of the course will be on the understanding the concepts and the engineering trade-offs involved in the design of cloud computing systems
2. Cloud deployment models, cloud service models (software-as-a-service, infrastructure- as-a-service, protocol-as-a-service), cloud architecture, cloud-edge security, service level agreements, and load balancing in cloud and datacenters
3. Learn about cloud computing, especially what are their fundamental components, how these components interact, and how the technology is evolving for the future (edge computing, cloudlets, mobile edge computing, etc.).

C S 525. Introduction to Cryptography
3 Credits (3)
The course covers basic cryptographic primitives, such as symmetric, public-key ciphers, digital signature schemes, and hash functions, and their mathematical underpinnings. Course helps students understand basic notions of security in a cryptographic sense: chosen plaintext and chosen ciphertext attacks, games, and reductions. Course also covers computational number theory relevant to cryptography. Consent of Instructor required. Taught with: C S 380. Requires more advanced graduate work than C S 380.
Prerequisite: C S 278 (or equivalent) with a C or better.
Learning Outcomes
1. Describe basic cryptographic primitives, including symmetric ciphers, asymmetric ciphers, digital signatures, message authentication codes, and hash functions.
2. Understand the mathematical, fundamental underpinnings of cryptography, and how to reason about the security of crypto primitives: indistinguishability (IND) properties of ciphertexts, CPA/CCA games, and reductions to fundamental math assumptions;
3. Be able to discuss number theory/algebra underpinning the design of cryptographic primitives, in some depth.

C S 532. Modern Web Technologies
3 Credits (3)
In this course, we will take a full-stack approach to modern web application design. We will start with the fundamentals including HTML5, CSS3, Javascript, JSON, and the underlying networking concepts and protocols driving the modern web. We will then move on to more advanced topics including javascript backend development with Node.js, NoSQL database design with MongoDB, cloud computing, and re-sponsive web design. Finally, we cover advanced topics including the design and im- plementation of browser extensions and real-time web technologies like WebRTC and WebSockets. Consent of Instructor required. Taught with: C S 382. Requires more advanced graduate work than C S 382.
Learning Outcomes
1. Understand the fundamental technologies and operation of the web.
2. Design and develop responsive interactive web sites.
3. Deploy web applications on Cloud Computing Platforms.
4. Leverage modern tools and packages to develop full stack web applications.
5. Be fluent in the application of emerging web technologies like browser extensions, WebSockets, and WebRTC.
6. Use existing materials and references on the web to learn new skills.

C S 533. Introduction to Deep Learning
3 Credits (3)
The course covers basic concepts of neural networks which include transition of classical machine learning to hierarchical feature learning, feedforward networks, regularization, optimization, hyperparameter tuning, deep convolutional networks and their applications in computer vision, deep sequence models, and deep generative models. Taught with C S 383. Requires more advanced graduate work than C S 383. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 272 or C S 153, and C S 278, or consent of instructor.
Learning Outcomes
1. Have significant familiarity with different state-of-the-art theories and practices of deep learning.
2. Be able to apply deep learning to a variety of tasks suitable for data science-based projects of academia and industry.
3. Understand much of the current literature on the topic, review papers, and extend their knowledge through further study.
4. Design and evaluate novel deep learning models.
5. Train and test deep learning models on real-life and benchmark datasets using Python libraries such as TensorFlow and PyTorch.

C S 534. Graph Data Mining
3 Credits (3)
The course covers graph terminology, representation, and techniques to extract patterns from large graph data. The topics include random and scale-free graph generation, link analysis (PageRank), graph representation learning, graph neural networks, deep graph generation, community detection, frequent subgraph mining, graph classification, influence maximization, and knowledge graph mining. Taught with C S 384. Requires more advanced graduate work than C S 384. May be repeated up to 3 credits.
Prerequisite: At least a C- in C S 272 or C S 153, and C S 278, or consent of instructor.
Learning Outcomes
1. Have significant familiarity with different state-of-the-art theories and practices of graph data mining.
2. Graph representation and graph querying using graph manipulating toolbox/library.
3. Use random and scale-free graph models to generate graphs and visualize complex network properties.
4. Apply algorithms such as PageRank, spectral clustering, and non-negative matrix factorization.

5. Implement graph representation learning algorithms and graph neural network.
6. Understand much of the current literature on the topic, review papers, extend their knowledge through further study, and present findings of the papers.

C S 570. Analysis of Algorithms
3 Credits (3)
Techniques for design and analysis of algorithms; time and space complexity; proving correctness of programs. Particular algorithms such as sorting, searching, dynamic programming. NP complete problems. Students are expected to have knowledge of algorithms and data structures equivalent to C S 372.
Learning Outcomes
1. Prove algorithm correctness by loop-invariant
2. Prove an algorithm to be incorrect by counterexamples
3. Develop efficient divide-and-conquer algorithms
4. Design and analyze binary search tree algorithms
5. Construct dynamic programming solutions
6. Prove the correctness of dynamic programming solutions by contraposition
7. Traverse graphs efficiently
8. Find paths in graphs efficiently
9. Determine if a problem is NP-Complete or NP-Hard 1
10. Basic concepts of quantum computing

C S 573. Architectural Concepts II
3 Credits (3)
Advanced topics related to computer architecture, guided by the current literature. Students are expected to have knowledge of computer architectures equivalent to C S 473 and of operating systems equivalent to C S 474. Crosslisted with: E E 564.
Learning Outcomes
1. Be able to explain the features in a modern multicore CPU architecture
2. Be able to utilize hardware counter features of a CPU in performance evaluation
3. Be able to explain the architecture of GPUs and their capabilities and drawbacks
4. Be able to evaluate novel cutting-edge architectural features and designs
5. Be able to present a research paper to an advanced audience

C S 574. Operating Systems II
3 Credits (3)
Advanced topics related to operating system principles, guided by the current literature. Students are expected to have knowledge of computer architectures and operating systems equivalent to C S 473 and C S 474.
Learning Outcomes
1. Further an understanding of the principles of operating systems.
2. Develop insight into process management and scheduling issues.
3. Understand memory management operation.
4. Develop an understanding of file system implementation and of multiple levels of hardware support and management.
5. Develop a deep understanding of the concepts of cooperating processes, including communication, synchronization, and deadlock (detection and avoidance).
6. Be able to evaluate operating system features.
7. Develop an understanding of the distributed operating system environment.

C S 575. Artificial Intelligence II
3 Credits (3)
Covers advanced theory and application of artificial intelligence. Concentration on several specific research areas, such as knowledge representation, problem solving, common-sense reasoning, natural language understanding, automated tutoring systems, learning systems. Students are expected to have knowledge of artificial intelligence equivalent to C S 475.
Learning Outcomes
1. Apply selected planning algorithms in solving problems
2. Identify problems where knowledge representation and reasoning techniques are applicable
3. Be able to apply answer set programming in problem solving
4. Be aware of various advanced research topics in Artificial Intelligence

C S 579. Special Topics
1-6 Credits
Topic announced in the Schedule of Classes.

C S 581. Advanced Software Engineering
3 Credits (3)
Advanced tools and methods for developing large software systems. Topics include object-oriented modeling and design, component architectures, templates and generic programming, software configuration and revision control, static and dynamic analysis tools, model, checking, advanced testing, and verification. Students are expected to have knowledge of software engineering equivalent to C S 371.
Learning Outcomes
1. Be able to explain modern software development process ideas
2. Be able to apply agile software development techniques in a project
3. Be able to specify, design, and develop a complex software system in a team
4. Be able to properly utilize both black box and white box testing techniques
5. Be able to explain how unsound and incomplete formal methods can aid in system verification and validation
6. Be able to utilize sound and complete formal methods to prove properties of a system

C S 582. Database Management Systems II
3 Credits (3)
Advanced data models and abstractions, dependencies, implementations, languages, database machines, and other advanced topics. Students are expected to have knowledge of data base management systems equivalent to C S 482.
Learning Outcomes
1. Analyze storage and file structures of an RDBMS
2. Analyze and apply indexing techniques of an RDBMS
3. Analyze query evaluation approaches of an RDBMS
4. Analyze the mechanisms of transaction management in an RDBMS

C S 584. Computer Networks II
3 Credits (3)
Advanced topics in computer networks. Covers advanced topics in networking, with emphasis on wireless, and IP networks. Students are expected to have knowledge of computer networks equivalent to C S 484, and of statistics equivalent to MATH 371 or MATH 470.

Learning Outcomes

1. Understand design of link layer protocols.
2. Understand challenges and implementations for multimedia streaming.
3. Be able to use basic security constructs in the networking context.
4. Understand the concepts of edge and cloud computing
5. Understand the concepts and challenges of Internet of Things
6. Learn concepts of distributed networking
7. Learn and evaluate future internet architectures

C S 586. Algorithms in Systems Biology
3 Credits (3)
The course will introduce important algorithms and computational models used in systems biology to study molecular mechanisms for cellular dynamics, processes, and systems. Cellular processes, such as metabolism and signal transduction, are studied as systems and networks quantitatively from high throughput molecular measurements. The topics include molecular biological systems, network alignment, model simulation, network inference, model optimization, and hybrid models. Students will be able to construct models and analyze their properties in the context of molecular biological systems. Students are expected to have knowledge of algorithms and data structures equivalent to C S 372.

Learning Outcomes

1. Create mathematical representation of biological systems
2. Infer biological network topology from observed omics data set
3. Simulate the behavior of a biological system using a mathematical model
4. Characterize behaviors of biological systems
5. Estimate parameters in a biological system model
6. Validate a model’s statistical relevance given observed data

C S 589. Special Research Problems
1-6 Credits (1-6)
Faculty-supervised investigation, to culminate in a written report. Maximally 6 credits can be applied to the student program of study. Written agreement with faculty supervisor is the required consent. May be repeated up to 18 credits.

Learning Outcomes

1. Research experience for graduate student.

C S 598. Master’s Project
1-6 Credits
Project-oriented capstone course to be completed by M.S. students under supervision of their advisor. Maximum of 6 credits may be applied toward M.S. degree. Restricted to C S majors.
Prerequisite: written agreement with instructor.

C S 599. Master’s Thesis
1-6 Credits (1-6)
Thesis to be developed by M.S. Students under supervision of their advisor. May be repeated for a maximum of 6 credits. Restricted to majors.
Prerequisite: consent of instructor.

C S 600. Pre-dissertation Research
1-15 Credits
Pre-dissertation research.

C S 700. Doctoral Dissertation
1-15 Credits
Dissertation.

Office Location: Science Hall 123

Phone: (575) 646-3723

Postal Address
Department of Computer Science
New Mexico State University
P.O. Box 30001, MSC CS
Las Cruces, NM 88003
Shipping Address
Department of Computer Science
New Mexico State University
1290 Frenger Mall, SH 123
Las Cruces, NM 88003

Website: https://computerscience.nmsu.edu/