Universidad de SanAndrés

Matemática Discreta

2do Semestre 2022

PROFESORES A CARGO
Juan I. Giribet
Juan M. Medina

AUXILIARES DOCENTES
Francisco J. Presenza
Reurison Silva Rodrigues

Sobre el curso

La Matemática Discreta proporciona una base fundamental para la informática, y sus aplicaciones son diversas. En el nivel más fundamental, todos los datos de una computadora se representan como bits (ceros y unos). Las computadoras hacen cálculos modificando estos bits de acuerdo con las leyes del álgebra booleana, que forman la base de todos los circuitos digitales actuales. Los lenguajes de programación de bajo nivel se basan directamente en operadores lógicos como and, not y or. Más allá de ser el lenguaje en el cual en última instancia uno se comunica con la computadora, en programación la lógica booleana es utilizada para controlar el flujo de un programa, es decir, qué instrucciones se ejecutan bajo ciertas condiciones.

Al programar, es importante tener garantías de que el código logrará los resultados deseados. Las matemáticas son de gran utilidad para describir con precisión los programas, y las herramientas de la lógica proposicional se pueden usar para razonar sobre su corrección. Esta habilidad es fundamental para el diseño y análisis de algoritmos, un área central de la informática. En programación imperativa, conceptos como el de iteración, o incluso un paradigma como la programación funcional, se basan en el principio de inducción matemática para verificar sus bucles (for y while), así como las llamadas a funciones recursivas, respectivamente. La lógica es el lenguaje utilizado para la mayoría de los lenguajes de especificación formal, y es fundamental para comprender gran parte de la literatura sobre verificación, fundamentos y diseño de lenguajes de programación. Por ejemplo, los lenguajes de la familia SQL son solo implementaciones de lógica relacional con características adicionales, y muchos otros lenguajes específicos de dominio son implementaciones similares de algún cálculo lógico particular. La verificación de programas y los métodos formales están experimentando una adopción cada vez mayor en la industria y se utilizan junto con las técnicas de prueba tradicionales para aumentar la confianza de que el software se comporta como se supone que debe hacerlo.

La inducción y la recursividad son conceptos clave para comprender el paradigma funcional de la programación. La recurrencia también es una forma común de definir

algoritmos y estructuras de datos, incluso si la implementación concreta se define de forma iterativa. Además, forman la columna vertebral de muchos modelos de computación y de áreas más teóricas de la informática. También son fundamentales para la verificación de software, otra área de la informática que está aumentando en adopción, ya que las propiedades de corrección y seguridad del software se vuelven cada vez más críticas en aplicaciones sensibles.

La teoría de números tiene aplicaciones críticas en blockchain, criptografía y seguridad informática. Los sistemas criptográficos modernos deben ser matemáticamente correctos para proteger los datos de los usuarios de adversarios malintencionados. La aritmética modular es la base matemática de las funciones hash, que son herramientas extremadamente útiles con muchas aplicaciones. Las sumas de verificación, basadas en hashing, pueden verificar que los archivos transferidos a través de Internet no contengan errores. Las estructuras de datos, como los mapas hash, se basan en la aritmética modular para operaciones eficientes. La teoría de números también tiene usos relacionados con la memoria en la arquitectura de computadoras y los sistemas operativos.

Las técnicas de conteo se utilizan para desarrollar la intuición cuantitativa. Por ejemplo, se pueden usar para determinar la cantidad de contraseñas válidas que se pueden formar a partir de un conjunto determinado de reglas y cuánto tiempo le tomaría a un atacante forzarlas todas. El principio del casillero explica por qué no existe un algoritmo universal de compresión sin pérdida: cada algoritmo de compresión debe hacer que ciertos archivos sean más pequeños y otros más grandes. Por lo tanto, cada algoritmo de compresión está diseñado para comprimir un tipo diferente de archivo (texto, imágenes, video, etc.). Contar es útil para analizar la complejidad de los algoritmos. En las aplicaciones del mundo real, existen compensaciones complicadas entre varios recursos diferentes que están disponibles. Ciertas tareas necesitan algoritmos rápidos y pueden permitirse usar mucho espacio para lograr velocidad, mientras que otras no tienen mucho espacio y, por lo tanto, necesitan sacrificar tiempo por espacio. En situaciones más complejas, se debe lograr un punto óptimo en el uso de recursos para que el sistema no se quede sin un recurso y pueda seguir funcionando. El conteo es la base para hacer tales consideraciones de manera estructurada y, de hecho, puede usarse para dar garantías formales sobre el uso de los recursos.

Los grafos son estructuras de datos que se utilizan en diversas aplicaciones, en particular para modelar relaciones y responder preguntas sobre dichos datos, puede utilizarse en problemas de navegación para hallar la ruta más rápida desde un punto a otro, o para modelar las cercanías de personas en una red profesional, también en telecomunicaciones para modelar la red celular (de hecho, red es un nombre alternativo para un gráfico) o la topología de internet. En informática los grafos son una de las estructuras de datos más utilizadas y permiten construir sistemas de archivos, son utilizadas para control de versiones de código, entre otras tantas aplicaciones.

Los párrafos anteriores fundamentan el programa del curso Matemática Discreta y su importancia dentro de la carrera Ingeniería en Inteligencia Artificial.

Programa

1. Diagonalización de matrices. Formas de Jordan. Diagonalización de matrices hermíticas y positivas.
2. Conceptos básicos de teoría de números: Representación de números, números binarios, aritmética módulo n y aplicaciones en criptografía. Máximo común divisor, algoritmo de Euclides.
3. Lógica formal. Fórmulas lógicas, equivalencia de fórmulas, tablas de verdad, leyes de De Morgan. Predicados y cuantificadores. Métodos de demostración.
4. Conteo: Principio de biyección y permutaciones. Cálculo combinatorio y variaciones. Triángulo de Pascal. Relaciones de equivalencia y relación de orden, conjuntos bien ordenados y conceptos básicos de cardinalidad, conjuntos contables y no contables. Clases de equivalencia y ejemplos.
5. Inducción y recursión: Principio de inducción, recursividad, recurrencia de primer orden, Recurrencia de Fibonacci, algoritmos de división y conquista.
6. Grafos: Definiciones básicas, conectividad, subgrafos, caminos y ciclos. Isomorfismo de grafos, Matriz de adyacencia, Laplaciano de un grafo. Grafos conectados, búsqueda de componentes conexas en un grafo. Árboles, árboles binarios, búsqueda binaria, árboles de decisión. Grafos dirigidos, búsqueda de ciclos en grafos dirigidos, conectividad.
7. Complejidad: Complejidad en tiempo y nociones de limitaciones en espacio. Orden de complejidad, complejidad algorítmica en algoritmos recursivos, aplicación a algoritmos clásicos. Diferencia entre la complejidad algorítmica y la complejidad de un problema. Problemas P, NP, NP-completos.

1. Modalidad de trabajo:

El curso incluye dos clases magistrales y dos sesiones tutoriales por semana. Durante las clases magistrales haremos una introducción teórico-práctica a los temas del programa. Las sesiones tutoriales estarán centradas en aspectos prácticos y de aplicación de los contenidos. Durante el curso responderemos y resolveremos un buen número de preguntas y problemas, y frecuentemente haremos uso de casos reales que se relacionen con los temas bajo estudio.

Los alumnos trabajarán de forma individual o grupal, dependiendo de la actividad.

2. Mecanismos de Comunicación: con el fin de mantener un orden y de maximizar las posibilidades de atender a consultas adecuadamente, se requiere que la comunicación entre profesores y estudiantes sea en los siguientes medios, en orden descendente de prioridad.

2.1. Avisos en Campus Virtual: toda la comunicación general de los profesores a los estudiantes se hará a través de "Avisos" en Campus Virtual. Esto asegura que todos los estudiantes registrados reciban los mensajes, y además los avisos quedarán registrados para futuro acceso. Por favor revisar los "spam detectors", para que los emails no sean ocultados.

2.2. Consultas en Clase: <u>se espera que la mayoría de las consultas en persona sean evacuadas durante las clases magistrales o tutoriales.</u> Los profesores intentarán dejar una porción al final de cada clase para contestar preguntas relacionadas al curso.

3. Calendario.

El curso se guiará por calendario de UDESA para el primer cuatrimestre, el cual estará publicado en el campus virtual del curso.

Bibliografía

1. V. K. Balakrishnan. Introductory Discrete Mathematics, Dover Books on Computer Science.
2. Gary Haggard, John Schlipf, Sue Whitesides. Discrete Mathematics for Computer Science. Thomson Brooks/Cole, 2006.
3. Clifford Stein, Robert L. Drysdale, Kenneth Bogart. Discrete Mathematics for computer scientists. Addison-Wesley, 2011.