



Universidad Nacional del Altiplano (UNA)

Escuela Profesional de
Ciencia de la Computación
Sílabo 2024-II

1. CURSO

CS111. Introducción a la Ciencia de la Computación (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS111. Introducción a la Ciencia de la Computación
2.2 Semestre	:	1 ^{er} Semestre.
2.3 Créditos	:	3
2.4 horas	:	2 HT; 2 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	Ninguno Ninguno

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Este curso es la base para entender los conceptos fundamentales de pensamiento computacional transversales a cualquier profesión.

El curso presenta, desde un nivel cero, una visión panorámica de: introductoria al pensamiento computacional, almacenamiento de datos, arquitectura de computadores, sistemas operativos, redes e Internet, algoritmos, métodos de ordenamiento, ingeniería de software, bases de datos, estructuras de datos, ingeniería de software, computación gráfica, inteligencia artificial.

Debido a que está diseñado como un curso introductorio a la Ciencia de la Computación, los conceptos son presentados de forma lúdica y utilizando una metodología de Aprendizaje Activo (*Active Learning*). Durante el dictado del curso, se busca siempre una participación activa de la audiencia al estilo de una obra de teatro.

Las áreas de conocimiento relacionadas que se tocan están directamente relacionadas a la Ciencia de la Computación de acuerdo a la *Computing Curricula ACM/IEEE-CS*.

El curso **no requiere** ningún tipo de conocimiento previo en temas de manejo de computadores y puede ser tomado por alumnos de cualquier carrera.

5. OBJETIVOS

- Introducir los conceptos fundamentales de Pensamiento Computacional y Ciencia de la Computación a estudiantes de cualquier carrera profesional.
- Desarrollar su capacidad de abstracción.
- Entender como el Pensamiento Computacional se aplica en cada una de sus profesiones.
- Aplicar conceptos avanzados de Ciencia de la Computación de una manera simple en cualquier carrera.

6. RESULTADOS DEL ESTUDIANTE

) ()

) ()

6) Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. ()

7. TEMAS

Unidad 1: Pensamiento Computacional. Parte I (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Indicaciones generales del curso. • Explicación del sistema de evaluación. • Definición de Computación. • Computación como un binomio Humano-Computador. • Distorsiones en la definición de computación. • Computación como la automatización de una abstracción. • Computación e Ingeniería: similitudes y diferencias. • Resolución de problemas algorítmicos. • Dinámica: Entendiendo la ejecución de un algoritmo a velocidad humana. 	<ul style="list-style-type: none"> • Aplicar los conceptos fundamentales de computación en situaciones reales. [Usar] • Identificar las distorsiones de la Computación en situaciones reales. [Usar] • Identificar con claridad al menos 3 contextos de uso de la palabra <i>Engineer</i> en inglés. [Evaluar] • Identificar las limitaciones del ser humano para resolver problemas computacionales. [Usar]
Lecturas : [Brookshear2019]	

Unidad 2: Pensamiento Computacional. Parte II (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Numeración binaria vs decimal. • Representación de caracteres: la tabla ASCII. • Representación interna de colores. • Entendiendo una imagen desde adentro. • Búsqueda binaria. • Complejidad computacional de un algoritmo. 	<ul style="list-style-type: none"> • Aplicar diversos sistemas de numeración para problemas reales. [Usar] • Entender la representación interna de los caracteres en la tabla ASCII y en UTF-8. [Familiarizarse] • Entender la representación de los colores en una imagen. [Familiarizarse] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima. [Usar] • Determine informalmente el tiempo y el espacio de complejidad de simples algoritmos. [Usar]
Lecturas : [Brookshear2019]	

Unidad 3: Lógica digital y sistemas digitales (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Revisión e historia de la Arquitectura de Computadores. • Lógica combinacional vs. secuencial/Arreglos de puertas de campo programables como bloque fundamental de construcción lógico combinacional-secuencial. • Múltiples representaciones / Capas de interpretación (El hardware es solo otra capa) 	<ul style="list-style-type: none"> • Describir el avance paulatino de los componentes de la tecnología de computación, desde los tubos de vacío hasta VLSI, desde las arquitecturas mainframe a las arquitecturas en escala warehouse [Familiarizarse] • Comprender que la tendencia de las arquitecturas modernas de computadores es hacia núcleos múltiples y que el paralelismo es inherente en todos los sistemas de hardware [Familiarizarse] • Explicar las implicancias de los límites de potencia para mejoras adicionales en el rendimiento de los procesadores y también en el aprovechamiento del paralelismo [Familiarizarse] • Relacionar las varias representaciones equivalentes de la funcionalidad de un computador, incluyendo expresiones y puertas lógicas, y ser capaces de utilizar expresiones matemáticas para describir las funciones de circuitos combinacionales y secuenciales sencillos [Familiarizarse] • Diseñar los componentes básicos de construcción de un computador: unidad aritmético lógica (a nivel de puertas lógicas), unidad central de procesamiento (a nivel de registros de transferencia), memoria (a nivel de registros de transferencia) [Usar]
Lecturas : [Brookshear2019]	

Unidad 4: Representación de programas (2)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Programas que tienen otros programas como entrada tales como interpretes, compiladores, revisores de tipos y generadores de documentación. • Árboles de sintaxis abstracta, para contrastar la sintaxis correcta. • Estructuras de datos que representan código para ejecución, traducción o transmisión. 	<ul style="list-style-type: none"> • Explicar como programas que procesan otros programas tratan a los otros programas como su entrada de datos [Familiarizarse] • Describir un árbol de sintaxis abstracto para un lenguaje pequeño [Usar] • Describir los beneficios de tener representaciones de programas que no sean cadenas de código fuente [Familiarizarse] • Escribir un programa para procesar alguna representación de código para algún propósito, tales como un interprete, una expresión optimizada, o un generador de documentación [Usar]
Lecturas : [Brookshear2019]	

Unidad 5: Criptografía (2)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Terminología básica de criptografía cubriendo las nociones relacionadas con los diferentes socios (comunicación), canal seguro / inseguro, los atacantes y sus capacidades, cifrado, descifrado, llaves y sus características, firmas. • Apoyo a la infraestructura de clave pública para la firma digital y el cifrado y sus desafíos. 	<ul style="list-style-type: none"> • Describir el propósito de la Criptografía y listar formas en las cuales es usada en comunicación de datos [Familiarizarse] • Explicar como los protocolos de intercambio de claves trabajan y como es que pueden fallar [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 6: Organización y Arquitectura del Sistema de Memoria (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Sistemas de Almacenamiento y su Tecnología. • Jerarquía de Memoria: importancia de la localización temporal y espacial. • Organización y Operaciones de la Memoria Principal. • Latencia, ciclos de tiempo, ancho de banda e intercalación. • Memorias caché (Mapeo de direcciones, Tamaño de bloques, Reemplazo y Políticas de almacenamiento) • Multiprocesador coherencia cache / Usando el sistema de memoria para las operaciones de sincronización de memoria / atómica inter-core. • Memoria virtual (tabla de página, TLB) • Manejo de Errores y confiabilidad. • Error de codificación, compresión de datos y la integridad de datos. 	<ul style="list-style-type: none"> • Identifique las principales tecnologías de memoria (Por ejemplo: SRAM, DRAM, Flash, Disco Magnético) y su relación costo beneficio [Familiarizarse] • Explique el efecto del retardo de la memoria en tiempo de ejecución [Familiarizarse] • Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Familiarizarse] • Describa como el uso de jerarquía de memoria (caché, memoria virtual) es aplicado para reducir el retardo efectivo en la memoria [Familiarizarse] • Explique el efecto del retardo de la memoria en tiempo de ejecución [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 7: Visión general de Sistemas Operativos (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Papel y el propósito del sistema operativo. • Funcionalidad de un sistema operativo típico. • Los mecanismos de apoyo modelos cliente-servidor, dispositivos de mano. • Cuestiones de diseño (eficiencia, robustez, flexibilidad, portabilidad, seguridad, compatibilidad) • Influencias de seguridad, creación de redes, multimedia, sistemas de ventanas. 	<ul style="list-style-type: none"> • Explicar los objetivos y funciones de un sistema operativo moderno [Familiarizarse] • Analizar las ventajas y desventajas inherentes en el diseño de un sistema operativo [Usar] • Describir las funciones de un sistema operativo contemporáneo respecto a conveniencia, eficiencia, y su habilidad para evolucionar [Familiarizarse] • Discutir acerca de sistemas operativos cliente-servidor, en red, distribuidos y cómo se diferencian de los sistemas operativos de un solo usuario [Familiarizarse] • Identificar amenazas potenciales a sistemas operativos y las características del diseño de seguridad para protegerse de ellos [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 8: Introducción a redes (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Organización de la Internet (proveedores de servicios de Internet, proveedores de contenido, etc) • Técnicas de Switching (por ejemplo, de circuitos, de paquetes) • Piezas físicas de una red, incluidos hosts, routers, switches, ISPs, inalámbrico, LAN, punto de acceso y firewalls. • Principios de capas (encapsulación, multiplexación) • Roles de las diferentes capas (aplicación, transporte, red, enlace de datos, física) 	<ul style="list-style-type: none"> • Articular la organización de la Internet [Familiarizarse] • Listar y definir la terminología de red apropiada [Familiarizarse] • Describir la estructura en capas de una arquitectura típica en red [Familiarizarse] • Identificar los diferentes tipos de complejidad en una red (bordes, núcleo, etc.) [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 9: Entrega confiable de datos (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Control de errores (técnicas de retransmisión, temporizadores) • El control de flujo (agradecimientos, ventana deslizante) • Problemas de rendimiento (pipelining) • TCP 	<ul style="list-style-type: none"> • Describir el funcionamiento de los protocolos de entrega fiables [Familiarizarse] • Listar los factores que afectan al rendimiento de los protocolos de entrega fiables [Familiarizarse] • Diseñar e implementar un protocolo confiable simple [Usar]
Lecturas : [Brookshear2019]	

Unidad 10: Análisis Básico (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Diferencias entre el mejor, el esperado y el peor caso de un algoritmo. • Análisis asintótico de complejidad de cotas superior y esperada. • Definición formal de la Notación Big O. • Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial. • Medidas empíricas de desempeño. • Compensación entre espacio y tiempo en los algoritmos. • Uso de la notación Big O. 	<ul style="list-style-type: none"> • Explique a que se refiere con “mejor”, “esperado” y “peor” caso de comportamiento de un algoritmo [Familiarizarse] • En el contexto de algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Evaluar] • Determine informalmente el tiempo y el espacio de complejidad de simples algoritmos [Usar] • Indique la definición formal de Big O [Familiarizarse] • Lista y contraste de clases estándares de complejidad [Familiarizarse] • Realizar estudios empíricos para validar una hipótesis sobre runtime stemming desde un análisis matemático Ejecute algoritmos con entrada de varios tamaños y compare el desempeño [Evaluar] • Da ejemplos que ilustran las compensaciones entre espacio y tiempo que se dan en los algoritmos [Familiarizarse] • Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Usar] • Explicar el uso de la notación theta grande, omega grande y o pequeña para describir la cantidad de trabajo hecho por un algoritmo [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 11: Algoritmos y Estructuras de Datos fundamentales (8)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo. • Algoritmos de búsqueda secuencial y binaria. • Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción) • Algoritmos de ordenamiento con peor caso o caso promedio en $O(N \lg N)$ (Quicksort, Heapsort, Mergesort) • Tablas Hash, incluyendo estrategias para evitar y resolver colisiones. • Árboles de búsqueda binaria: <ul style="list-style-type: none"> – Operaciones comunes en árboles de búsqueda binaria como seleccionar el mínimo, máximo, insertar, eliminar, recorrido en árboles. • Grafos y algoritmos en grafos: <ul style="list-style-type: none"> – Representación de grafos (ej., lista de adyacencia, matriz de adyacencia) – Recorrido en profundidad y amplitud • Montículos (Heaps) • Grafos y algoritmos en grafos: <ul style="list-style-type: none"> – Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) – Árbol de expansión mínima (algoritmos de Prim y Kruskal) • Búsqueda de patrones y algoritmos de cadenas/texto (ej. búsqueda de subcadena, búsqueda de expresiones regulares, algoritmos de subsecuencia común más larga) 	<ul style="list-style-type: none"> • Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse] • Resolver problemas usando algoritmos básicos de grafos, incluyendo búsqueda por profundidad y búsqueda por amplitud [Usar] • Implementar algoritmos numéricos básicos [Usar] • Implementar algoritmos de búsqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuadráticos y $O(N \log N)$ [Usar] • Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, búsqueda y hashing [Familiarizarse] • Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Explicar como el balanceamiento del árbol afecta la eficiencia de varias operaciones de un árbol de búsqueda binaria [Familiarizarse] • Resolver problemas usando algoritmos básicos de grafos, incluyendo búsqueda por profundidad y búsqueda por amplitud [Usar] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección, y para implementar el algoritmo en un contexto en específico [Evaluar]
Lecturas : [Brookshear2019]	

Unidad 12: Sistemas de Bases de Datos (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Enfoque y Evolución de Sistemas de Bases de Datos. • Componentes del Sistema de Bases de Datos. • Diseño de las funciones principales de un DBMS. • Arquitectura de base de datos e independencia de datos. • Uso de un lenguaje de consulta declarativa. • Sistemas de apoyo a contenido estructurado y / o corriente. • Enfoques para la gestión de grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce). 	<ul style="list-style-type: none"> • Describe los enfoques principales para almacenar y procesar largos volúmenes de datos [Familiarizarse] • Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Familiarizarse] • Describe los componentes de un sistema de bases de datos y da ejemplos de su uso [Familiarizarse] • Cita las metas básicas, funciones y modelos de un sistema de bases de datos [Familiarizarse] • Describe los componentes de un sistema de bases de datos y da ejemplos de su uso [Familiarizarse] • Identifica las funciones principales de un SGBD y describe sus roles en un sistema de bases de datos [Familiarizarse] • Explica las características que distinguen un esquema de base de datos de aquellos basados en la programación de archivos de datos [Familiarizarse] • Usa un lenguaje de consulta declarativo para recoger información de una base de datos [Usar] • Describe las capacidades que las bases de datos brindan al apoyar estructuras y/o la secuencia de flujo de datos, ejm. texto [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 13: Programación orientada a objetos (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> ● Diseño orientado a objetos: <ul style="list-style-type: none"> – Descomposición en objetos que almacenan estados y poseen comportamiento – Diseño basado en jerarquía de clases para modelamiento ● Definición de las categorías, campos, métodos y constructores. ● Las subclases, herencia y método de alteración temporal. ● Asignación dinámica: definición de método de llamada. ● Subtipificación: <ul style="list-style-type: none"> – Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos. – Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos. – Relación entre subtipos y la herencia. ● Lenguajes orientados a objetos para la encapsulación: <ul style="list-style-type: none"> – privacidad y la visibilidad de miembros de la clase – Interfaces revelan único método de firmas – clases base abstractas ● Uso de colección de clases, iteradores, y otros componentes de la librería estándar. 	<ul style="list-style-type: none"> ● Diseñar e implementar una clase [Usar] ● Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar] ● Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar] ● Comparar y contrastar (1) el enfoque proceduracional/funcional- definiendo una función por cada operación con el uso de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar] ● Explicar la relación entre la herencia orientada a objetos (código compartido y <i>overriding</i>) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse] ● Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar] ● Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, seleccionar la forma más natural por cada lenguaje [Usar]
Lecturas : [Brookshear2019]	

Unidad 14: Procesos de Software (4)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Consideraciones a nivel de sistemas, ejem., la interacción del software con su entorno. • Introducción a modelos del proceso de software (e.g., cascada, incremental, ágil): <ul style="list-style-type: none"> – Actividades con ciclos de vida de software. • Programación a gran escala versus programación individual. • Evaluación de modelos de proceso de software. • Conceptos de calidad de software. • Mejoramiento de procesos. • Modelos de madurez de procesos de software. • Mediciones del proceso de software. 	<ul style="list-style-type: none"> • Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Familiarizarse] • Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Familiarizarse] • Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Familiarizarse] • Diferenciar entre las fases de desarrollo de software [Familiarizarse] • Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Familiarizarse] • Explicar el papel de los modelos de madurez de procesos en la mejora de procesos [Familiarizarse] • Comparar varios modelos comunes de procesos con respecto a su valor para el desarrollo de las clases particulares de sistemas de software, teniendo en cuenta diferentes aspectos tales como, estabilidad de los requisitos, tamaño y características no funcionales [Usar] • Definir la calidad del software y describir el papel de las actividades de aseguramiento de la calidad en el proceso de software [Familiarizarse] • Describir el objetivo y similitudes fundamentales entre los enfoques de mejora de procesos [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 15: Cuestiones fundamentales (2)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Descripción general de los problemas de Inteligencia Artificial, ejemplos recientes de aplicaciones de Inteligencia artificial. • ¿Qué es comportamiento inteligente? <ul style="list-style-type: none"> – El Test de Turing – Razonamiento Racional versus No Racional 	<ul style="list-style-type: none"> • Describir el test de Turing y el experimento pensado "cuarto chino" (<i>Chinese Room</i>) [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 16: Estrategias de búsquedas básicas (1)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Espacios de Problemas (estados, metas y operadores), solución de problemas mediante búsqueda. • Factored representation (factoring state hacia variables) • Uninformed search (breadth-first, depth-first, depth-first with iterative deepening) • Heurísticas y búsqueda informada (hill-climbing, generic best-first, A*) • El espacio y el tiempo de la eficiencia de búsqueda. • Dos jugadores juegos (introducción a la búsqueda minimax). • Satisfacción de restricciones (backtracking y métodos de búsqueda local). 	<ul style="list-style-type: none"> • Formula el espacio eficiente de un problema para un caso expresado en lenguaje natural (ejm. Inglés) en términos de estados de inicio y final, así como sus operadores [Usar] • Describe el rol de las heurísticas y describe los intercambios entre completitud, óptimo, complejidad de tiempo, y complejidad de espacio [Familiarizarse] • Describe el problema de la explosión combinatoria del espacio de búsqueda y sus consecuencias [Familiarizarse] • Selecciona e implementa un apropiado algoritmo de búsqueda no informado para un problema, y describe sus complejidades de tiempo y espacio [Usar] • Selecciona e implementa un apropiado algoritmo de búsqueda no informado para un problema, y describe sus complejidades de tiempo y espacio [Usar] • Evalúa si una heurística dada para un determinado problema es admisible/puede garantizar una solución óptima [Evaluar] • Compara y contrasta tópicos de búsqueda básica con temas jugabilidad de juegos [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 17: Aprendizaje Automático Básico (1)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> Definición y ejemplos de la extensa variedad de tareas de aprendizaje de máquina, incluida la clasificación. Aprendizaje inductivo Aprendizaje simple basado en estadísticas, como el clasificador ingenuo de Bayes, árboles de decisión. El problema exceso de ajuste. Medición clasificada con exactitud. 	<ul style="list-style-type: none"> Listar las diferencias entre los tres principales tipos de aprendizaje: supervisado, no supervisado y por refuerzo [Familiarizarse] Identificar ejemplos de tareas de clasificación, considerando las características de entrada disponibles y las salidas a ser predecidas [Familiarizarse] Describir el sobre ajuste (<i>overfitting</i>) en el contexto de un problema [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 18: Conceptos Fundamentales (2)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> Aplicaciones multimedia, incluyendo interfaces de usuario, edición de audio y vídeo, motores de juego, cad, visualización, realidad virtual. Digitalización de datos analógicos, la resolución y los límites de la percepción humana, por ejemplo, los píxeles de la pantalla visual, puntos para impresoras láser y muestras de audio El uso de las API estándar para la construcción de interfaces de usuario y visualización de formatos multimedia estándar Formatos estándar, incluyendo formatos sin pérdidas y con pérdidas. Modelos de color sustractivo Aditivo y (CMYK y RGB) y por qué estos proporcionan una gama de colores. Soluciones de compensación entre el almacenamiento de datos y los datos re-computing es personalizado por vectores y raster en representaciones de imágenes. Animación como una secuencia de imágenes fijas. Almacenamiento doble. 	<ul style="list-style-type: none"> Identificar usos comunes de presentaciones digitales de humanos (por ejemplo, computación gráfica,sonido) [Familiarizarse] Explicar en términos generales cómo las señales analógicas pueden ser representadas por muestras discretas, por ejemplo,cómo las imágenes pueden ser representadas por pixeles [Familiarizarse] Explicar cómo las limitaciones en la percepción humana afectan la selección de la representación digital de señales analógicas [Usar] Describir las diferencias entre técnicas de compresión de imágenes con pérdida y sin pérdida ejemplificando cómo se reflejan en formatos de archivos de imágenes conocidos como JPG, PNG, MP3, MP4, y GIF [Familiarizarse] Describir modelos de color y su uso en los dispositivos de visualización de gráficos [Familiarizarse] Describir las ventajas y desventajas entre el almacenamiento de información vs almacenar suficiente información para reproducir la información, como en la diferencia entre el vector y la representación de la trama [Familiarizarse] Describir los procesos básico de la producción de movimiento continuo a partir de una secuencia de cuadros discretos(algunas veces llamado it flicker fusion) [Familiarizarse] Describir cómo el doble buffer puede eliminar el parpadeo de la animación [Familiarizarse]
Lecturas : [Brookshear2019]	

Unidad 19: Rendering Básico (2)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Renderizado en la naturaleza, por ejemplo, la emisión y dispersión de la luz y su relación con la integración numérica. • Renderizado Forward and Backward (i.e., <i>ray-casting</i> y rasterización) • Representación poligonal • Radiometría básica, triángulos similares y modelos de proyecciones • Afinamiento y Transformaciones de Sistemas de coordenadas • <i>Ray tracing</i> • Visibilidad y oclusión, incluyendo soluciones a este problema, como el almacenamiento en búfer de profundidad, algoritmo del pintor, y el trazado de rayos. • Representación de la ecuación de adelante hacia atrás. • Rasterización triangular simple. • Mapeo de texturas, incluyendo minificación y magnificación (e.g., MIP-mapping trilineal) • Aplicación de la representación de estructuras de datos espaciales. • Muestreo y anti-aliasing. • Gráficos en escena y la canalización de gráficos. 	<ul style="list-style-type: none"> • Discutir el problema de transporte de la luz y su relación con la integración numérica, es decir, se emite luz, dispersa alrededor de la escena, y es medida por el ojo [Familiarizarse] • Describir la tubería básica gráficos y cómo el factor de representación va hacia adelante y atrás en esta [Familiarizarse] • Crear un programa para visualizar modelos 3D de imágenes gráficas simples [Usar] • Derivar la perspectiva lineal de triángulos semejantes por conversión de puntos (x,y,z) a puntos $(x/z, y/z, 1)$ [Usar] • Obtener puntos en 2-dimensiones y 3-dimensiones por aplicación de transformaciones afín [Usar] • Aplicar sistema de coordenadas de 3-dimensiones y los cambios necesarios para extender las operaciones de transformación 2D para manejar las transformaciones en 3D [Usar] • Explicar la dualidad de rastreo de rayos/rasterización para el problema de visibilidad [Familiarizarse] • Implementar simples procedimientos que realicen la transformación y las operaciones de recorte de imágenes simples en 2 dimensiones [Usar] • Calcular las necesidades de espacio en base a la resolución y codificación de color [Evaluar] • Calcular los requisitos de tiempo sobre la base de las frecuencias de actualización, técnicas de rasterización [Evaluar]
Lecturas : [Brookshear2019]	

Unidad 20: Clase de cierre: ¿Cómo funciona un buscador como Google? (2)	
Resultados esperados:	
Temas	Objetivos de Aprendizaje (<i>Learning Outcomes</i>)
<ul style="list-style-type: none"> • Análisis del problema • El índice no crece de forma lineal con relación al tamaño de la información indexada. • El tiempo de respuesta no depende del tamaño de la “base de datos”. • El tiempo de respuesta no depende del número de ocurrencias encontradas. • Combinando diversas estructuras de datos para llegar a la solución. • Analizando la escalabilidad de la solución. 	<ul style="list-style-type: none"> • Entender los principios bajo los cuales se crea un motor de búsqueda [Usar] • Aplicar correctamente estructuras de datos para dar solución al problema [Usar] • Aplicar los conceptos relacionados a complejidad algorítmica en motor de búsqueda [Usar].
Lecturas : [Brookshear2019]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

***** EVALUATION MISSING *****

10. BIBLIOGRAFÍA BÁSICA