Universidad de Ingeniería y Tecnología
Escuela Profesional de
Ciencia de la Computación
Silabo del curso – Periodo Académico 2018-I

1. **Código del curso y nombre:** CS212. Análisis y Diseño de Algoritmos
2. **Créditos:** 4
3. **Horas de Teoría y Laboratorio:** 2 HT; 4 HP;
4. **Docente(s)**

   Atención previa coordinación con el profesor
5. **Bibliografía**

[Als99] H. Alsuwaiyel. *Algorithms: Design Techniques and Analysis*. World Scientific, 1999. ISBN: 9789810237400.

[DPV06] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill Education, 2006. ISBN: 9780073523408.

[GT09] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. 2nd. John Wiley & Sons, Inc., 2009. ISBN: 0470088540, 9780470088548.

[Knu97] D.E. Knuth. *The Art of Computer Programming: Fundamental algorithms Vol 1*. Third Edition. Addison-Wesley, 1997. ISBN: 9780201896831. URL: `http://www-cs-faculty.stanford/~knuth/taocp.html`.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321295358.

[Raw92] G.J.E. Rawlins. *Compared to What?: An Introduction to the Analysis of Algorithms*. Computer Science Press, 1992. ISBN: 9780716782438.

[RS09] Thomas H. Cormen; Charles E. Leiserson ; Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844.

[SF13] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Pearson Education, 2013. ISBN: 9780133373486.

[SW11] R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, 2011. ISBN: 9780132762564.

[Tar83] Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, 1983. ISBN: 0-89871-187-8.

6. **Información del curso**

   (a) **Breve descripción del curso** Un algoritmo es, esencialmente, un conjunto bien definido de reglas o instrucciones que permitan resolver un problema computacional. El estudio teórico del desempeño de los algoritmos y los recursos utilizados por estos, generalmente tiempo y espacio, nos permite evaluar si un algoritmo es adecuado para un resolver un problema específico, compararlo con otros algoritmos para el mismo problema o incluso delimitar la frontera entre lo viable y lo imposible.

   Esta materia es tan importante que incluso Donald E. Knuth definió a Ciencia de la Computación como el estudio de algoritmos.

   En este curso serán presentadas las técnicas más comunes utilizadas en el análisis y diseño de algoritmos eficientes, con el propósito de aprender los principios fundamentales del diseño, implementación y análisis de algoritmos para la solución de problemas computacionales.

   (b) **Prerrequisitos:** CS210. Algoritmos y Estructuras de Datos. (4^{to} Sem)

   (c) **Tipo de Curso:** Obligatorio

7. **Competencias**

- Desarrollar la capacidad para evaluar la complejidad y calidad de algoritmos propuestos para un determinado problema.
- Estudiar los algoritmos más representativos, introductorios de las clases más importantes de problemas tratados en computación.
- Desarrollar la capacidad de resolución de problemas algorítmicos utilizando los principios fundamentales de diseño de algoritmos aprendidos.
- Ser capaz de responder a las siguientes preguntas cuando le sea presentado un nuevo algoritmo: ¿Cuán buen desempeño tiene?, ¿Existe una mejor forma de resolver el problema?

8. Contribución a los resultados (*Outcomes*)

a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (**Evaluar**)

b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (**Evaluar**)

h) Incorporarse a un proceso de aprendizaje profesional continuo. (**Usar**)

i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (**Usar**)

9. Competencias (IEEE)

C1. La comprensión intelectual y la capacidad de aplicar las bases matemáticas y la teoría de la informática (*Computer Science*).⇒ **Outcome a**

C2. Capacidad para tener una perspectiva crítica y creativa para identificar y resolver problemas utilizando el pensamiento computacional.⇒ **Outcome b**

C3. Una comprensión intelectual de, y el aprecio por el papel central de los algoritmos y estructuras de datos.⇒ **Outcome b**

C5. Capacidad para implementar algoritmos y estructuras de datos en el software.⇒ **Outcome i**

C6. Capacidad para diseñar y poner en práctica las unidades estructurales mayores que utilizan algoritmos y estructuras de datos y las interfaces a través del cual estas unidades se comunican.⇒ **Outcome i**

C9. Comprensión de las limitaciones de la computación, incluyendo la diferencia entre lo que la computación es inherentemente incapaz de hacer frente a lo que puede lograrse a través de un futuro de ciencia y tecnología.⇒ **Outcome a**

C16. Capacidad para identificar temas avanzados de computación y de la comprensión de las fronteras de la disciplina.⇒ **Outcome h**

10. Lista de temas a estudiar en el curso

1. Análisis Básico
2. Estrategias Algorítmicas
3. Algoritmos y Estructuras de Datos fundamentales
4. Computabilidad y complejidad básica de autómatas
5. Estructuras de Datos Avanzadas y Análisis de Algoritmos

11. Metodologia y Evaluación
Metodología:

Sesiones Teóricas:
El desarrollo de las sesiones teóricas está focalizado en el estudiante, a través de su participación activa, resolviendo problemas relacionados al curso con los aportes individuales y discutiendo casos reales de la industria. Los alumnos desarrollarán a lo largo del curso un proyecto de aplicación de las herramientas recibidas en una empresa.

Sesiones de Laboratorio:
Las sesiones prácticas se desarrollan en laboratorio. Las prácticas de laboratorio se realizan en equipos para fortalecer su

comunicación. Al inicio de cada laboratorio se explica el desarrollo de la práctica y al término se destaca las principales conclusiones de la actividad en forma grupal.

Exposiciones individuales o grupales:
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.

Lecturas:
A lo largo del curso se proporcionan diferentes lecturas, las cuales son evaluadas. El promedio de las notas de las lecturas es considerado como la nota de una práctica calificada. El uso del campus virtual UTEC Online permite a cada estudiante acceder a la información del curso, e interactuar fuera de aula con el profesor y con los otros estudiantes.
Sistema de Evaluación:

12. Contenido

<table>
<tr><th colspan="2">Unidad 1: Análisis Básico (10)</th></tr>
<tr><th colspan="2">Competences esperadas: C1</th></tr>
<tr><th>Objetivos de Aprendizaje</th><th>Tópicos</th></tr>
<tr><td>
Explique a que se refiere con “mejor”, “esperado” y “peor” caso de comportamiento de un algoritmo [Evaluar]
En el contexto de a 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 [Evaluar]
Indique la definición formal de Big O [Evaluar]
Lista y contraste de clases estándares de complejidad [Evaluar]
Use la notación formal de la Big O para dar límites superiores asintóticos en la complejidad de tiempo y espacio de los algoritmos [Evaluar]
Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Evaluar]
Explicar el uso de la notación theta grande, omega grande y o pequeña para describir la cantidad de trabajo hecho por un algoritmo [Evaluar]
Usar relaciones recurrentes para determinar el tiempo de complejidad de algoritmos recursivamente definidos [Evaluar]
Resuelve relaciones de recurrencia básicas, por ejemplo. usando alguna forma del Teorema Maestro [Evaluar]
</td><td>
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.
Uso de la notación Big O.
Relaciones recurrentes.
Análisis de algoritmos iterativos y recursivos.
Algunas versiones del Teorema Maestro.
</td></tr>
<tr><td colspan="2">Lecturas : [KT05], [DPV06], [RS09], [SF13], [Knu97]</td></tr>
</table>

Unidad 2: Estrategias Algorítmicas (30)	
Competences esperadas: C2	
Objetivos de Aprendizaje	**Tópicos**
• Para cada una de las estrategias (fuerza bruta, algoritmo goloso, divide y vencerás, recursividad en reversa y programación dinámica), identifica un ejemplo práctico en el cual se pueda aplicar [Evaluar] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar] • Usa un algoritmo de divide-y-vencerás para resolver un determinado problema [Evaluar] • Usa programación dinámica para resolver un problema determinado [Evaluar] • Determina el enfoque algorítmico adecuado para un problema [Evaluar]	• Algoritmos de fuerza bruta. • Algoritmos voraces. • Divide y vencerás. • Programación Dinámica.
Lecturas : [KT05], [DPV06], [RS09], [Als99]	

Unidad 3: Algoritmos y Estructuras de Datos fundamentales (10)	
Competences esperadas: C6	
Objetivos de Aprendizaje	**Tópicos**
• Implementar algoritmos numéricos básicos [Evaluar] • Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Evaluar] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Usar] • 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] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Evaluar] • 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] • Describir la propiedad del heap y el uso de heaps como una implementación de colas de prioridad [Evaluar] • Resolver problemas usando algoritmos de grafos, incluyendo camino más corto de una sola fuente y camino más corto de todos los pares, y como mínimo un algoritmo de arbol de expansion minima [Evaluar]	• 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) • Grafos y algoritmos en grafos: – Representación de grafos (ej., lista de adyacencia, matriz de adyacencia) – Recorrido en profundidad y amplitud • Montículos (Heaps) • Grafos y algoritmos en grafos: – Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) – Árbol de expansión mínima (algoritmos de Prim y Kruskal)
Lecturas : [KT05], [DPV06], [RS09], [SW11], [GT09]	

Unidad 4: Computabilidad y complejidad básica de autómatas (2)	
Competences esperadas: C9	
Objetivos de Aprendizaje	**Tópicos**
• Define las clases P y NP [Familiarizarse] • Explique el significado de NP-Completitud [Familiarizarse]	• Introducción a las clases P y NP y al problema P vs. NP. • Introducción y ejemplos de problemas NP- Completos y a clases NP-Completos.
Lecturas : [KT05], [DPV06], [RS09]	

Unidad 5: Estructuras de Datos Avanzadas y Análisis de Algoritmos (8)	
Competences esperadas: C16	
Objetivos de Aprendizaje	**Tópicos**
• Entender el mapeamento de problemas del mundo real a soluciones algorítmicas (ejemplo, problemas de grafos, programas lineares,etc) [Familiarizarse] • Seleccionar y aplicar técnicas de algoritmos avanzadas (ejemplo, randonmización, aproximación) para resolver problemas reales [Usar] • Seleccionar y aplicar técnicas avanzadas de análisis (ejemplo, amortizado, probabilistico,etc) para algoritmos [Usar]	• Grafos (ej. Ordenamiento Topológico, encontrando componentes puertemente conectados) • Algoritmos Teórico-Numéricos (Aritmética Modular, Prueba del Número Primo, Factorización Entera) • Algoritmos aleatorios. • Análisis amortizado. • Análisis Probabilístico.
Lecturas : [KT05], [DPV06], [RS09], [Tar83], [Raw92]	

Universidad de Ingeniería y Tecnología
Escuela Profesional de
Ciencia de la Computación
Silabo del curso – Periodo Académico 2018-I

1. **Código del curso y nombre:** CS342. Compiladores
2. **Créditos:** 4
3. **Horas de Teoría y Laboratorio:** 2 HT; 4 HP;
4. **Docente(s)**

   Atención previa coordinación con el profesor
5. **Bibliografía**

[Aho+08] Alfred Aho et al. *Compiladores. Principios, técnicas y herramientas.* 2nd. ISBN:10-970-26-1133-4. Addison Wesley, 2008.

[Aho90] Alfred Aho. *Compiladores Principios, técnicas y herramientas.* Addison Wesley, 1990.

[ALe96] Karen A.Lemone. *Fundamentos de Compiladores.* CECSA-Mexico, 1996.

[App02] A. W. Appel. *Modern compiler implementation in Java.* 2.a edición. Cambridge University Press, 2002.

[Lou04a] Kenneth C. Louden. *Construccion de Compiladores Principios y Practica.* Thomson, 2004.

[Lou04b] Kenneth C. Louden. *Lenguajes de Programacion.* Thomson, 2004.

[PV98] Terrence W. Pratt and Marvin V.Zelkowitz. *Lenguajes de Programacion Diseño e Implementacion.* Prentice-Hall Hispanoamericana S.A., 1998.

[TS98] Bernard Teufel and Stephanie Schmidt. *Fundamentos de Compiladores.* Addison Wesley Iberoamericana, 1998.

6. **Información del curso**

   (a) **Breve descripción del curso** Que el alumno conozca y comprenda los conceptos y principios fundamentales de la teoría de compilación para realizar la construcción de un compilador

   (b) **Prerrequisitos:** CS211. Teoría de la Computación. (4^{to} Sem)

   (c) **Tipo de Curso:** Obligatorio

7. **Competencias**

   - Conocer las técnicas básicas empleadas durante el proceso de generación intermedio, optimización y generación de código.
   - Aprender a implementar pequeños compiladores.

8. **Contribución a los resultados (*Outcomes*)**

   a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (**Evaluar**)

   b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (**Evaluar**)

   j) Aplicar la base matemática, principios de algoritmos y la teoría de la Ciencia de la Computación en el modelamiento y diseño de sistemas computacionales de tal manera que demuestre comprensión de los puntos de equilibrio involucrados en la opción escogida. (**Evaluar**)

9. **Competencias (IEEE)**

   C8. Entendimiento de lo que las tecnologías actuales pueden y no pueden lograr.⇒ **Outcome a**