UNIVERSIDAD EAFIT

PLANTILLA BÁSICA PARA EL DISEÑO CURRICULAR DE PROGRAMA

1. IDENTIFICACIÓN:

NOMBRE ESCUELA:	**Escuela de Ingeniería**
NOMBRE DEPARTAMENTO:	**Departamento de Informática y Sistemas**
NOMBRE DEL PROGRAMA:	**Ingeniería de sistemas**
NOMBRE ASIGNATURA EN ESPAÑOL:	**Estructuras de Datos y Algoritmos 1**
NOMBRE ASIGNATURA EN INGLÉS:	**Introduction to Algorithms and Data Structures**
MATERIA PREREQUISITO:	**Fundamentos de Programación**
CÓDIGO:	**ST0245**
SEMESTRE DE UBICACIÓN:	**II**
INTENSIDAD HORARIA SEMANAL:	**3 horas**
INTENSIDAD HORARIA SEMESTRAL:	**48 horas**
CRÉDITOS:	**Generado por el sistema**
CARACTERÍSTICAS:	**Generados por el sistema**
	Generada por el sistema

2. JUSTIFICACIÓN DEL CURSO:

El Ingeniero de Sistemas egresado de Eafit debe estar en capacidad de hacer el análisis, diseño e implementación de sistemas intensivos en Software, y, por esa razón, es imperativo que sea capaz de seleccionar un algoritmo y realizar su implementación para resolver eficientemente problemas.

3. PROPÓSITO U OBJETIVO GENERAL DEL PROGRAMA ACADÉMICO:

Resolver un problema con un algoritmo que utiliza estructuras de datos fundamentales (grafos, listas, pilas, colas, y árboles), calcular la complejidad del algoritmo, y argumentar sus criterios de selección de algoritmo y estructura de datos

3.1. COMPETENCIAS GENÉRICAS:

1. Realizar búsqueda de literatura impresa y electrónica (CDIO 2.2.2)
2. Capacidad de trabajar en distintos tipos de equipos (CDIO 3.1.5)
3. Capacidad de comunicación escrita efectiva (CDIO 3.2.3)
4. Capacidad de comunicación por medios Multimedia (CDIO 3.2.4)
5. Capacidad de comunicación por presentaciones orales (CDIO 3.2.6)

3.2 COMPETENCIAS ESPECÍFICAS:

1. Resolver un problema con un algoritmo que utiliza estructuras de datos fundamentales (grafos, listas, pilas, colas, y árboles), calcular la complejidad del algoritmo, y argumentar sus criterios de selección de algoritmo y estructura de datos

3.3 RESULTADOS DE APRENDIZAJE

Competencia específica:

1. Resolver un problema con un algoritmo que utiliza estructuras de datos fundamentales (grafos, listas, pilas, colas, y árboles), calcular la complejidad del algoritmo, y argumentar sus criterios de selección de algoritmo y estructura de datos

Resultados de aprendizaje de la competencia:

1. Calcula la complejidad de un algoritmo (el comportamiento asintótico)
2. Conoce las notaciones Big O, o, Omega, Theta
3. Asocia el cálculo de complejidad de un algoritmo con la disponibilidad de recursos, la dimensión del problema y la naturaleza de los datos (por ejemplo, si están ordenados)
4. Conoce las estructuras de datos fundamentales: grafos, listas, pilas, colas, y árboles
5. Justifica el uso de una estructura de datos para resolver un problema utilizando criterios técnicos
6. Identifica el caso base y la relación de recurrencia para resolver un problema recursivamente
7. Diferencia una estructura de datos conceptual y sus implementaciones
8. Aplica los conceptos de referencias, encapsulamiento, iteradores y tipos de datos paramétricos en la implementación de estructuras de datos

4. DESCRIPCION DE LOS CONTENIDOS: TEMAS Y SUBTEMAS:

1. Cálculo de complejidad
   1.1 Preliminares matemáticos (sumatorias y relaciones de recurencia)
   1.2 Definición y notaciones (omega, O, theta)
2. Estructuras de datos fundamentales
   2.1 Grafos
   2.2 Listas, pilas, colas
   2.3 Árboles

5. ESTRATEGIAS METODOLÓGICAS Y DIDÁCTICAS:

Se sugiere al maestro orientar clases magistrales donde se exponen los temas del curso, tareas donde se trabaja en la aplicación de algoritmos para resolver eficientemente problemas, y se implementan en un lenguaje de programación. Adicionalmente, hacer énfasis en la práctica final de semestre donde se resuelve un problema que se implementa en un lenguaje de programación y orientar a los estudiantes en la escritura de un reporte técnico sobre la práctica final. No se recomienda dedicar mucho tiempo a la fundamentación matemática necesaria para calcular la complejidad computacional (por ejemplo, sumatorias, teorema maestro y resolución de ecuaciones de recurrencia); en su lugar, dedicar ese tiempo a la interpretación de los cálculos de complejidad y hacer uso de herramientas computacionales para resolver las ecuaciones y sumatorias como Wolfram Alpha.

5.1. Metodología docente y estimación de volumen de trabajo del estudiante

5.2. Temporalización o cronograma

1. Cálculo de complejidad (6h)
    - 1.1 Preliminares matemáticos (3h)
    - 1.2 Definición y notaciones (3h)
2. Estructuras de datos fundamentales (33h)
    - 2.1 Grafos (12h)
    - 2.2 Listas, pilas, colas (9h)
    - 2.3 Árboles (12h)
3. Evaluaciones parciales y del proyecto (9h)
    - 3.1 Parcial 1 (3h)
    - 3.2 Parcial 2 (3h)
    - 3.3 Sustentación del proyecto (3h)

6. RECURSOS:

Se sugiere al maestro utilizar las salas de cómputo disponibles en la universidad para realizar ejercicios de programación que ayuden a afianzar los conceptos teóricos y a desarrollar habilidades en la implementación de algoritmos.

6.1. Locativos:

6.2. Tecnológicos:

6.3. Didácticos:

6.4. Bibliográficos

7. CRITERIOS Y POLÍTICAS DE SEGUIMIENTO Y EVALUACIÓN ACADÉMICA:

Criterios de Evaluación

Se recomienda al maestro realizar un seguimiento periódico y sumativo a través de tareas que evaluén no sólo el análisis de algoritmos, sino también la capacidad de aplicar algoritmos para solucionar eficientemente problemas. Adicionalmente, se propone realizar un proyecto donde deba solucionarse un problema del mundo real que no tenga una única respuesta y que implique elegir la mejor solución para el problema, teniendo como principal criterio la eficiencia en tiempo y espacio. Se sugiere realizar 2 evaluaciones parciales donde se evalúe, principalmente, el análisis de algoritmos, y la aplicación de algoritmos para resolver problemas. Se proponen los siguientes porcentajes para las evaluaciones: Parcial 1 20%, Parcial 2 25%, Proyecto 30%, Tareas 25%.

A continuación la relación entre la evaluación y los objetivos de aprendizaje

Tareas 25 % - [3, 4, 6, 7, 8] Debe haber retroalimentación
Parcial 1 20 % [1,2,6]
Parcial 2 25 % [1,2,4,5,6]
Proyecto 30% [1,2,3,4,5,8]

8. BLIOGRAFIA GENERAL:

Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2nd edition), 2011

Narasimha Karumanchi, Data Structures and Algorithms made easy in Python, (2nd edition), 2011

Robert Lafore, Data Structures and Algorithms in Java (2nd edition), 2002

9. NOMBRE DEL PROFESOR COORDINADOR DE MATERIA Y NOMBRE DE PROFESORES DE LA MATERIA QUE PARTICIPARON EN LA ELABORACIÓN.

Coordinador: Mauricio Toro
Participantes: Juan G. Lalinde, Juan C. Montoya, Edwin Montoya, Paola Vallejo, Juan F. Cardona

10. REQUISITOS DEL PROCESOS DE ASEGURAMIENTO DE LA CALIDAD

10.1. Versión número: _____

10.2. Fecha elaboración: _____

10.3. Fecha actualización: _____

10.4. Responsable: _____

10.5. Aprobación: _____

Jefe del Programa

Jefe del Departamento