# Descubriendo el Mundo de los Algoritmos: Técnicas y Representación para Ingenieros

*Ingeniería | Ingeniería de sistemas | Aprendizaje Basado en Problemas*

## Descripción

Este plan de clase está diseñado para estudiantes universitarios de Ingeniería de Sistemas y tiene como propósito introducir y profundizar en los fundamentos de los algoritmos y las técnicas para su representación. A partir de problemas reales y simulados, los estudiantes explorarán cómo se construyen, interpretan y aplican algoritmos, utilizando diversas técnicas de representación como pseudocódigo, diagramas de flujo y otros métodos gráficos. El aprendizaje se centra en desarrollar pensamiento crítico y habilidades prácticas para diseñar soluciones efectivas en el contexto de la ingeniería.

Comprender la lógica detrás de los algoritmos es esencial para cualquier ingeniero, ya que esta habilidad es la base para resolver problemas complejos y optimizar procesos en la industria tecnológica y más allá. Los estudiantes aprenderán a analizar problemas, construir algoritmos adecuados y representarlos visualmente para facilitar su comprensión y posterior implementación en sistemas computacionales. Este conocimiento no solo fortalece su formación académica, sino que también los prepara para enfrentar desafíos reales en su futuro profesional.

## Objetivos de Aprendizaje

- Analizar problemas reales para identificar la estructura lógica requerida en un algoritmo.
- Diseñar algoritmos utilizando técnicas formales de representación como pseudocódigo y diagramas de flujo.
- Evaluar la eficacia de diferentes técnicas de representación para comunicar algoritmos de manera clara y precisa.
- Crear soluciones algorítmicas aplicables a problemas simulados y reales del ámbito de la ingeniería de sistemas.
- Argumentar la importancia del diseño adecuado de algoritmos en la optimización y eficiencia de sistemas computacionales.

## Recursos Necesarios

- Computadoras o laptops con software de diagramas (por ejemplo, draw.io o Microsoft Visio).
- Pizarras blancas y marcadores.
- Proyector multimedia para presentaciones y videos.
- Material impreso con ejemplos de algoritmos y técnicas de representación.
- Acceso a plataforma digital para compartir documentos y recursos (Google Drive, Moodle o similar).
- Cuadernos y bolígrafos para toma de notas y bosquejos.

## Requisitos Previos

- Conocimientos básicos de lógica matemática y programación introductoria.
- Familiaridad con estructuras básicas de control como condicionales y bucles.
- Habilidades para trabajar en equipo y comunicarse de forma clara.
- Capacidad para analizar problemas simples y descomponerlos en pasos secuenciales.

## Actividades

### Sesión 1: Introducción y Construcción de Algoritmos

### Fase de Inicio

**Tiempo estimado: 10 minutos**

**Propósito de la sesión:**

Conectar a los estudiantes con el concepto de algoritmos, su importancia en la ingeniería y establecer el objetivo de diseñar y representar algoritmos para resolver problemas reales.

**Activación de conocimientos previos:**

**Docente:** "¿Recuerdan algún proceso que hayan seguido paso a paso para resolver un problema o realizar una tarea diaria? Por ejemplo, preparar una receta, armar un mueble o configurar un dispositivo. ¿Podrían describir brevemente los pasos que siguieron?"

**Estudiantes:** Responden brevemente describiendo pasos de un proceso cotidiano.

**Motivación y enganche:**

**Docente:** "Un algoritmo está detrás de casi todo lo que hace la tecnología que usamos cada día, desde aplicaciones móviles hasta sistemas complejos en la nube. ¿Sabían que diseñar un algoritmo eficiente puede ahorrar millones de dólares y tiempo en una empresa? Hoy empezaremos a descubrir cómo diseñar esos algoritmos."

**Contextualización:**

**Docente:** "Al finalizar esta sesión, serán capaces de analizar un problema práctico y diseñar un algoritmo que lo resuelva, representándolo de manera clara, una habilidad fundamental para su carrera en ingeniería de sistemas."

### Fase de Desarrollo

**Tiempo estimado: 100 minutos**

**Presentación del contenido:**

**Docente:** Presenta brevemente con ejemplos reales la definición de algoritmo y las técnicas básicas de representación: pseudocódigo y diagramas de flujo, enfatizando sus usos y ventajas.

### Actividad 1: Análisis y diseño de algoritmo en equipo

- **Objetivo:** Analizar un problema y diseñar un algoritmo en pseudocódigo.
- **Instrucciones:**
  - **Docente:** Divide a la clase en grupos de 3-4 estudiantes. Entrega un problema real simulado, por ejemplo: "Diseñar un algoritmo para calcular el promedio de notas de un estudiante y determinar si aprueba o no."
  - Los estudiantes analizan el problema, identifican las entradas, salidas y procesos necesarios.
  - Diseñan un algoritmo en pseudocódigo que resuelva el problema.
  - Preparan para presentar su algoritmo al grupo.
- **Organización:** Grupos de 3-4 estudiantes.
- **Producto:** Documento con el algoritmo en pseudocódigo.
- **Tiempo:** 40 minutos.
- **Rol del docente:** Circula entre grupos, plantea preguntas como "¿Qué pasos son necesarios para obtener la salida?", "¿Cómo manejan condiciones como aprobar o reprobar?", "¿El pseudocódigo es claro para que otro lo entienda?"

### Actividad 2: Representación gráfica mediante diagramas de flujo

- **Objetivo:** Representar el algoritmo diseñado en un diagrama de flujo claro y preciso.
- **Instrucciones:**
  - **Docente:** Solicita a los mismos grupos que a partir del pseudocódigo desarrollen un diagrama de flujo usando herramientas digitales o papel y marcadores.
  - Se enfatiza el uso correcto de símbolos y secuencia lógica.
  - Cada grupo prepara una breve explicación de su diagrama para compartir con la clase.
- **Organización:** Grupos de 3-4 estudiantes.
- **Producto:** Diagrama de flujo del algoritmo.
- **Tiempo:** 50 minutos.
- **Rol del docente:** Asiste en el manejo de símbolos, verifica la coherencia lógica y hace preguntas de profundización: "¿Cómo refleja el diagrama la estructura del algoritmo?", "¿Qué ventaja tiene esta representación para entender el proceso?"

### Diferenciación

**Para estudiantes que terminan temprano:** Invitarlos a mejorar el diagrama incorporando validaciones o casos especiales (por ejemplo, validar que las notas estén dentro de un rango válido).

**Para estudiantes que necesitan apoyo:** Proporcionar ejemplos guiados de pseudocódigo y diagramas de flujo para que puedan seguir paso a paso, y ofrecer ayuda personalizada.

### Transición:

**Docente:** "Ahora que han diseñado y representado su primer algoritmo, en la próxima sesión analizaremos técnicas avanzadas y cómo aplicar estos conceptos en problemas más complejos y reales."

## Fase de Cierre

### Tiempo estimado: 10 minutos

### Síntesis:

**Docente:** Propone que cada grupo comparta en 2 minutos la idea central de su algoritmo y cómo lo representaron gráficamente.

### Reflexión metacognitiva:

- "¿Qué elementos del problema fueron más difíciles de traducir a un algoritmo?"
- "¿Cómo les ayudó la representación gráfica a entender mejor el algoritmo?"
- "¿Qué ventajas encuentran en usar pseudocódigo frente a otros métodos?"

### Retroalimentación:

**Docente:** Ofrece comentarios constructivos sobre la claridad, lógica y creatividad en los algoritmos y diagramas presentados, resaltando aciertos y áreas de mejora.

### Transferencia:

**Docente:** Explica que en la siguiente sesión aplicarán estas técnicas para resolver problemas más complejos y se enfocarán en la optimización y análisis crítico de algoritmos.

### Tarea o reto:

**Docente:** Asignar a los estudiantes individualmente diseñar un algoritmo y su diagrama de flujo para un problema cotidiano de su elección (por ejemplo, organizar horarios, cálculo de gastos, etc.) para compartir en la próxima sesión.

## Sesión 2: Aplicación y Optimización de Algoritmos

## Fase de Inicio

### Tiempo estimado: 10 minutos

### Propósito de la sesión:

Revisar brevemente la tarea asignada y preparar a los estudiantes para aplicar técnicas de optimización y análisis crítico de algoritmos.

### Activación de conocimientos previos:

**Docente:** "¿Quién quiere compartir el algoritmo que diseñaron para su tarea y qué dificultades encontraron?"

**Estudiantes:** Presentan brevemente sus algoritmos y experiencias.

### Motivación y enganche:

**Docente:** "En ingeniería, no basta con tener un algoritmo funcional, sino que debe ser eficiente y adaptable. Hoy aprenderemos cómo mejorar y analizar nuestros algoritmos para que sean más efectivos."

### Contextualización:

**Docente:** Relaciona la optimización de algoritmos con aplicaciones reales como reducción de tiempos de procesamiento y ahorro de recursos en sistemas computacionales.

## Fase de Desarrollo

### Tiempo estimado: 100 minutos

### Presentación del contenido:

**Docente:** Introduce conceptos de complejidad algorítmica básica y estructuras de control avanzadas (condicionales anidados, bucles anidados), explicando su impacto en el diseño y rendimiento.

### Actividad 1: Evaluación crítica de algoritmos

- **Objetivo:** Analizar y evaluar la eficiencia y claridad de algoritmos existentes en pseudocódigo y diagramas de flujo.
- **Instrucciones:**
  - Presentar dos algoritmos que resuelvan el mismo problema pero con diferente estructura y complejidad.
  - En grupos, los estudiantes comparan ambos y discuten cuál es más eficiente y por qué.
  - Proponen mejoras o alternativas para optimizar los algoritmos.
- **Organización:** Grupos de 3-4 estudiantes.
- **Producto:** Informe breve con análisis comparativo y propuestas de mejora.
- **Tiempo:** 45 minutos.
- **Rol del docente:** Facilita la discusión, formula preguntas guía como: "¿Qué partes del algoritmo generan más trabajo?", "¿Cómo cambia el algoritmo si el volumen de datos aumenta?"

### Actividad 2: Diseño de algoritmo optimizado para un problema complejo

- **Objetivo:** Crear un algoritmo eficiente aplicando técnicas de representación y optimización.
- **Instrucciones:**
  - Se presenta un problema más complejo, por ejemplo: "Diseñar un algoritmo para ordenar una lista de números y contar cuántos intercambios se realizan".
  - Los grupos diseñan el algoritmo en pseudocódigo y diagrama de flujo, buscando optimizar pasos y reducir complejidad.
  - Preparan una explicación que justifique las decisiones de diseño y optimización.

- **Organización:** Grupos de 3-4 estudiantes.
- **Producto:** Algoritmo optimizado en pseudocódigo y diagrama de flujo con justificación escrita.
- **Tiempo:** 50 minutos.
- **Rol del docente:** Observa el proceso, fomenta el pensamiento crítico preguntando: "¿Cómo afecta esta estructura al tiempo de ejecución?", "¿Existen pasos redundantes que se puedan eliminar?"

### Diferenciación

**Para estudiantes que terminan temprano:** Retan a diseñar un algoritmo alternativo para el mismo problema con diferente enfoque.

**Para estudiantes que necesitan apoyo:** Se les asigna asistir a un taller guiado con ejemplos y acompañamiento directo para entender estructuras más complejas.

### Transición:

**Docente:** "Con estos ejercicios han aplicado fundamentos, técnicas y análisis crítico, habilidades clave que fortalecerán su desempeño profesional."

## Fase de Cierre

### Tiempo estimado: 10 minutos

### Síntesis:

**Docente:** Propone un mapa mental colectivo en la pizarra con las ideas clave sobre algoritmos, técnicas de representación y optimización discutidas.

### Reflexión metacognitiva:

- "¿Qué aprendieron sobre la importancia de representar claramente un algoritmo?"
- "¿Cómo influye la optimización en la aplicación práctica de un algoritmo?"
- "¿Qué aspectos del diseño algorítmico consideran que deben seguir practicando?"

### Retroalimentación:

**Docente:** Brinda observaciones generales sobre el nivel alcanzado, destacando avances y áreas a reforzar, y responde preguntas finales.

### Transferencia:

**Docente:** Invita a los estudiantes a aplicar estas técnicas de diseño y análisis de algoritmos en futuras asignaturas y proyectos profesionales.

### Tarea o reto:

**Docente:** Proponer que cada estudiante diseñe y documente un algoritmo para un problema de interés personal o profesional, aplicando técnicas vistas y presentándolo en la siguiente semana para retroalimentación.

## Evaluación

**Tipo de evaluación:**

- **Diagnóstica:** En la fase de inicio de la sesión 1, mediante preguntas sobre procesos cotidianos para activar conocimientos previos.
- **Formativa:** Durante las actividades de diseño, representación y análisis de algoritmos en ambas sesiones, con observación directa y retroalimentación continua.
- **Sumativa:** Al cierre de la sesión 2, mediante la presentación del algoritmo optimizado y el análisis crítico realizado en grupo, además de la tarea final individual.

**Criterios de evaluación:**

- Claridad y coherencia en el diseño del algoritmo (vinculado a diseñar algoritmos usando técnicas formales).
- Correcta utilización de técnicas de representación (pseudocódigo y diagramas de flujo) (vinculado a representar algoritmos de manera clara).
- Capacidad para analizar y optimizar algoritmos, justificando mejoras propuestas (vinculado a evaluar la eficacia y optimización).
- Participación activa y trabajo colaborativo en las actividades grupales (vinculado a crear soluciones aplicables y argumentar su importancia).

**Instrumentos sugeridos:**

- Lista de cotejo para evaluar la estructura y claridad del pseudocódigo y diagramas de flujo.
- Rúbrica para análisis crítico y justificación de optimizaciones en algoritmos.
- Observación directa durante actividades grupales y plenarias.
- Autoevaluación y coevaluación para fomentar reflexión sobre el propio aprendizaje y el de pares.

**Evidencias de aprendizaje:**

- Documentos con algoritmos en pseudocódigo diseñados por los grupos y estudiantes individuales.
- Diagramas de flujo que representan los algoritmos.
- Informes de análisis comparativo y optimización de algoritmos.
- Participación activa en discusiones y exposiciones.