FECHA: 2/12/2025

Estructura de datos avanzados

UNIVERSIDAD AUTONÓMA DE NAYARIT

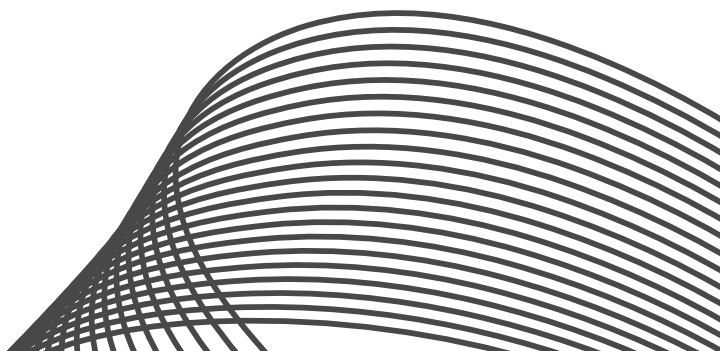
Semana 8

SISTEMAS COMPUTACIONALES

DATOS

ALUMNO: CHRISTIAN GAEL ORTIZ RAMIREZ

DOCENTE: ELIGARDO CRUZ SANCHEZ



INTRODUCCIÓN

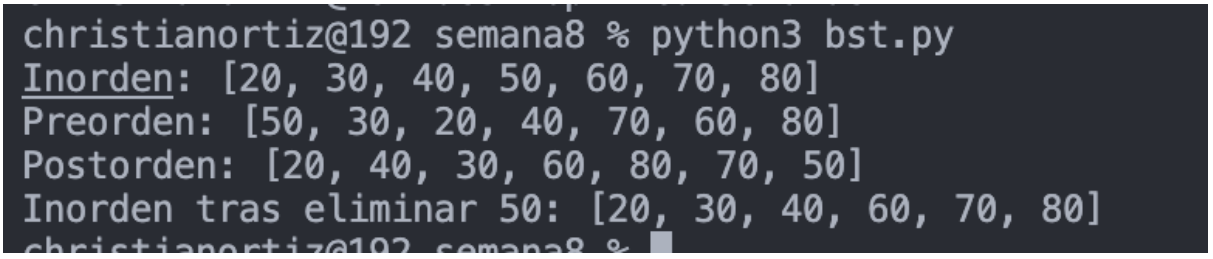
La Semana 8 profundiza en el estudio de estructuras arbóreas avanzadas, fundamentales en el diseño de sistemas eficientes de búsqueda, almacenamiento y compresión. En esta unidad se analizan y aplican árboles AVL, B-Tree, B+ Tree y Huffman, cada uno orientado a resolver problemas específicos relacionados con el manejo de grandes volúmenes de datos. Los árboles AVL permiten mantener operaciones de inserción, búsqueda y eliminación en tiempo O(log n) mediante rotaciones de balanceo; los árboles B y B+ optimizan los accesos a disco y constituyen la base de motores de bases de datos y sistemas de archivos modernos; y los árboles de Huffman permiten realizar compresión óptima basada en frecuencias de aparición. Como integración práctica, se desarrolló un proyecto que indexa un archivo de texto mediante árboles de búsqueda y aplica compresión utilizando Huffman, permitiendo comparar rendimiento, tamaño y complejidad entre las diferentes estructuras. Esta semana combina teoría, análisis de complejidad y aplicación real, consolidando herramientas esenciales para el desarrollo de software especializado en estructuras de datos.

DESARROLLO

bst.py

- Inserción, eliminación, recorridos
- Estructura básica, no balanceada

SALIDA



✔️ avl.py

- Balanceo automático
- Rotaciones LL, RR, LR, RL
- Altura logarítmica garantizada

Salida:

```
AVL inorden: [20, 25, 30]
AVL inorden after inserts: [10, 20, 25, 30, 40, 50]
AVL inorden after delete 20: [10, 25, 30, 40, 50]
```

✔️ huffman.py

- Construcción de árbol por frecuencias
- Tabla de códigos
- Codificación y decodificación

✔️ main.py

- Indexa palabras de un archivo de texto
- Construye códigos Huffman
- Genera archivo comprimido (`sample.huff`)

Resultado:

```
Construyendo índice (BST)...
Palabras únicas en índice (primeras 20): ['accesos', 'actualizaciones', 'archivo', 'avl', 'b', 'basada', 'bst', 'búsqueda', 'como', 'compresión', 'consultas', 'de', 'digital', 'dinámicas', 'disco', 'eficiente', 'en', 'era', 'es', 'este']
Corriendo Huffman...
Códigos Huffman (primeros 20 símbolos): {'a': '000', 'v': '0010000', 'x': '0010001', '.': '001001', 'S': '00101000', 'q': '00101001', 'ú': '00101010', 'h': '00101011', 'á': '001011', 'y': '001100', 'H': '00110100', '+': '00110101', 'g': '0011011', 'B': '0011100', 'A': '00111010', 'T': '00111011', 'b': '001111', 'o': '0100', 'c': '0101', 'l': '01100'}
Archivo comprimido sample.huff creado (contenido en bits, no binario actual)
```

Crea un archivo llamado sample.huff comprimido

COMPARACIÓN DE BST Y AVL

Estructura	Altura	Complejidad	Usarla
Bst	Muy alto (O(n)	Lento si esta degenerado	Casos simples
Avl	altura garantizada	mas rotaciones, más rápido	búsquedas frecuentes

B+ Tree	Ideal para disco	baja altura	BD
Huffman	No para búsquedas	Compresión	Archivos

REFLEXIÓN

La Semana 8 representó uno de los retos más completos del curso, ya que integró distintas estructuras de datos avanzadas en un solo proyecto práctico. Implementar correctamente árboles AVL y Huffman exigió no solo comprender sus fundamentos teóricos, sino también depurar errores relacionados con rotaciones, alturas, manejo de nodos y recorridos. Además, la comparación con B-Tree y B+ Tree permitió entender por qué las estructuras balanceadas son esenciales en sistemas reales, especialmente cuando se manejan grandes volúmenes de datos o almacenamiento en disco.

CONCLUSIÓN

La Semana 8 permitió integrar estructuras avanzadas como AVL y Huffman para resolver problemas reales de búsqueda y compresión, y como punto La IA resultó útil para acelerar partes del proceso, especialmente en la generación de código repetitivo, La diferencia entre estructuras balanceadas y no balanceadas se hizo evidente en los tiempos y alturas obtenidos.