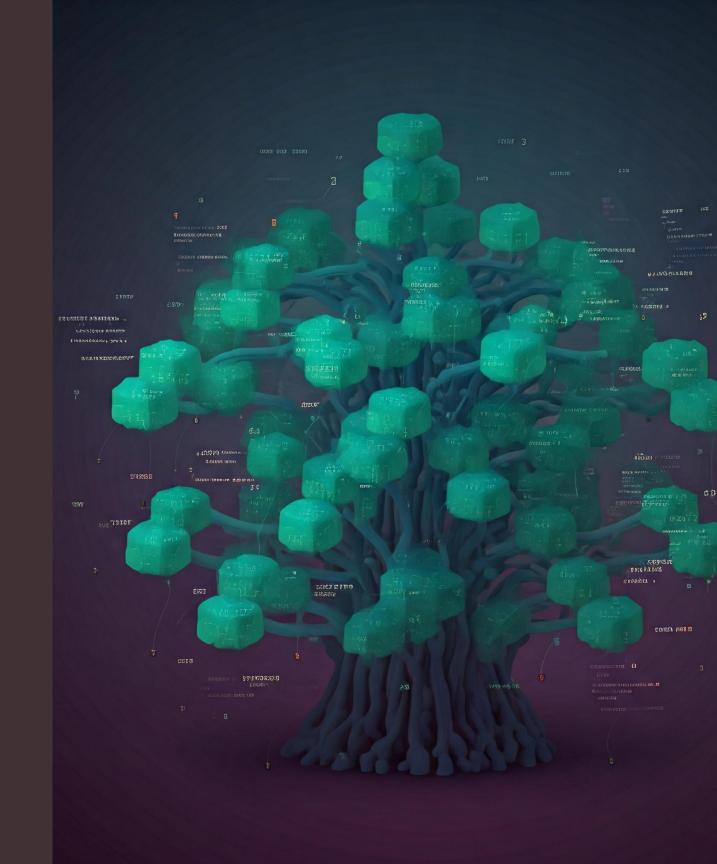
# Implementación de un Diccionario con Árboles Binarios de Búsqueda en Python

Alumnos: Matias Costantini y Ivan Daniliuk.

Materia: Programación 1.

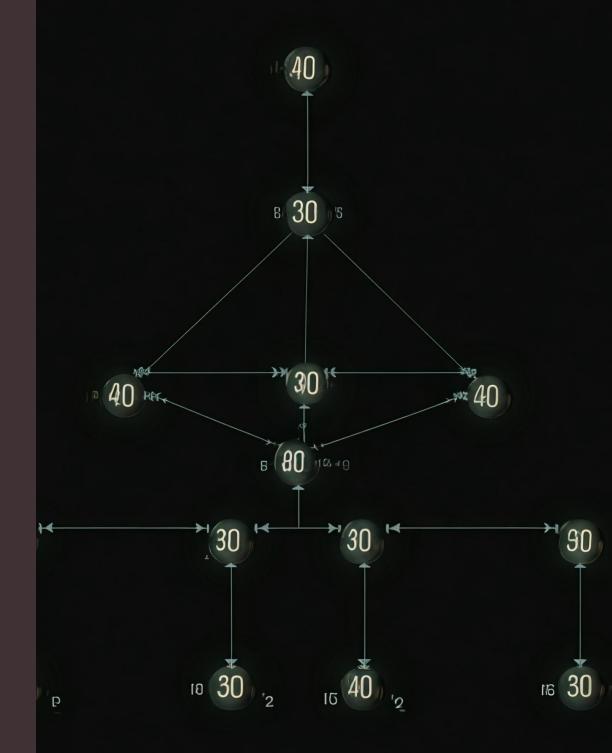
Comisión: 12.

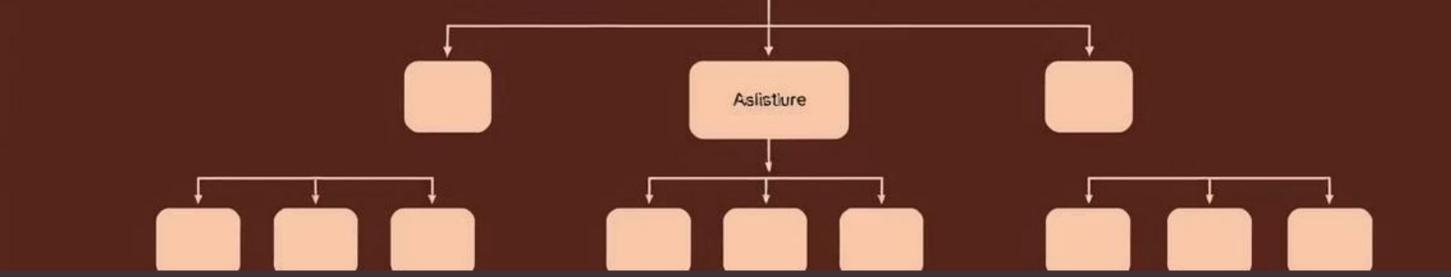


## Introducción

Este trabajo presenta la implementación de un diccionario digital utilizando Árboles Binarios de Búsqueda (ABB) en Python. Los ABB son estructuras de datos jerárquicas que permiten almacenar y buscar información de manera eficiente, con complejidad promedio O(log n).

El proyecto abarca desde la fundamentación teórica, diseño, implementación, hasta la validación práctica del diccionario, que permite agregar, buscar y listar palabras ordenadamente.





# Fundamentos y Tipos de Árboles

### **Árboles Binarios**

Estructura donde cada nodo tiene máximo dos hijos, izquierdo y derecho. La raíz es el nodo principal.

### Árboles AVL y B-Trees

Árboles auto-balanceables y optimizados para grandes volúmenes y acceso a disco, mejorando eficiencia.

### Árboles Binarios de Búsqueda (ABB)

Especialización con orden: claves menores a la izquierda y mayores a la derecha, facilitando búsquedas rápidas.

### Árboles Trie

Almacenan prefijos para búsquedas rápidas y autocompletado, útiles para diccionarios pero no implementados aquí.



# Operaciones Fundamentales en ABB

1 Inserción

Agregar nodos manteniendo la propiedad de orden del ABB, con complejidad promedio O(log n).

2 Búsqueda

Localizar palabras
eficientemente siguiendo la
estructura ordenada del
árbol.

3 Recorridos

Preorden, inorden y postorden para copiar, listar en orden alfabético o eliminar nodos.

# Diseño e Implementación del Diccionario en Python

### **Clases Nodo y Diccionario ABB**

El nodo almacena palabra, definición y referencias a hijos.

Diccionario ABB maneja inserción, búsqueda y listado.

### Decisiones de Diseño

- Palabras almacenadas en minúsculas para búsqueda insensible a mayúsculas.
- No se permiten duplicados ni actualización de definiciones.
- Uso de métodos recursivos para claridad y eficiencia.

### . → 〉 (3 )・ □ · Admst Auritasly Neek Servles Hjp Remadion Dictionary Moils Vritcome sectof(agory terrius: Luste; Reales; lodefelucion organing) III Legies II Prone censimal: Chalse: (UTILIS) # Recleas cardilinglater Meal) ontale = 132) ☐ Recipie Chies: lyrmate: Reskile tracile() onits: Tormtar() Grangs ontes: (owneney) \* rorle (lle-500) (lness - suri/lenctle) # Applings crise: (oratumloct00) wetts: lgriles fracilied) Morasties T001) Sortione cortus: =station for SOREI@natinest) Surings Virtranerts rase eroggnery Ohetat: Strop fry (Inection, decignspaily, tester pontertar) ► Sastonus elster hertendr Cefomaty taste: - leste: Packilve Noderlisadis tacls: Spotes aloum\_cortese Fregnac()); toule: tarile: (ew packfectoor; fratals; Trar Pasiges tants: sperie: Racially decolpatitor Replications shciatent compacte) wmlles, caste: igging scoked of antertor nerly) Replay Chtes: SostHel: - Fragrecion!)) न्नी Inagade Contactpolls: faviralef for emergnee Toetlg) Q Intsliem Poyarts Vilcane diccionary cmtes: "este: Sortall etuFC\_Partier\_CILTT; **M** Replions 8 Selong acheniater wetalls andicadls; q crrtes tertation) 3 Mioredun Aecivita ☑ Syntenics ▶ Appriore Sevil Virliel Serigeanet

# Validación y Pruebas del Diccionario

### 1 — Agregar Palabras

Se verifica la correcta inserción y confirmación de nuevas palabras y definiciones.

### P — Búsqueda

Pruebas con casos exactos, insensibles a mayúsculas y búsqueda de palabras inexistentes.

### B — Listado

Se comprueba que las palabras se muestran en orden alfabético mediante recorrido inorden.

### . ── Manejo de Errores

Validación de opciones inválidas y comportamiento con diccionario vacío.

# Resultados y Funcionamiento Correcto

### **Funcionalidades Principales**

- Inserción y almacenamiento consistente en minúsculas.
- Búsqueda eficiente y no sensible a mayúsculas.
- Listado ordenado alfabéticamente.

### Interfaz y Usabilidad

Menú interactivo en consola que guía al usuario y maneja entradas válidas e inválidas.

# Dificultades y Limitaciones Identificadas

### **Palabras Duplicadas**

No se actualizan definiciones ni se permiten duplicados, limitando flexibilidad.

### **Funcionalidades Avanzadas**

Faltan eliminación y modificación de palabras, fuera del alcance actual.

### Persistencia

Datos almacenados sólo en memoria, sin guardado en archivos.

### **Posible Desbalanceo**

ABB simple puede degradar rendimiento en casos adversos, sin auto-balanceo.



# Conclusiones y Futuras Mejoras

### Logros

Implementación funcional que consolida conocimientos sobre ABB y operaciones básicas.

### **Mejoras Futuras**

- Agregar actualización y eliminación de palabras.
- Implementar persistencia de datos en archivos.
- Considerar árboles auto-balanceables para mayor eficiencia.
- Desarrollar interfaz gráfica para mejor experiencia de usuario.