Estructura de datos para búsqueda de directories y ficheros

Andrés Sánchez Castrillón
Santiago Soto Marulanda
Jamerson Stive Correa Correa

Medellín
30/10/2017

Estructura de Datos Diseñada

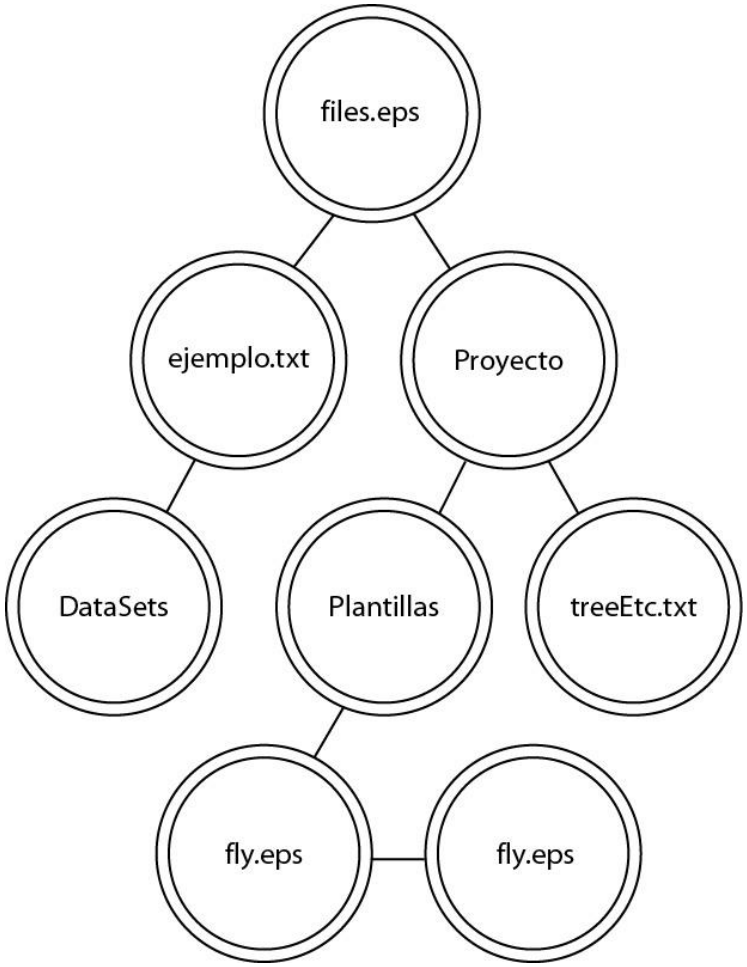


Gráfico 1: Árbol AVL de archivos. Un archivo es una clase que contiene nombre y dirección.

Operaciones de la Estructura de Datos

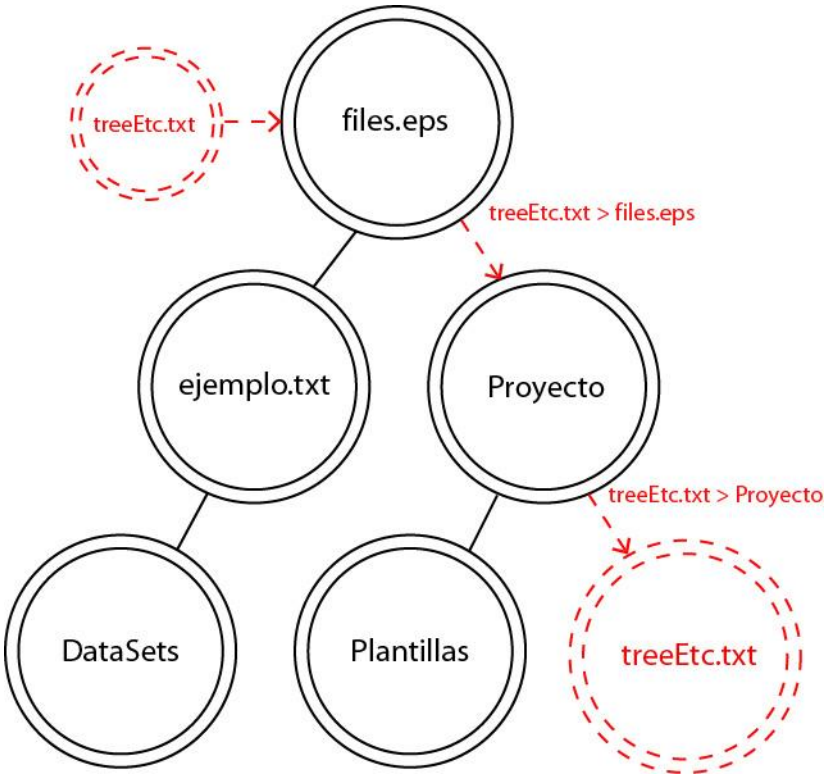


Gráfico 2: Operación insert
de la estructura de datos implementada.

Búsqueda y inserción comparando el nombre de los archivos, se organizan lexicográficamente

Funcion	Complejidad
insert	O(log n)
search	O(log n)

Tabla 1. Complejidad de las operaciones de la estructura de datos 2.

Operaciones de la Estructura de Datos

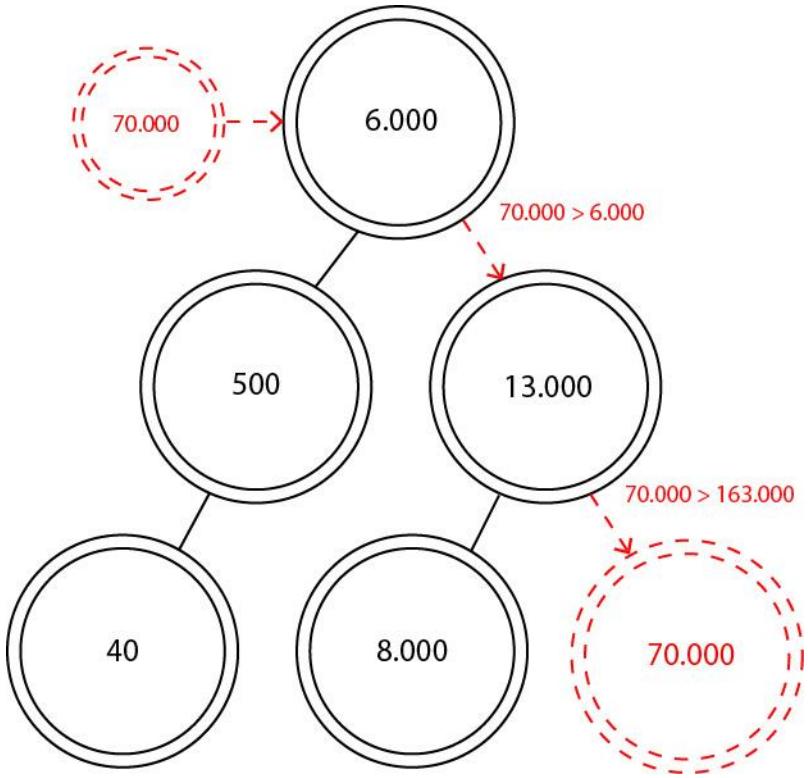


Gráfico 3: Operación insert
de la estructura respect tamaño (No implementada).

Búsqueda y inserción comparando el tamaño de los archivos

Funcion	Complejidad
insert	O(log n)
search	O(log n)

Tabla 2. Complejidad de las operaciones de la estructura de datos 2.

Operaciones de la Estructura de Datos

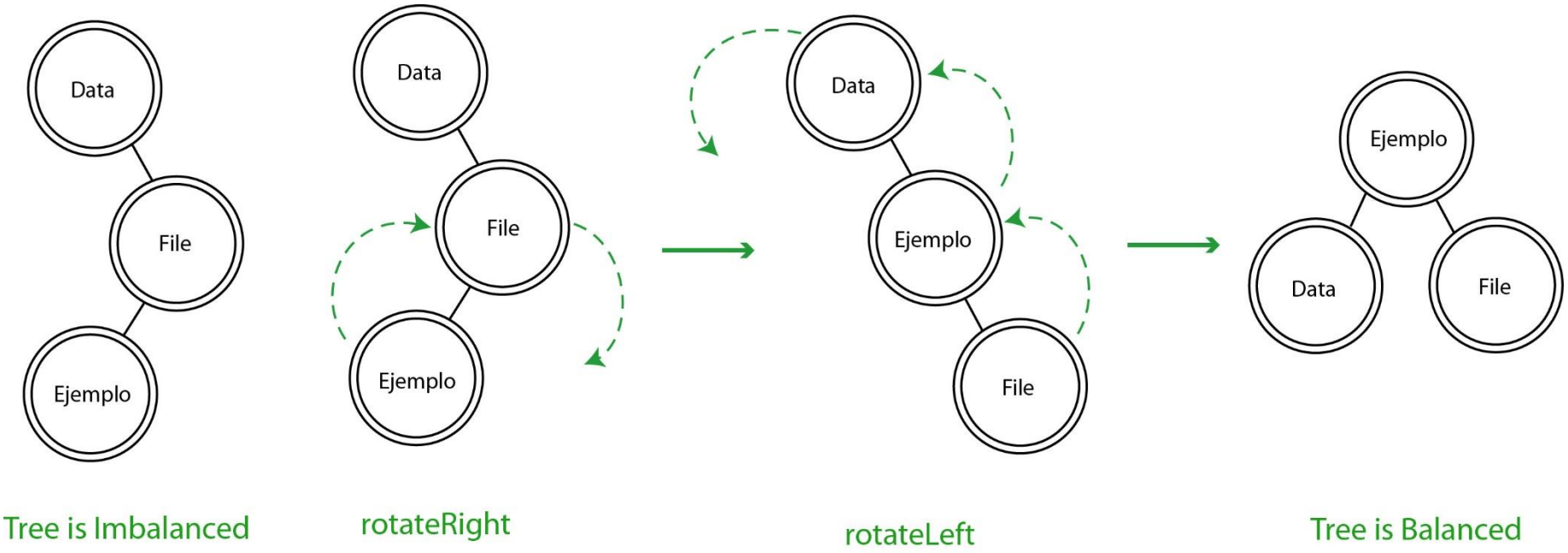


Gráfico 4 . Imagen de las operaciones de rotación en el árbol AVL.

Criterios de Diseño de la Estructura de Datos

- Los árboles AVL son árboles de búsqueda binarios con excelentes garantías de rendimiento gracias a la restricción en la ubicación de cada nodo, lo que permite un árbol en todo momento equilibrado y balanceado respecto a su orden lexicográfico lo que permite lograr un tiempo de ejecución correspondiente a la complejidad O(log(n)).

Árbol AVL VS Árbol Rojo-Negro

Para datos pequeños:

- **Insertar** : El árbol RN y el árbol AVL tienen un número constante de rotación máxima, pero el árbol RB será más rápido porque, en promedio, el árbol RB usa menos rotación.
- **Búsqueda** : El árbol AVL es más rápido, porque tiene menos profundidad.

Para datos grandes:

- **Insertar** : El árbol AVL es más rápido. porque necesita buscar un nodo particular antes de la inserción. a medida que tenga más datos, la diferencia de tiempo al buscar el nodo particular crece proporcionalmente
- **Búsqueda** : El árbol AVL es más rápido, porque tiene menos profundidad.

Árbol AVL VS Tabla hash

Árbol AVL

Toda la memoria se puede asignar dinámicamente, por lo que la cantidad de datos no debe ser conocido de antemano.

No hay colisiones, por lo que se garantiza la complejidad de búsqueda O(log n)

No es necesario conocer el conjunto de datos.

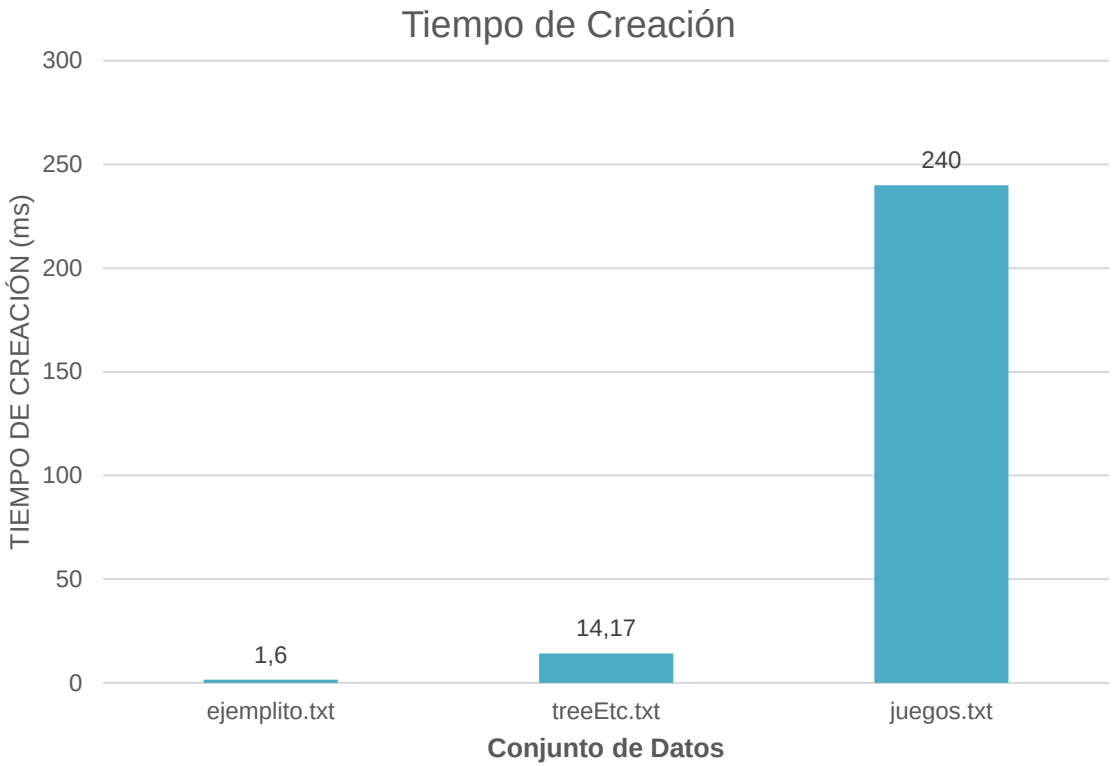
Tabla hash

Los datos en la memoria NO se asignan dinámicamente, por lo que la cantidad de datos debe ser conocido.

Si hay colisiones, por lo que la complejidad en el peor de los caso es O (n/k)

Una buena función hash requiere conocimiento sobre los datos, y se debe adaptar preferiblemente a cada conjunto de datos.

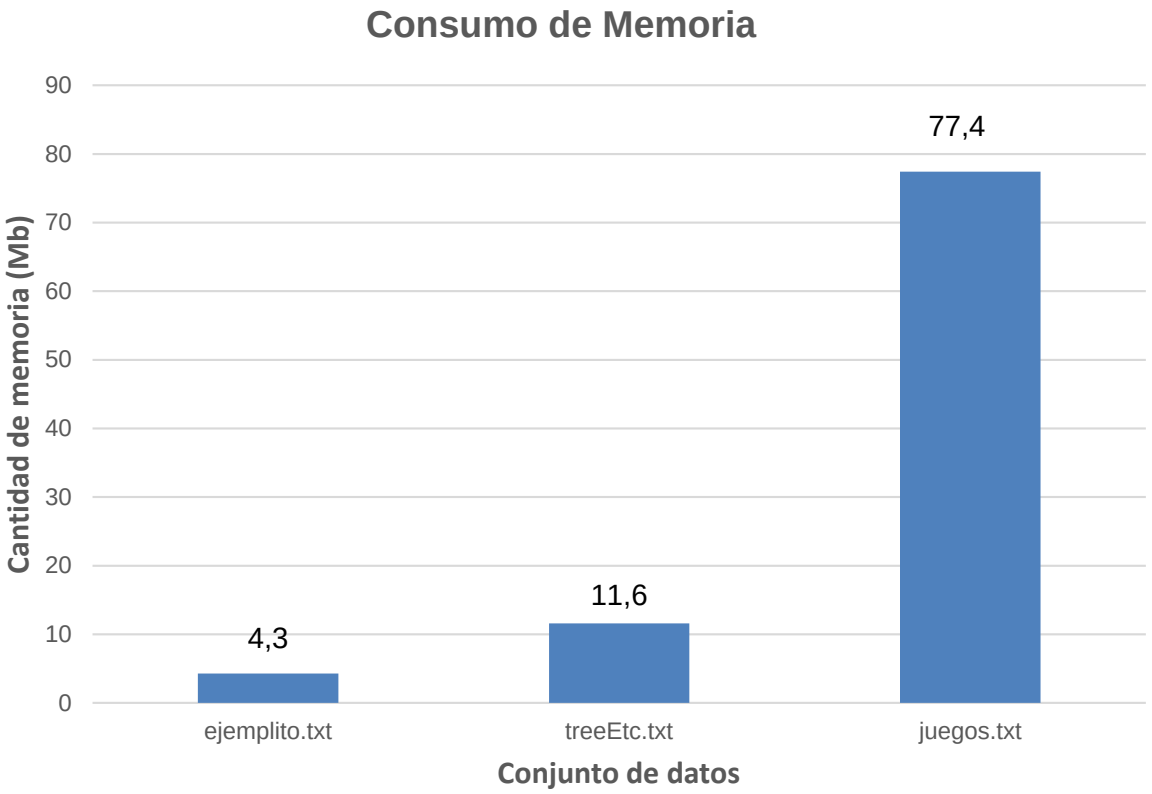
Consumo de Tiempo y Memoria



Función	Creación (ms)
ejemplito.txt	1,6
treeEtc.txt	14,1
juegos.txt	0,24

Grafico 5: Calculo del tiempo de ejecución del conjunto de datos proporcionados, usando la Estructura de datos implementada.

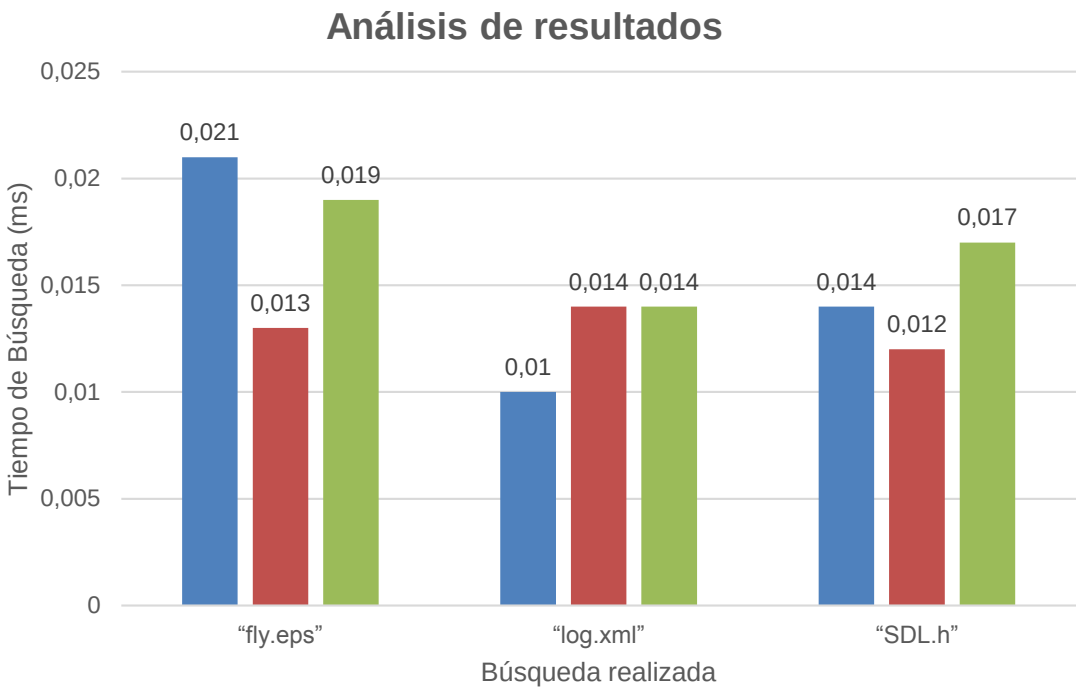
Consumo de Tiempo y Memoria



Estructura	Memoria(Mb)
ejemplito.txt	4,3
treeEtc.txt	11,6
juegos.txt	77,4

Grafico 6: Calculo del consumo de memoria del conjunto de datos proporcionado, usando la Estructura de datos implementada.

Análisis de los resultados



Búsqueda	ejemplito.txt t(ms)	treeEtc.txt t(ms)	juegos.txt t(ms)
"fly.eps"	0,021	0,013	0,019
"log.xml"	0,01	0,014	0,014
"SDL.h"	0,014	0,012	0,017

Conjuntos de datos:
ejemplito.txt **treeEtc.txt** **juegos.txt**

Grafico 7: Tiempos de búsqueda de la estructura de datos 2 con diferentes conjuntos de datos.