

Data Structure for efficient indexing of files

Mateo Restrepo Sierra

Nicolas Restrepo López

Medellín, October 30th, 2017

Data Structure design

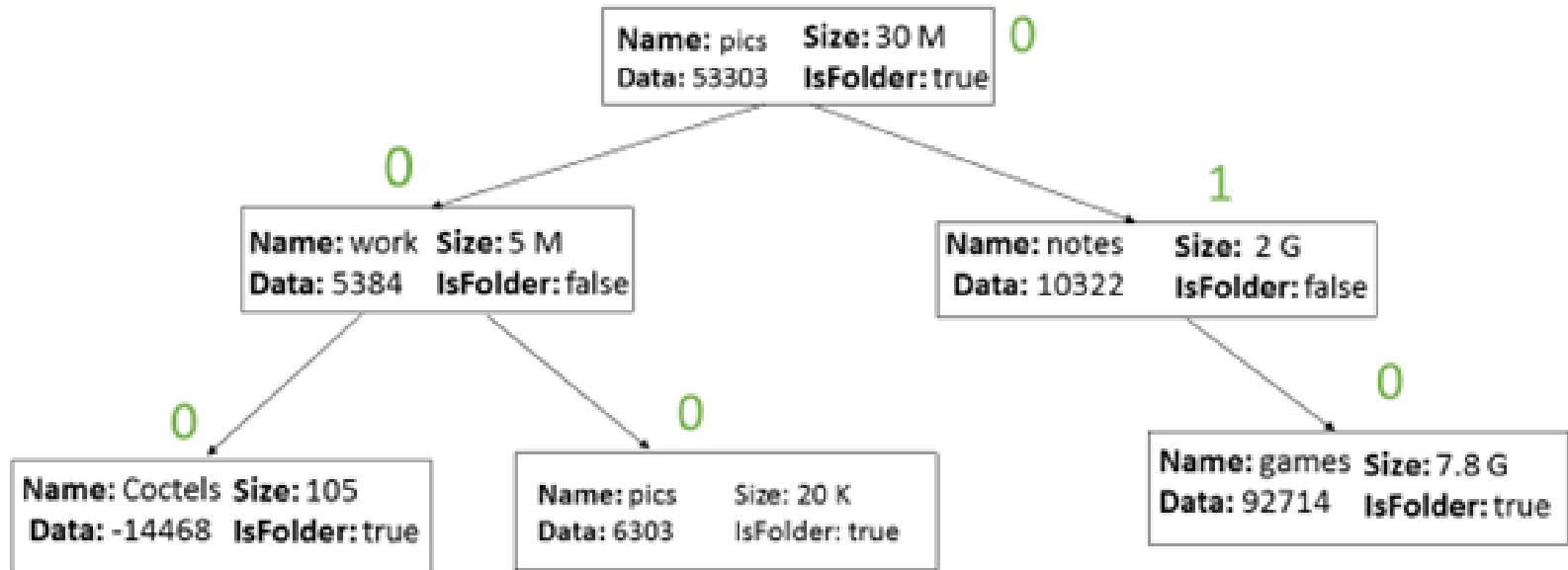


Figure 1: This is an early implementation of our AVL Tree. The numbers above the tree are the balanced factor. It does not have all the fields that the objects have.

Data Structure Operations

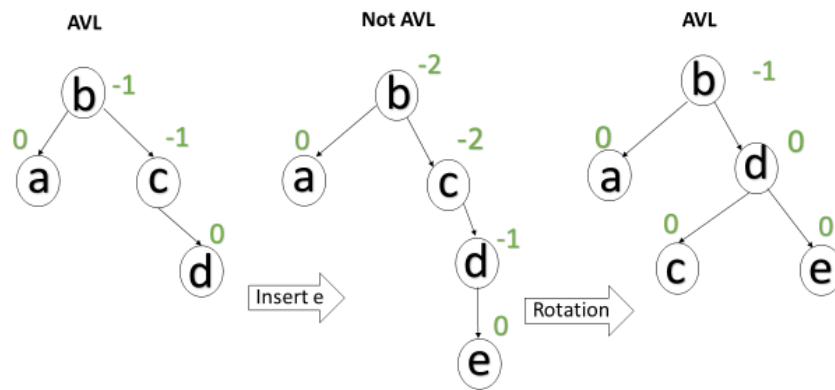


Figure 2: Image of an operation of insertion.

Method	Complexity
Insert a Node	$O(\log n)$
Delete a Node	$O(\log n)$
Search an Element	$O(\log n)$
Print Pre-Order	$O(n)$
Print In Order	$O(n)$
Print Post Order	$O(n)$

Figure 3: Complexity analysis for AVL Tree standard operations.

Design Criteria of the Data Structure

- To solve the problem, we should use, extensively, the operation of searching
- Trees are the most ideal to solve this problem
- Searching has a time complexity of $O(\log n)$
- The AVL Tree is one of the most efficient Data Structures, because it has shorter height in the subtrees than other Self-Balancing Trees like the Red-Black Tree
- AVL Tree provides faster look-ups
- Hash functions were used for insertion

Time and Memory Consumption

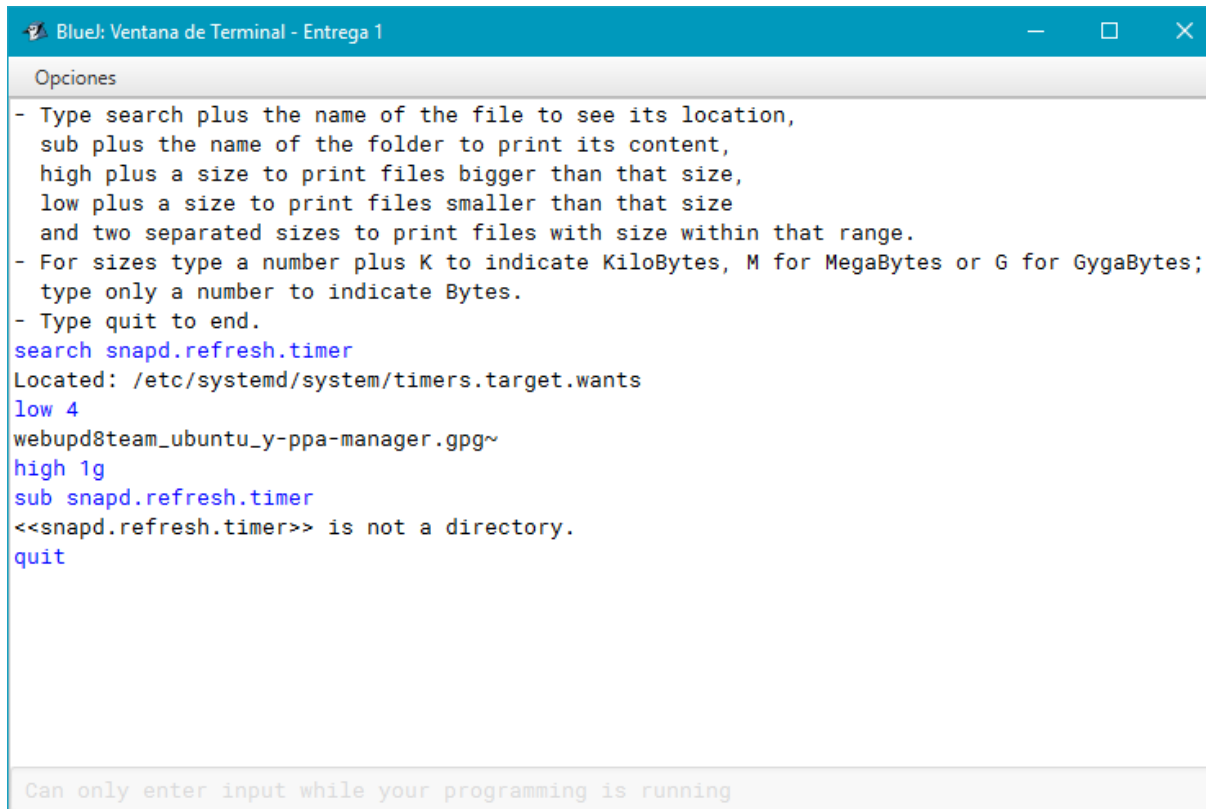
Running Times (Milliseconds)			
	ejemplito.txt	treeEtc.txt	juegos.txt
Insert	15	184	3236
Search Directory	~0	~0	~0
Search Sub directories	~0	~0	335,7
Search Low	1215	4343	15440
Search High	~0	~0	~0
Search Range	~0	~0	1117

Figure 4: Execution time in milliseconds by three different .txt files.

Memory Used (Megabytes)			
	ejemplito.txt	treeEtc.txt	juegos.txt
First Try	9,566	13,750	48,253
Second Try	9,565	13,749	47,595
Third Try	9,565	13,750	52,018

Figure 5: Memory use in Megabytes by three different .txt files.

Implementation



The screenshot shows a BlueJ terminal window titled "BlueJ: Ventana de Terminal - Entrega 1". The window contains a list of instructions for the program, followed by several commands entered by the user. The program's output shows the location of a file and the result of a search for files larger than 1GB.

```
Opciones
- Type search plus the name of the file to see its location,
  sub plus the name of the folder to print its content,
  high plus a size to print files bigger than that size,
  low plus a size to print files smaller than that size
  and two separated sizes to print files with size within that range.
- For sizes type a number plus K to indicate KiloBytes, M for MegaBytes or G for GygaBytes;
  type only a number to indicate Bytes.
- Type quit to end.
search snapd.refresh.timer
Located: /etc/systemd/system/timers.target.wants
low 4
webupd8team_ubuntu_y-ppa-manager.gpg~
high 1g
sub snapd.refresh.timer
<<snapd.refresh.timer>> is not a directory.
quit

Can only enter input while your programming is running
```

Figure 6: Our implementation running in BlueJ for treeEtc.txt.