

## Laboratorio Nro. 4: Arboles

**Santiago Pulgarin Vasquez**

Universidad Eafit  
Medellín, Colombia  
spulgarinv@eafit.edu.co

**Juan Camilo Guerrero Alarcon**

Universidad Eafit  
Medellín, Colombia  
jcguerrera@eafit.edu.co

### 3) Simulacro de preguntas de sustentación de Proyectos

1. Concretamente si se puede implementar un mecanismo o mejor dicho una estructura de datos más eficiente que la que veníamos usando denominada Binar tree, en casos en los que no se requiere un mayor esfuerzo trabaja muy bien y sin problemas, trabajando incluso sobre tiempos excelente, pero en el peor de los casos o con datos más complejos no nos ayuda demasiado ya que actuaría un poco más lento que una tabla de hash. Sin embargo, para que nuestro árbol genealógico funcione aún más rápido existe una alternativa la cual es el AVL tree, uno de los primeros arboles implementados, y que ofrece una velocidad y tiempos óptimos a grandes cantidades de datos, sin ningún problema podemos realizar nuestro árbol genealógico en tiempos de  $O(n \log n)$ , y se ajusta a nuestro ejercicio ya que principalmente trabaja con operaciones básicas.
2. El funcionamiento de este programa se basa en que inicialmente se introduce un árbol binario de búsqueda en Pre-Orden y debe imprimirlo en Pos-Orden, ¿Como se logra esto? Primero que todo hay que almacenar en un árbol binario de búsqueda (Pre-Orden) y luego mediante el método ImprimirEnPosOrden() se imprime en Pos-Orden sabiendo que el primer dato introducido (root) en el Pre-Orden será el último en el Pos-Orden.
3.  $O(n)$
4. En este caso la variable que existe es Node el cual indica la cantidad de nodos que tiene el árbol y dependiendo de dicha cantidad el árbol tiene “n” complejidad.

### 4) Simulacro de Parcial

1. altura(raiz.izq)  
altura(raiz.der)
2. C
3. false  
a.dato  
a.izq, suma – a.dato  
a.der, suma – a.dato
4. C  
A  
D  
A
5. p.dato == toInsert

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

- toInsert > p.dato
- 6. D  
Return 0  
== 0
  - 7. 1  
2
  - 8. B
  - 9. A
  - 10. B