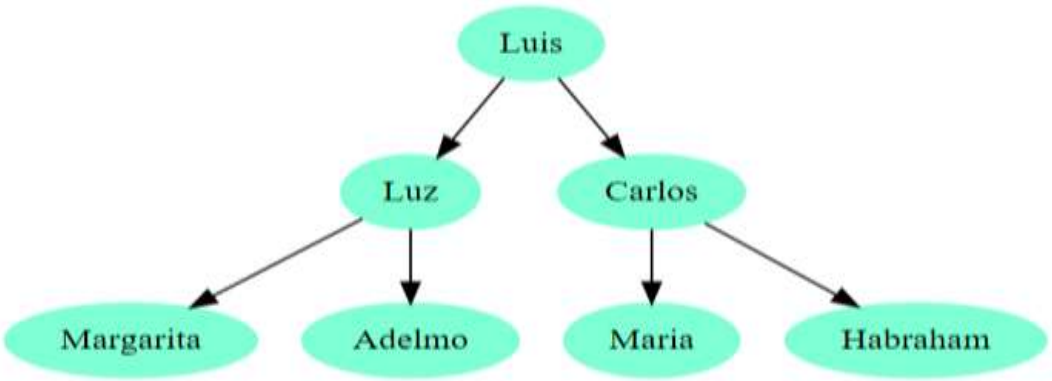
Laboratorio Nro. 5: Árboles binarios

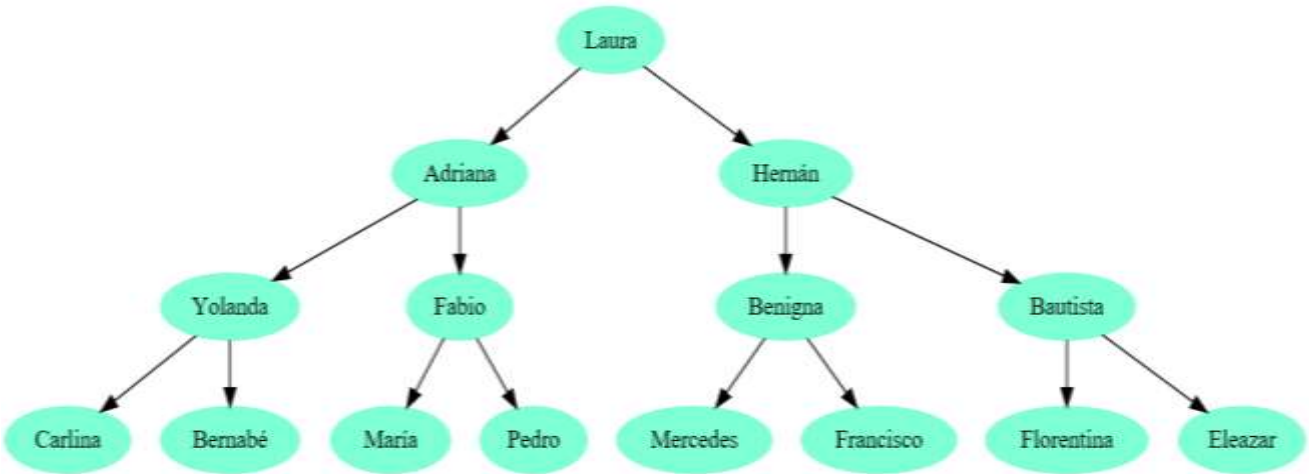
Luis Carlos Rodríguez Zúñiga
Universidad Eafit
Medellín, Colombia
lcrodriguz@eafit.edu.co

Laura Sánchez Córdoba
Universidad Eafit
Medellín, Colombia
lsanchezc@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1.





DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

2. En este caso no se podría, pues no hay una forma directa de comparar todos los datos para que estos métodos se lleven a cabo de manera más eficiente. Supóngase por ejemplo si nos guiáramos solamente por el género (mujeres izquierda, hombres derecha) solo agregaría bien los primeros (raíz del árbol padre y madre), pero si se quiere agregar; por ejemplo, el abuelo materno, al ser hombre iría directamente a la derecha del padre y análogamente sucedería con buscar. Por lo que se haría necesario que al método se le tenga que dar un nodo raíz del cual partir, lo que reduce su eficiencia.

3. El algoritmo funciona de la siguiente manera: el usuario entra cierta cantidad de datos mediante la consola y debe escribir “listo” para indicar que ya entró todos los datos que quería. Mientras no se escriba la palabra de control, cada dato se convierte s tipo int y pasa a ser evaluado por el método insertar que a su vez llama al método esta(), el cual verifica que un valor no se halla ya en el árbol, si no está se pasa a añadir dicho valor al árbol comparándolo con los valores de los nodos existentes, sí está vacío se pone como raíz, si es menor a la izquierda y si es mayor a la derecha. Una vez armado el árbol se procede a llamar el método recursivePrint(), el cual tiene un auxiliar que recibe la raíz del árbol y lo recorre imprimiendo los datos de la forma: izquierda, derecha, raíz, es decir, pos-orden

4. **//inserta los valores que vayan entrando al árbol**

```
public void insertar(int n){
   if(!esta(n)){ //T(n)
      Node nuevo = new Node(n); //c1
      if(root==null){ //c2
         root = nuevo; //c3
         return; //c4
      }

      Node temp = root; //c5
      Node padre = null; //c6

      while(true){ //c7*n
         padre = temp; //c8
         if(n < temp.data){ //c9
            temp = temp.left; //c10
            if(temp==null){ //c11
               padre.left= nuevo; //c12
```

```
                return; //c13
            }
        }else{
            temp= temp.right; //c14
            if(temp==null){ //c15
                padre.right=nuevo; //c16
                return; //c17
            }
        }
    }

    }else{
        System.out.println(" el data ya se encuentra"); //c18
    }
  }
```

T(n)=c+n+c1*n
O(n)=c+n+c1*n
O(n)=c+n+n R.P.
O(n)=n+n R.S
O(n)=2n
O(n)=n R.P.
O(n)

//Verifica si un valor se encuentra en el árbol

```
  private boolean esta(int n){
    Node temp = root; //c1
    while(temp!=null){ //c2*n
        if(temp.data == n){ //c3
            return true; //c4
        }else if(temp.data > n){ //c5
            temp = temp.left; //c6
        }else{
            temp = temp.right; //c7
        }
    }
    return false; //c8
  }
```

T(n)=c+ c2*n
O(n)=c+ c2*n
O(n)=c2*n R.S.
O(n)=n R.P.
O(n)

```
//Para imprimir en posorden
    private void recursivePrintAUX(Node node)
    {
        if (node != null) //c1
        {
            recursivePrintAUX(node.left); //T(n/2)
            recursivePrintAUX(node.right); //T(n/2)
            System.out.println(node.data);//c2
        }

    }

    public void recursivePrint()
    {
        recursivePrintAUX(root); //c1*n
    }
```

T (n)= 2*T(n/2)+c
T(n)=c(n-1)+c1*n/2
O(n)=c(n-1)+c1*n/2
O(n)=c1*n/2 R.S
O(n)=n/2 R.P
O(n)=n R.P
O(n)

```
public static void main(String[] args) throws IOException{
        //Crea el árbol
        BinaryTree tree = new BinaryTree(); //c1
        String entrada=""; //c2
        //ingreso datos
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in)); //c3
        System.out.println("Por favor ingrese los datos para crear el arbol"); //c4
        System.out.println("Si ya finalizó, por favor escriba listo"); //c5

        while(entrada!="listo"){  //c6
         entrada = br.readLine();//c7
         if(entrada.equals("listo")) break; //c8
         int aux= 0;//c9
         aux= Integer.parseInt(entrada); //c10
```

```
        tree.insertar(aux); //T(n)
       }
       //Imprime el árbol en pos-orden
       tree.recursivePrint(); //T(n)
T(n)=c+ 2T(n)
T(n)=c+ 2*n
O(n)=c+ 2*n
O(n)=2*n R.S
O(n)=n R.P
O(n)
```

5. La variable n usada en la complejidad hace referencia a la cantidad de datos que el usuario entra (en este caso, números) de la cual dependerá el número de pasos que debe realizar el programa para emitir una respuesta y el tiempo que tomará hacerlo.

4) Simulacro de Parcial

1. *a) 1+altura(raiz.izq); b) 1+altura(raiz.der);*
2. *c) 3*
3. *a) false*
   b) a.dato
   c) a.izq, suma-a.dato
   d) a.der, suma-a.dato
4. *4.1. c*
   4.2. a
   4.3. d
   4.4. a
5. *a)* p.dato == toInsert
   b) toInsert > p.dato

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co