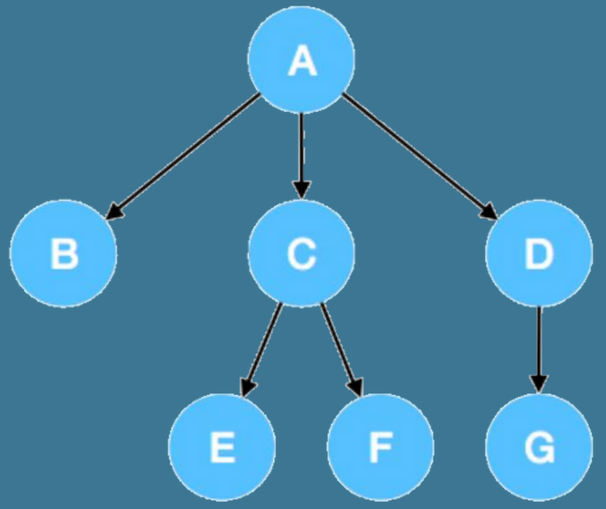
INSTITUTO TECNOLÓGICO DE COSTA RICA



N -ARY TREE

Isac Marí Sirias

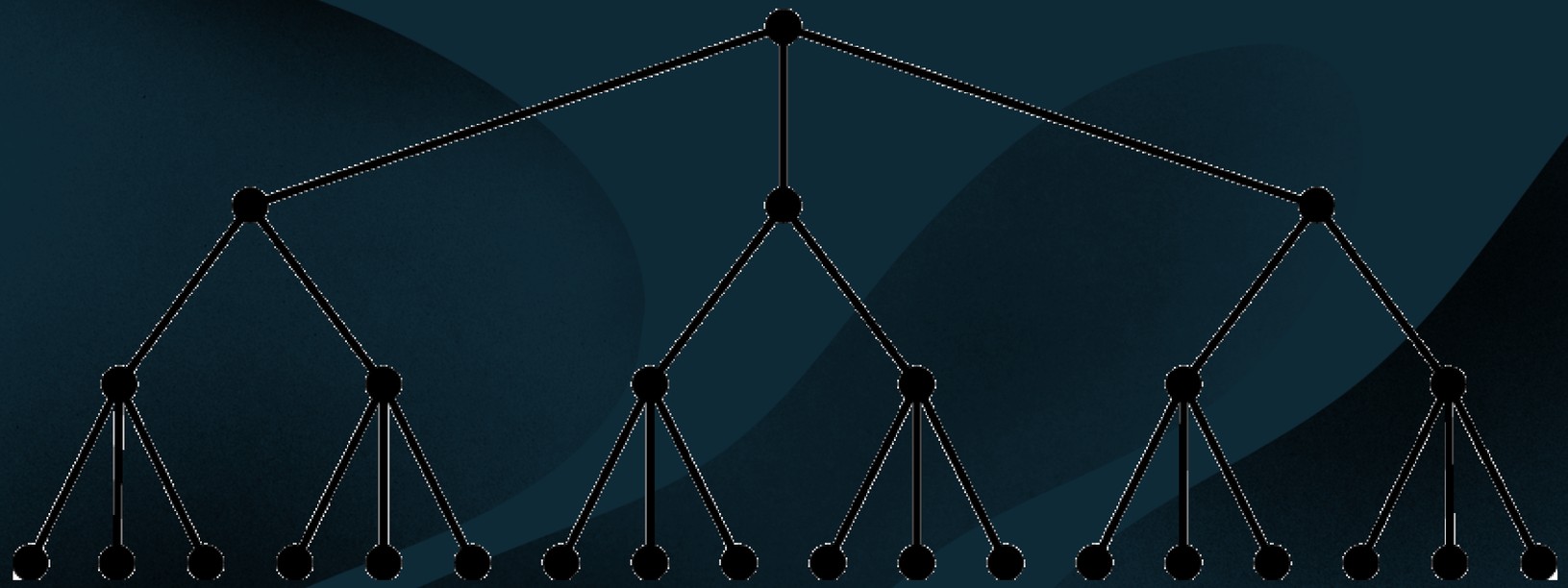
Valerin Calderón

Alisson Redondo
Andrés Rayo
Ludwin Ramos

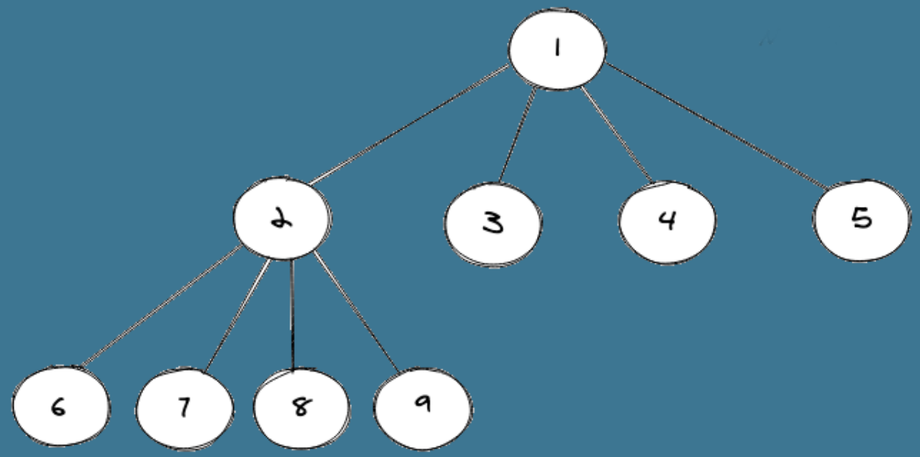
Fecha: San José, Costa Rica, octubre, 20, 2021

Prof. Antonio Gonzáles

¿QUÉ ES UN N-ARY TREE?

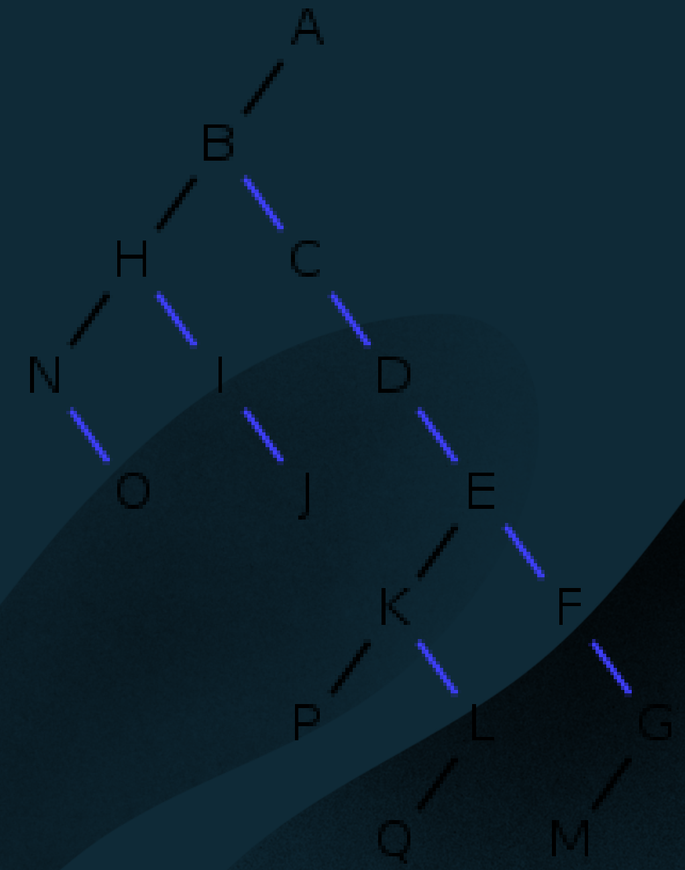


N-ARY TREE

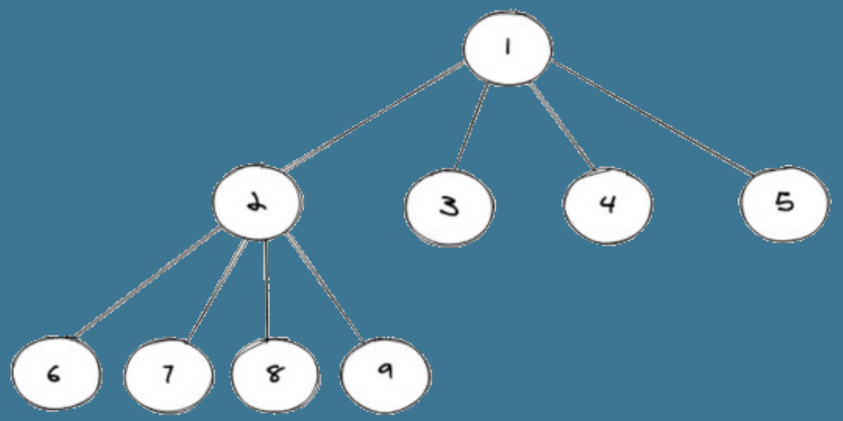


ES UN ÁRBOL QUE NOS PERMITE TENER UN "N" NÚMERO DE HIJOS DE UN NODO EN PARTICULAR, DE AHÍ EL NOMBRE N-ARIO , LO QUE LO HACE UN POCO MÁS COMPLEJO QUE LOS ÁRBOLES BINARIOS MUY COMUNES QUE NOS PERMITEN TENER COMO MÁXIMO 2 HIJOS DE UN DETERMINADO NODO.

TIPOS DE N-ARY TREE

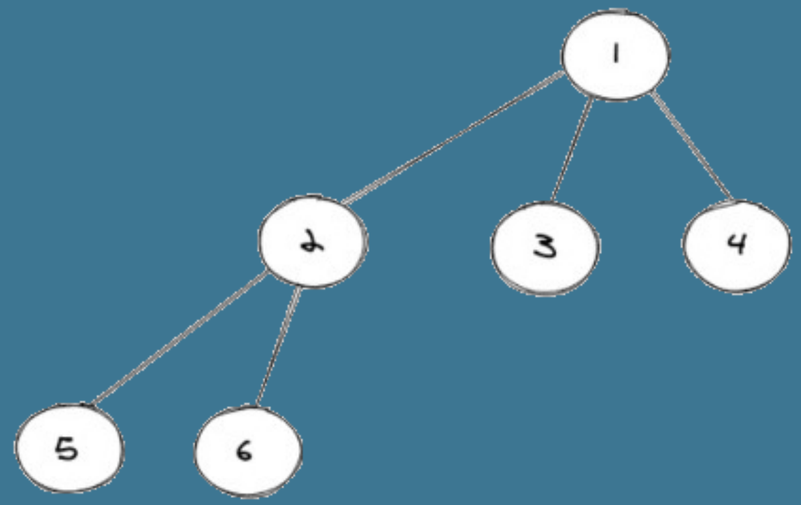


I. FULL N-ARY TREE



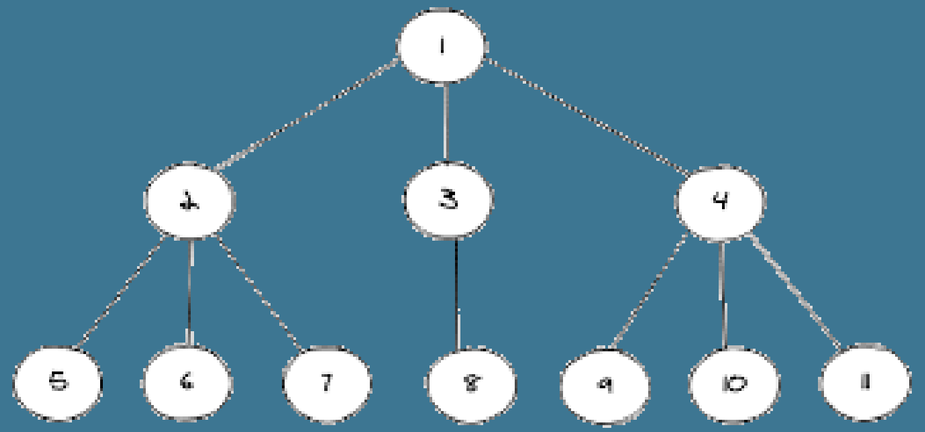
Un árbol Full N-ario es un árbol N-ario que permite que cada nodo tenga 0 o N hijos.

II. COMPLETE N-ARY TREE



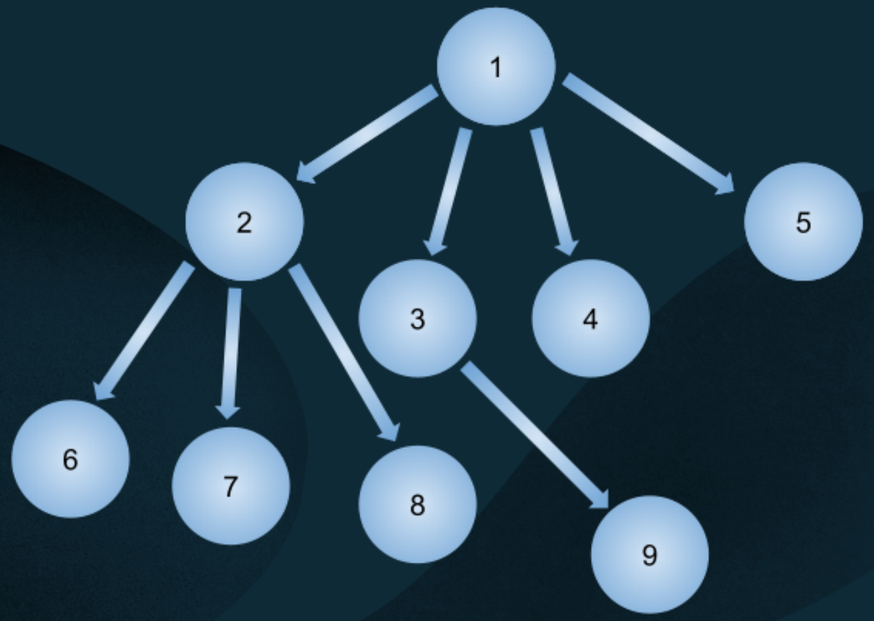
Un árbol N-ario completo es un árbol N-ario en el que los nodos en cada nivel del árbol deben estar completos (deben tener exactamente N hijos) excepto los nodos del último nivel y si los nodos del último nivel no están completos, los nodos debe estar "lo más a la izquierda posible".

III. PERFECT N-ARY TREE



Un árbol N-ario perfecto es un árbol N-ario completo, pero el nivel de los nodos de las hojas debe ser el mismo

IMPLEMENTACIÓN



N-ARY NODE

Como en el caso de un árbol N ario, hacemos uso de una clase N-aryNode y dentro de esa clase, creamos nuestros constructores y tenemos nuestras variables de nivel de clase.

```java
public class NaryNode{

    int cantidadH;
    String dato;
    ArrayList <NaryNode> hijos;

    public NaryNode(String dato){
        hijos = new ArrayList<NaryNode>();
        this.dato = dato;
    }

    public void AumentarHijo (NaryNode nodo){
        hijos.add(nodo);
        cantidadH =hijos.size();
    }

    public void ActualizarNodosHijos(){
        cantidadH = hijos.size();
    }

    public void verInfo(){
        System.out.println("{ "+dato+" }");
    }

    public void Verhijos(){
        System.out.println(cantidadH);
    }

    public void setDato(String dato){
        this.dato = dato;
    }

    public String getDato(){
        return dato;
    }

    public int getHijos(){
        return cantidadH;
    }

    public void RestarHijos(){
        cantidadH--;
    }

    public NaryNode retornarNodo(){
        return this;
    }

}
```

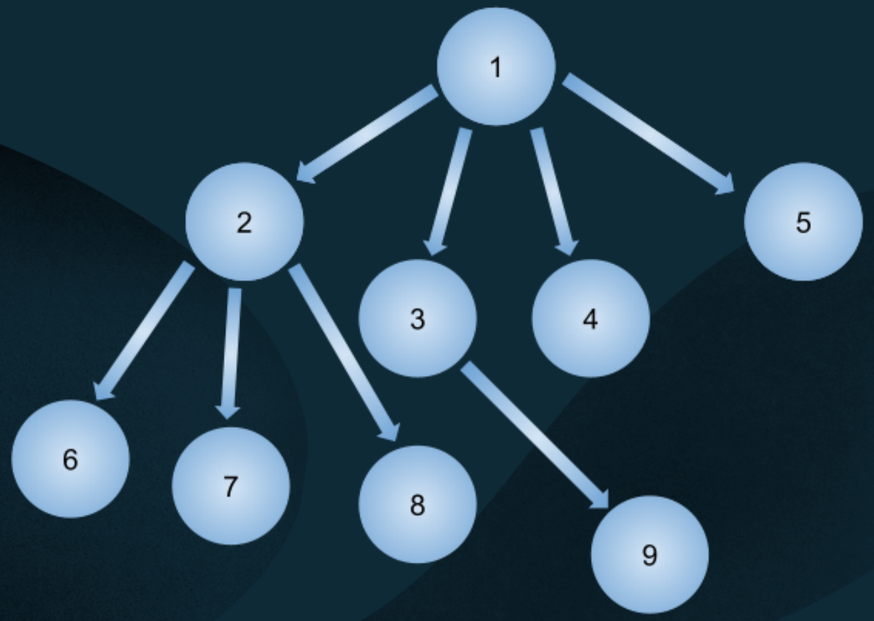
N-ARY TREE

A la hora de implementar un N-aryTree, se debe de crear una clase la cual tenga la estructura del árbol, además de esto, dicha clase debe de tener métodos; entre los mas importantes están los de insertar raíz, insertar nodos, recorrer y buscar.

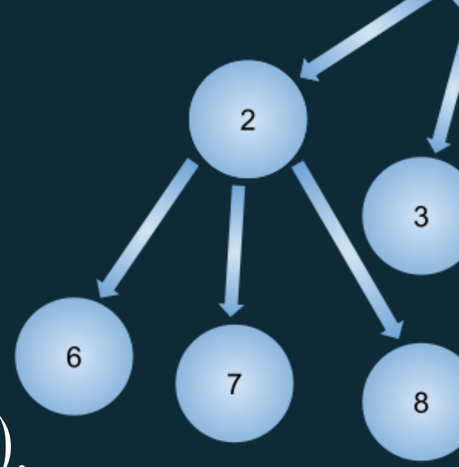
```java
public class NaryTree {

    NaryNode raiz;

    public NaryTree(){}

    public NaryNode insertroot(String dato){

        raiz = new NaryNode(dato);
        return raiz;

    }

    public void Insert (NaryNode nodo, String dato, String Padre){
        NaryNode nuevo = new NaryNode(dato);

        //Si el padre es la raiz
        if(nodo.getDato().equals(Padre)){
            nodo.AumentarHijo(nuevo);
        }else{
            //Busca entre los hijos que tiene el padre
            for(int i = 0; i<nodo.getHijos(); i++){
                if(nodo.hijos.get(i).getDato().equals(Padre)){

                    //Se coloca el hijo en el nodo
                    nodo.hijos.get(i).AumentarHijo(nuevo);
                }else{
                    Insert(nodo.hijos.get(i), dato, Padre);
                }
            }
        }
    }
```

```java
public void recorreer(NaryNode raiz){
    raiz.verInfo();
    for(int i = 0; i < raiz.getHijos(); i++){
        recorreer(raiz.hijos.get(i));
    }
}

public void recorreerHijosRaiz(NaryNode raiz){
    for(int i = 0; i < raiz.getHijos(); i++){
        recorreerHijosRaiz(raiz.hijos.get(i));
    }
    raiz.verInfo();
}


public boolean Buscar(NaryNode raiz, String buscar, boolean Encontrado){
    if(raiz.getDato().equals(buscar)){
        Encontrado = true;
    }
    for (int i = 0; i < raiz.getHijos(); i++){
        Encontrado = Buscar(raiz.hijos.get(i), buscar, Encontrado);
    }
    return Encontrado;

}
```

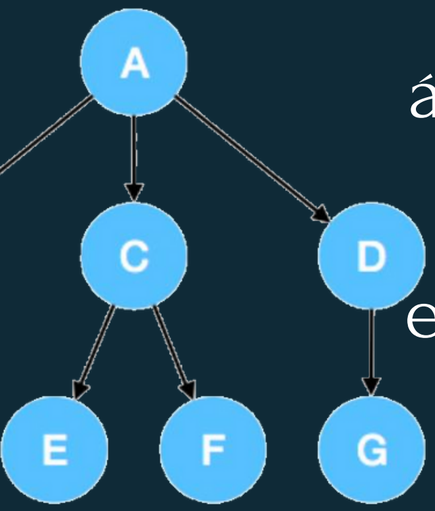
CASO DE PRUEBA



```java
public static void main(String[] args) {
    NaryTree objTree = new NaryTree();
    NaryNode raiz = objTree.insertroot("0");

    objTree.Insert(raiz, "1", "0");
    objTree.Insert(raiz, "2", "0");
    objTree.Insert(raiz, "3", "0");
    objTree.Insert(raiz, "4", "0");

    objTree.Insert(raiz, "1.1", "1");
    objTree.Insert(raiz, "2.1", "2");
    objTree.Insert(raiz, "3.1", "3");

    objTree.Insert(raiz, "3.1.1", "3.1");
    objTree.Insert(raiz, "3.1.1.4", "3.1.1");
    objTree.Insert(raiz, "1.1.1", "1.1");
    objTree.Insert(raiz, "1.1.1.2", "1.1.1");

    objTree.recorreer(raiz);
    System.out.println("Buscar  " + objTree.Buscar(raiz, "2", false));
    System.out.println("Altura  " + objTree.altura(raiz));
    System.out.println("Altura2  " + objTree.altura2(raiz));
    System.out.println("Altura3  " + objTree.altura3(raiz, 0));

    System.out.println("hojas  " + objTree.numeroHojas(raiz));
    System.out.println("Nivel de elemento  " + objTree.nivelElemento(raiz, "1", 0));
    System.out.println("Nivel de elemento  " + objTree.nivelElemento(raiz, "1.1", 0));
    System.out.println("Nivel de elemento  " + objTree.nivelElemento(raiz, "1.1.1", 0));
    System.out.println("Nivel de elemento  " + objTree.nivelElemento(raiz, "1.1.1.2", 0));
```



En la siguiente implementación se inserta una raíz, después de esto se insertan los nodos hijos de esa raíz y posteriormente los hijos de los hijos (las hojas). Además en el código presente, se implementaron métodos secundarios, los cuales recorren el árbol por la raíz, búsqueda de nodos, busca altura 1, 2 y 3, sin contar que también cuneta el numero de hojas que tienen el árbol y el nivel del elemento que se quiere buscar, esto con la finalidad de ejemplificar de mejor manera la estructura del N-ary Tree y su correcto funcionamiento.



```
{ 0 }
{ 1 }
{ 1.1 }
{ 1.1.1 }
{ 1.1.1.2 }
{ 2 }
{ 2.1 }
{ 3 }
{ 3.1 }
{ 3.1.1 }
{ 3.1.1.4 }
{ 4 }
Buscar  true
Altura  5
Altura2  8
Altura3  8
hojas  4
Nivel de elemento  -1
Nivel de elemento  2
Nivel de elemento  3
Nivel de elemento  4
PS D:\II Semestre 2021\Algoritmos y Estructuras de Datos I\N-ary Tree>
```