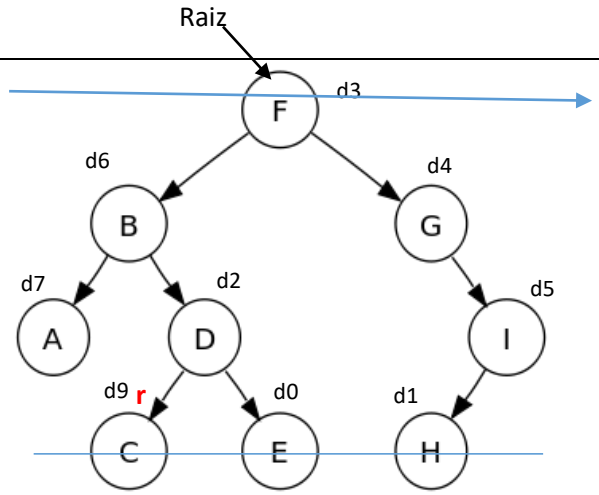
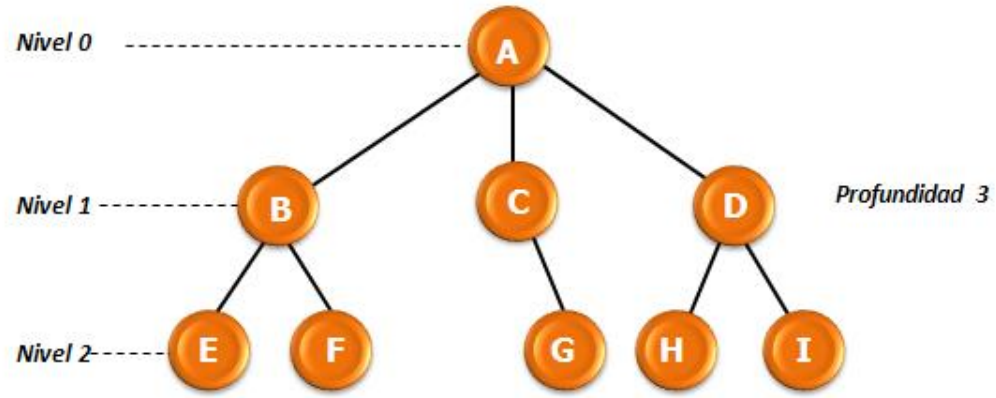
ARBOLES BINARIOS
RECORRIDOS EN AMPLITUD



Estructuras Auxiliares (pilas/colas)
r= d9
Cola **Nivel**

Cola **Descendientes**

Arbol Binario Completo
{0,2}

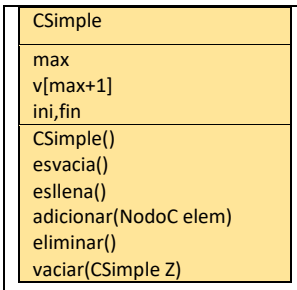


EJEMPLO: Sea un Arbol Binario de contactos (nombre, contacto)

NodoC
nombre contacto izq der
NodoP() Getters setters

ABinarioC
Raíz
ABinario() getRaiz setRaiz crear niveles hojas completar mayor

```
Class NodoC
{ private
    String nombre
    Int contacto
    NodoC izq,der
  public
   NodoC()
  {
      Izq=der=null
  }
  Getters
  setters
}
```




```
Class ABinarioC
{  private
       NodoC raíz
   Public
    ABinarioC(){raíz=null} //árbol vacio
    GetRaiz/setRaiz
    Crear() //por niveles
    {  nivel,desc=new CSimple()
      raíz=new NodoC()//setRaiz(new NodoC())
      nivel.adicionar(getRaiz())
    while not nivel.esvacia()
    {
      while not nivel.esvacia()
      {   r=nivel.eliminar()
        read(n,c)
        r.setNombre(n);r.setContacto(c)
        print(r.getNombre(),"Tendra Izq? s/n") read(resp)
        if resp="s"
        {
            nue=new NodoC()
            r.setIzq(nue)
            desc.adicionar(nue)
        }
        Print("Tendra Der? s/n") read(resp)
        if resp="s"
        {
            nue=new NodoC()
            r.setDer(nue)
            desc.adicionar(nue)
        }
      }//while
      nivel.vaciar(desc)
    }
    }
 Mostrar()
 {  nivel,desc=new CSimple()
```

```
Crear() //por niveles
    {  nivel,desc=new CSimple()
      raíz=new NodoC()//setRaiz(new NodoC())
      read(n,c)
      raiz.setNombre(n)
      raiz.setContacto(c)
      nivel.adicionar(getRaiz())
    while not nivel.esvacia()
    {
      while not nivel.esvacia()
      {   r=nivel.eliminar()
        print(r.getNombre(),"Tendra Izq? s/n") read(resp)
        if resp="s"
        {
            nue=new NodoC()
            read(n,c)
           nue.setNombre(n);nue.setContacto(c)
            r.setIzq(nue)
            desc.adicionar(nue)
        }
        Print("Tendra Der? s/n") read(resp)
        if resp="s"
        {
            nue=new NodoC()
           read(n,c)
           nue.setNombre(n);nue.setContacto(c)
            r.setDer(nue)
            desc.adicionar(nue)
        }
      }//while
      nivel.vaciar(desc)
    }
    }
```

```
   nivel.adicionar(getRaiz())
  while not nivel.esvacia()
 {
    while not nivel.esvacia()
    {  r=nivel.eliminar()
       Print(r.getNombre(),r.getContacto())
       If r.getIzq()<>null
            desc.adicionar(r.getIzq())
        if r.getDer()<>null
             desc.adicionar(r.getDer())
     }
     nivel.vaciar(desc)
  }
}
}
```

un arbol de objetos personas (nom,pat,mat,edad)

Persona
nom pat mat edad
Getters Setters Leer mostrar

NodoP
Persona p izq der
NodoP() Getters setters

ABinario
Raíz
ABinario() getRaiz setRaiz crear niveles hojas completar mayor

Cola
max NodoP v[mx+1] Ini,fin
Cola() Esvacia Esllena Nroelem Adicionar Eliminar vaciar

```
Class NodoP
{ private
     Persona p
     NodoP izq,der
  Public
    NodoP()
   {   izq=der=null
    }
    Getters
    setters
}
```

```
Class ABinario
{ private
       NodoP raíz
```

```
completar()
{  nivel, desc=new Cola()
   nivel.adicionar(getRaiz())
```

```
  public
     ABinario()
    { raiz=null}
    getRaiz
    setRaiz
    crea()
   {  nivel, desc=new Cola()
      raíz=new NodoP()
      nivel.adicionar(getRaiz())

while not nivel.esvacia()
{       while not nivel.esvacia()//procesa un nivel
        {    r=nivel.eliminar()
             px=new Persona()
             px.leer()
             r.setP(px)
             print("tendrá izq? s/n") read(resp)
             if resp="s"
             {   nue=new NodoP()
                 r.setIzq(nue)
                 desc.adicionar(nue)
             }
             print("tendrá der? s/n") read(resp)
             if resp="s"
             {   nue=new NodoP()
                 r.setDer(nue)
                 desc.adicionar(r.getDer())
             }
        }
        nivel.vaciar(desc)//pasa sgte nivel
} //paso de nivel

   }
```

```
 while not nivel.esvacia()
 {
    while not nivel.esvacia()// recorre un nivel
    {
        r=nivel.eliminar()
       //proceso
        If r.getIzq()=null and r.getDer()<>null
        { nue=new NodoP()
          px=new Persona() ; px.leer();
          nue.setP(px)
          r.setIzq(nue)
        }
       If r.getIzq()<>null and r.getDer()=null
      {
          nue=new NodoP()
          px=new Persona() ; px.leer();
          nue.setP(px)
          r.setDer(nue)
      }
       if r.getIzq()<>null
            desc.adicionar(r.getIzq())
       if r.getDer()<>null
            desc.adicionar(r.getDer())
    }
   nivel.vaciar(desc)//paso de nivel

 }
}
```

```
Niveles()
{  nivel, desc=new Cola()
  nivel.adicionar(getRaiz())
 c=0
  while not nivel.esvacia()
 {  print("NIVEL →" ,c)
    while not nivel.esvacia()// recorre un nivel
    {
        r=nivel.eliminar()
       //proceso
```

```
//nodos con 2 descendientes
2descendientes()
{  nivel, desc=new Cola()
  nivel.adicionar(getRaiz())
   while not nivel.esvacia()
 {
    while not nivel.esvacia()// recorre un nivel
    {
        r=nivel.eliminar()
       //proceso
```

```
          r.getP().mostrar()
          if r.getIzq()<>null
              desc.adicionar(r.getIzq())
          if r.getDer()<>null
              desc.adicionar(r.getDer())
      }
      nivel.vaciar(desc)//paso de nivel
      c=c+1
   }
}
MayorEdad()
{  nivel, desc=new Cola()
   nivel.adicionar(getRaiz())
  may=0
   while not nivel.esvacia()
  {
      while not nivel.esvacia()// recorre un nivel
      {
          r=nivel.eliminar()
          //proceso
          If r.getP().getEdad()>may
              may=r.getP().getEdad()
          if r.getIzq()<>null
              desc.adicionar(r.getIzq())
          if r.getDer()<>null
              desc.adicionar(r.getDer())
      }
      nivel.vaciar(desc)//paso de nivel
  }
print(may)
}
```

```
          If r.getIzq()<>null and r.getDer()<>null
              r.getP().mostrar()
          if r.getIzq()<>null
              desc.adicionar(r.getIzq())
          if r.getDer()<>null
              desc.adicionar(r.getDer())
      }
      nivel.vaciar(desc)//paso de nivel
  }
}
2desc(NodoP r)
{
   If r<>null
   {
        If r.getIzq()<>null and r.getDer()<>null
              r.getP().mostrar()
        2desc(r.getIzq())

         2desc(r.getDer())

   }
}
```

```
Contar(NodoP r)
{  nivel, desc=new Cola()
   nivel.adicionar(r)
  c=0
   while not nivel.esvacia()
  {
      while not nivel.esvacia()// recorre un nivel
      {
           r=nivel.eliminar()
          //proceso
           c=c+1
          if r.getIzq()<>null
               desc.adicionar(r.getIzq())
          if r.getDer()<>null
               desc.adicionar(r.getDer())
      }
     nivel.vaciar(desc)//paso de nivel
  }
 return c
}
```

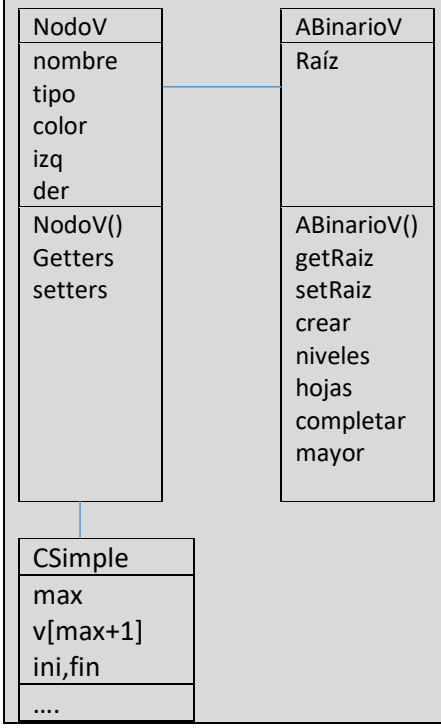
```
Verifica()
{
     r=getRaiz()
     If Contar(r.getIzq())=Contar(r.getDer())
          Print("SI")
    Else
       Print("NO")
}
ContarR(NodoP r)
{
    If r<>null
        Return ContarR(r.getIzq())+ContarR(r.getDer())+1
    Else
      Return 0
}
Verifica2()
{
  if ContarR(getRaiz().getIzq())=ContarR(getRaiz().getDer())
      Print("SI")
    Else
       Print("NO")

}
```

EJEMPLO: Sea un Arbol binario de Verduras(nombre, tipo, color)

TIPO:

- Semillas: haba, frijol, soya.
- Tubérculos: papa, camote.
- Raíces: zanahoria, rábano.
- Tallos: espárrago, puerro.
- Hojas: espinaca, col.
- Bulbos: cebolla, ajo.
- Flores: brócoli, alcachofa.

DIAGRAMA DE CLASES



```
Class NodoV
{ private
   String nombre,tipo,color
    NodoV  izq,der
  Public
      NodoV()
      { izq=der=null}
     Getters/setters
}
```

```
Class ABinarioV
{ private
       NodoV raíz
  Public
     ABinarioV(){raiz=null}
     getRaiz/setRaiz
 crear(
 { nivel,desc=new CSimple()
         raiz=new NodoV()
         nivel.adicionar(getRaiz())
  while not nivel.esvacia
  {
      while not nivel.esvacia()
      {     r=nivel.eliminar()
            read(n,t,c)
            r.setNombre(n);r.setTipo(t);r.setColor(c)
          print("tendrá izq? s/n") read(resp)
          if resp="s"
          {   nue=new NodoV()
              r.setIzq(nue)
              desc.adicionar(nue)
          }
          print("tendrá der? s/n") read(resp)
          if resp="s"
          {   nue=new NodoV()
              r.setDer(nue)
              desc.adicionar(nue)
          }
      }
      nivel.vaciar(desc)
  }
 }
 mostrarNiv()
 {  nivel,desc=new CSimple()
    c=0  //nivel 0
    nivel.adicionar(getRaiz())
  while not nivel.esvacia()
  {  print("Nivel →", c)
    while not nivel.esvacia()
    {    NodoV r=nivel.eliminar()
         Print(r.getNombre(),r.getTipo(),r.getColor())
        If r.getIzq()<>null
            desc.adicionar(r.getIzq()
        if r.getDer()<>null
            desc.adicionar(r.getDer())
    }
    nivel.vaciar(desc)
    c=c+1  //siguiente nivel
```

```
  }
 }
mostrarNivK(int k)
 {  nivel,desc=new CSimple()
   c=0  //nivel 0
    nivel.adicionar(getRaiz())
  while not nivel.esvacia()
  {  print("Nivel →", c)
    while not nivel.esvacia()
    {    r=nivel.eliminar()
        If c=k
           Print(r.getNombre(),r.getTipo(),r.getColor())
        If r.getIzq()<>null
            desc.adicionar(r.getIzq()
        if r.getDer()<>null
            desc.adicionar(r.getDer())
    }
    nivel.vaciar(desc)
    c=c+1  //siguiente nivel
  }
 }
 //Contar hojas por nivel
hojasNiv()
 {  nivel,desc=new CSimple()
   c=0  //nivel 0
    nivel.adicionar(getRaiz())
  while not nivel.esvacia()
  {  print("Nivel →", c)
     K=0
    while not nivel.esvacia()
    {    r=nivel.eliminar()
        If r.getIzq()=null and r.getDer()=null
            K=K+1
        If r.getIzq()<>null
            desc.adicionar(r.getIzq()
        if r.getDer()<>null
            desc.adicionar(r.getDer())
    }
    Print("Hojas → ",K)
    nivel.vaciar(desc)
    c=c+1  //siguiente nivel
  }
 }
```

```
completar()
 {  nivel,desc=new CSimple()
    nivel.adicionar(getRaiz())
  while not nivel.esvacia()
  {
    while not nivel.esvacia()
    {    r=nivel.eliminar()

        If r.getIzq()≠null and r.getDer()=null
        { nue=new NodoV()
          Read(n,t,c);nue.setNombre(n)
           nue.setTipo(t); nue.setColor(c)
           r.setDer(nue)
        }
       If r.getIzq()=null and r.getDer()<>null
        { nue=new NodoV()
          Read(n,t,c);nue.setNombre(n)
           nue.setTipo(t);  nue.setColor(c)
           r.setIzq(nue)
        }
        If r.getIzq()<>null
            desc.adicionar(r.getIzq()
        if r.getDer()<>null
            desc.adicionar(r.getDer())
    }
    nivel.vaciar(desc)
   }
 }

}
```

```
proceso()
 {  nivel,desc=new CSimple()
    nivel.adicionar(getRaiz())
  while not nivel.esvacia()
  {
    while not nivel.esvacia()
    {    r=nivel.eliminar()
        //PROCESO
        If r.getIzq()<>null
            desc.adicionar(r.getIzq()
        if r.getDer()<>null
            desc.adicionar(r.getDer())
    }
    nivel.vaciar(desc)
   }
 }
```

TAREA: con recorrido en amplitud

1. Determinar la cantidad total de verduras tipo X
2. Determinar la cantidad de verduras del color X en cada nivel
3. Verificar si en los nodos hoja existe la verdura “Coliflor”
4. Mostrar a cada con su descendiente izquierdo.
5. Cuantos nodos tienen solo un descendiente
6. Cuantos nodos tienen sus dos descendientes
7. Determinar la cantidad de nodos de un árbol