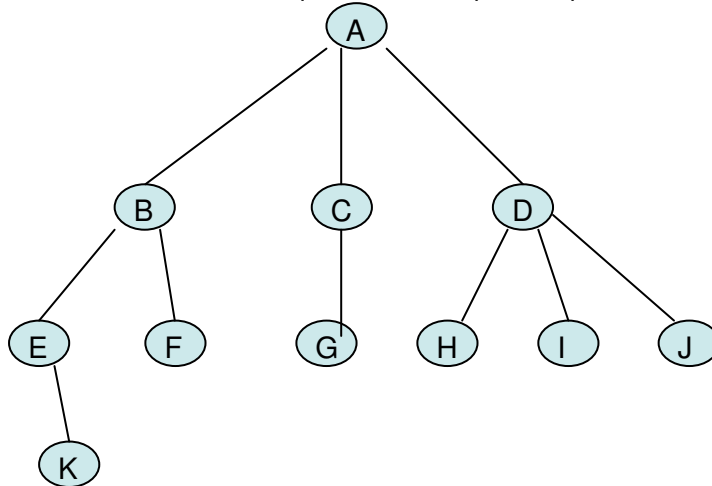




Introducción

Un árbol es una estructura no lineal en la que cada nodo puede apuntar a uno o varios nodos.



Clasificación con respecto a su relación:

- **Nodo hijo:** cualquiera de los nodos apuntados por uno de los nodos del árbol. En el ejemplo, 'E' y 'F' son hijos de 'B'.
- **Nodo padre:** nodo que contiene una referencia al nodo actual. En el ejemplo, el nodo 'A' es padre de 'B', 'C' y 'D'.

Clasificación con respecto a su posición:

- **Nodo raíz:** Si el árbol no está vacío es el primer nodo del árbol; este nodo que no tiene padre. Este es el nodo que usaremos para referirnos al árbol. En el ejemplo, ese nodo es el 'A'.
- **Nodo hoja:** nodo que no tiene hijos. En el ejemplo hay varios: 'K', 'F', 'G', 'H', 'I' y 'J'.
- **Nodo rama:** aunque esta definición apenas la usaremos, estos son los nodos que no pertenecen a ninguna de las dos categorías anteriores. En el ejemplo: 'B', 'C', 'D' y 'E'.

Clasificación con respecto a su tamaño:

- **Orden:** es el número potencial de hijos que puede tener cada elemento de árbol. De este modo, diremos que un árbol en el que cada nodo puede apuntar a otros dos es de orden dos, si puede apuntar a tres será de orden tres, etc.
- **Grado:** el número de hijos que tiene el elemento con más hijos dentro del árbol. En el árbol del ejemplo, el grado es tres, ya que tanto 'A' como 'D' tienen tres hijos, y no existen elementos con más de tres hijos.
- **Nivel:** se define para cada elemento del árbol como la distancia a la raíz, medida en nodos. El nivel de la raíz es cero y el de sus hijos uno. Así sucesivamente. En el ejemplo, el nodo 'D' tiene nivel 1, el nodo 'G' tiene nivel 2, y el nodo 'K', nivel 3.
- **Altura:** la altura de un árbol se define como el nivel del nodo de mayor nivel. Como cada nodo de un árbol puede considerarse a su vez como la raíz de un árbol, también podemos hablar de altura de ramas. El árbol del ejemplo tiene altura 3, la rama 'B' tiene altura 2, la rama 'G' tiene altura cero, la 'H' cero, etc.



TEMA3. ESTRUCTURAS DE DATOS COMPUESTAS

Árboles binarios: Son aquellos árboles de orden 2.

Cada flecha en un árbol representará un puntero a otro nodo del árbol. No es necesario que todos los hijos de un nodo concreto existan, podrán utilizarse varios, uno o ninguno de los punteros. Es importante conservar siempre el nodo raíz ya que es el nodo a partir del cual se desarrolla el árbol, si perdemos este nodo, perderemos el acceso a todo el árbol.

Declaraciones de arboles en C#

Una estructura de un arbol binario sería la siguiente:

```
Class arbol
{
    nodo Raiz=null;
    nodo hoja = new nodo();
}
```

```
Class nodo
{
    public int dato;
    public nodo izq;
    public nodo der;
}
```

Generalizando para cualquier tipo de arbol tendríamos:

```
Class arbol
{
    nodo Raiz=null;
    nodo hoja = new nodo();
}
```

```
Class nodo
{
    public int dato;
    public nodo[ ] rama;
}
```

El movimiento a través de árboles, salvo que implementemos punteros al nodo padre, será siempre partiendo del nodo raíz hacia un nodo hoja. Cada vez que lleguemos a un nuevo nodo podremos optar por cualquiera de los nodos a los que apunta para avanzar al siguiente nodo.



Arboles binarios de búsqueda

Un árbol binario de búsqueda(ABB) es un árbol binario con la propiedad de que todos los elementos almacenados en el subárbol izquierdo de cualquier nodo x son menores que el elemento almacenado en x ,y todos los elementos almacenados en el subárbol derecho de x son mayores que el elemento almacenado en x.

La figura 1 muestra dos ABB construidos en base al mismo conjunto de enteros:

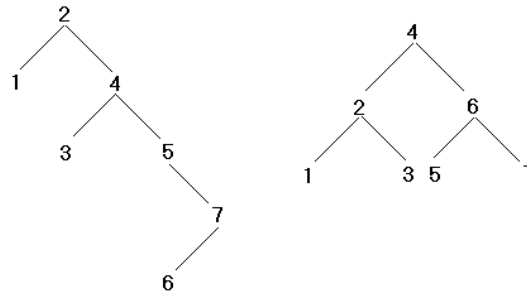


Figura 1: Dos ABB con los mismos elementos.

Obsérvese la interesante propiedad de que si se listan los nodos del ABB en inorden nos da la lista de nodos ordenada. Esta propiedad define un método de ordenación similar al Quicksort, con el nodo raíz jugando un papel similar al del elemento de partición del Quicksort aunque con los ABB hay un gasto extra de memoria mayor debido a los punteros. La propiedad de ABB hace que sea muy simple diseñar un procedimiento para realizar la búsqueda. Para determinar si k está presente en el árbol la comparamos con la clave situada en la raíz, r. Si coinciden la búsqueda finaliza con éxito, si k<r es evidente que k, de estar presente, ha de ser un descendiente del hijo izquierdo de la raíz, y si es mayor será un descendiente del hijo derecho.

Operaciones básicas en árboles

- Añadir o insertar elementos.
• Buscar o localizar elementos.
• Borrar elementos.
• Moverse a través del árbol.
• Recorrer el árbol completo.

Recorridos en árboles

Pre-orden

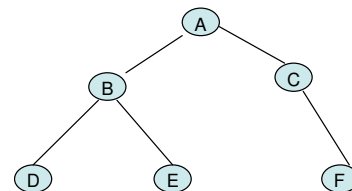
- Nodo raíz
• Subárbol izquierdo
• Subárbol derecho

1) A-B-D-E-C-F

In-orden

- Subárbol izquierdo
• Nodo raíz

2) D-B-E-A-C-F





TEMA3. ESTRUCTURAS DE DATOS COMPUESTAS

- Subárbol derecho

Post-orden

- Subárbol izquierdo
 - Subárbol derecho
 - Nodo raíz
- 3) D-E-B-F-C-A

Eliminar nodos en un árbol

El proceso general es muy sencillo en este caso, pero con una importante limitación, sólo podemos borrar nodos hoja:

El proceso sería el siguiente:

1. Buscar el nodo padre del que queremos eliminar.
2. Buscar el puntero del nodo padre que apunta al nodo que queremos borrar.
3. Liberar el nodo.
4. padre.nodo[i] = NULL

Cuando el nodo a borrar no sea un nodo hoja, diremos que hacemos una "poda", y en ese caso eliminaremos el árbol cuya raíz es el nodo a borrar. Se trata de un procedimiento recursivo, aplicamos el recorrido PostOrden, y el proceso será borrar el nodo.

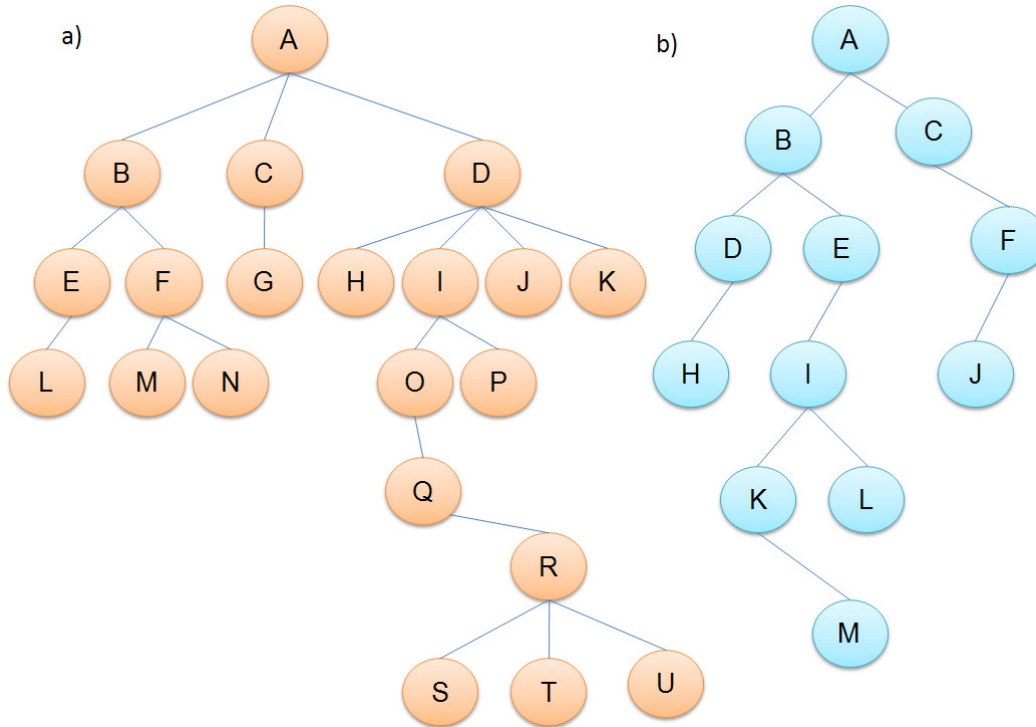
El procedimiento es similar al de borrado de un nodo:

1. Buscar el nodo padre del que queremos eliminar.
2. Buscar el puntero del nodo padre que apunta al nodo que queremos borrar.
3. Podar el árbol cuyo padre es nodo.
4. padre.nodo[i] = NULL;



Ejercicios:

1) Responder a las siguientes preguntas de con respecto los siguientes árboles:



Para el árbol del inciso a

- 1.1. ¿Qué nodo es la raíz?
- 1.2. ¿Cuál es el grado del árbol?
- 1.3. ¿Cuál es la altura del árbol?
- 1.4. ¿Qué nodos son los hijos de D?
- 1.5. ¿Qué nodos son las hojas?
- 1.6. ¿Cuál es el nivel del nodo Q?
- 1.7. ¿Cuál es la altura de Q?
- 1.8. ¿Es G hermano a la izquierda de H?
- 1.9. ¿Cuántos hijos tiene R?
- 1.10. ¿Cuál es el nivel del nodo U?

Para el árbol del inciso b

- 1.11. Listar los nodos del árbol en preorden, postorden e inorden.
- 2. Dada la siguiente secuencia de números, dibuje en un diagrama la estructura del árbol binario de búsqueda resultante.
5, 7, 9, 11, 3, 2, 1, 6, 5, 0, 8, 10, 25, 19, 15, 20
- 3. El recorrido en preorden de un determinado árbol binario es ABDGILECFHJMK y en inorden ILGDBEACJMHKF
Contestar a los siguientes incisos:
 - a) Dibuje el árbol binario resultante
 - b) Listar el recorrido postorden



Arboles binarios de búsqueda

1. Crea una nueva solución llamada **practica17arbolesbb** y llama a tu proyecto de tipo librería de clases **arboles**. Dentro del proyecto escribe el código de la parte inferior.
 - a. A continuación compíllalo y anota tus observaciones.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace arboles
{
    public class class_arboles
    {
        public nodo RAIZ=null;

        public void inserta_nodo(nodo A, nodo padre, int valor, int rama)
        {
            if(A == null)
            {
                A = new nodo();
                A.dato = valor;
                A.izq = null;
                A.der = null;
                if (RAIZ == null)
                    RAIZ = A;
                else
                    if (rama==1)
                        padre.izq = A;
                    else if (rama==2)
                        padre.der = A;
            }
            else
            {
                if (valor < A.dato)
                    inserta_nodo(A.izq, A, valor, 1);
                else if (valor > A.dato)
                    inserta_nodo(A.der, A, valor, 2);
                else
                    Console.WriteLine("Dato duplicado");
            }
        }

        public void preorden(nodo A)
        {
            if (A != null)
            {
                Console.Write("{0}-> ", A.dato);
                preorden(A.izq);
                preorden(A.der);
            }
        }

        public void inorden(nodo A)
```



TEMA3. ESTRUCTURAS DE DATOS COMPUESTAS

```
{
    if(A!=null)
    {
        inorden(A.izq);
        Console.WriteLine("{0}->", A.dato);
        inorden(A.der);
    }
}

public void postorden(nodo A)
{
    if (A != null)
    {
        postorden(A.izq);
        postorden(A.der);
        Console.WriteLine("{0}->", A.dato);
    }
}
}
public class nodo
{
    public int dato;
    public nodo izq;
    public nodo der;
}
}
```

- b. Dentro de la misma solución crea otro proyecto de tipo Consola. Agrega una referencia a la librería **class_arboles.dll**; posteriormente agregar el código de la parte inferior. Coméntalo, compílalo y ejecútalo.

```
using System;
using System.Collections.Generic;
using System.Text;
using arboles;

namespace ejercicios
{
    class Program
    {
        static void Main(string[] args)
        {
            class_arboles miArbol=new class_arboles();
            for (int i = 1; i <= 10; i++)
            {
                Console.WriteLine("Insertado {0} en el arbol binario", i);
                miArbol.inserta_nodo(miArbol.RAIZ, null, i, 0);
            }
            Console.WriteLine("\nRecorrido preorden: \n");
            miArbol.preorden(miArbol.RAIZ);
            Console.WriteLine("\n\n Recorrido inorden:\n");
            miArbol.inorden(miArbol.RAIZ);
            Console.WriteLine("\n\n Recorrido postorden:\n");
            miArbol.postorden(miArbol.RAIZ);
        }
    }
}
```

- c. Modifica el código para que puedas ingresar los valores desde teclado.
d. Crea un método dentro de la librería que permita calcular la altura de un árbol bb.

Practica17. Arboles