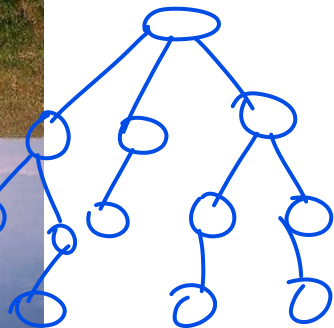


Estructura de Datos: Árbol



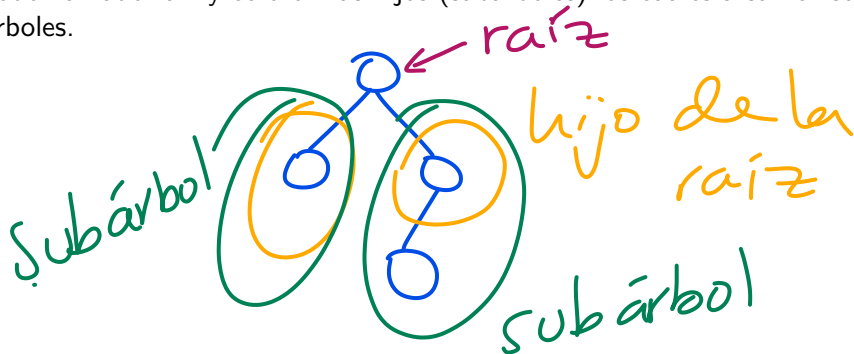
Dr. Jaime Osorio Ubaldo

Árbol



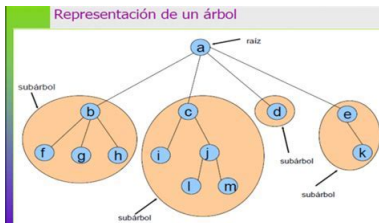
Árbol

Un árbol es una colección de nodos, la cual puede ser vacía o tener un nodo llamado raíz y cero o más hijos (subárboles) los cuales a su vez son árboles.



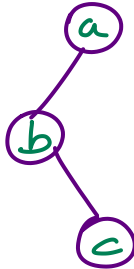
Árbol

Un árbol es una colección de nodos, la cual puede ser vacía o tener un nodo llamado raíz y cero o más hijos (subárboles) los cuales a su vez son árboles.



Elementos de un Árbol

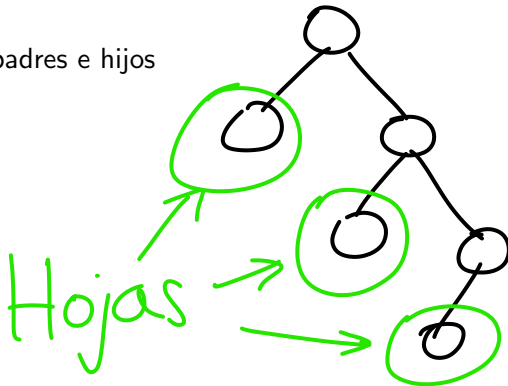
- 1 Nodos padres e hijos



b es hijo de a
a es padre de b
b es padre de c
c es hijo de b

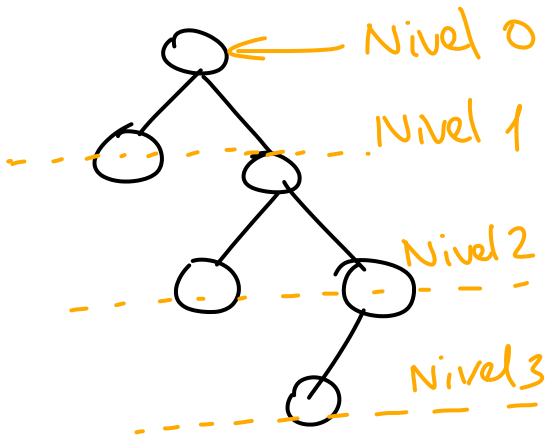
Elementos de un Árbol

- 1 Nodos padres e hijos
- 2 Hojas



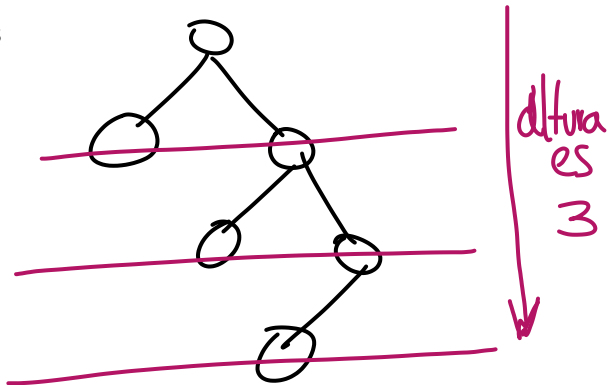
Elementos de un Árbol

- 1 Nodos padres e hijos
- 2 Hojas
- 3 Nivel del nodo



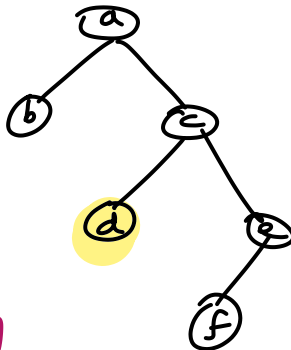
Elementos de un Árbol

- 1 Nodos padres e hijos
- 2 Hojas
- 3 Nivel del nodo
- 4 Altura del árbol.



Elementos de un Árbol

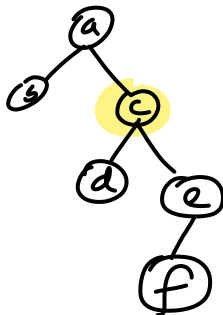
- 1 Nodos padres e hijos
- 2 Hojas
- 3 Nivel del nodo
- 4 Altura del árbol.
- 5 Ascendientes del nodo.



ascendientes de d son c y a

Elementos de un Árbol

- 1 Nodos padres e hijos
- 2 Hojas
- 3 Nivel del nodo
- 4 Altura del árbol.
- 5 Ascendientes del nodo.
- 6 Descendientes del nodo.



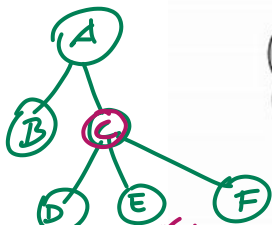
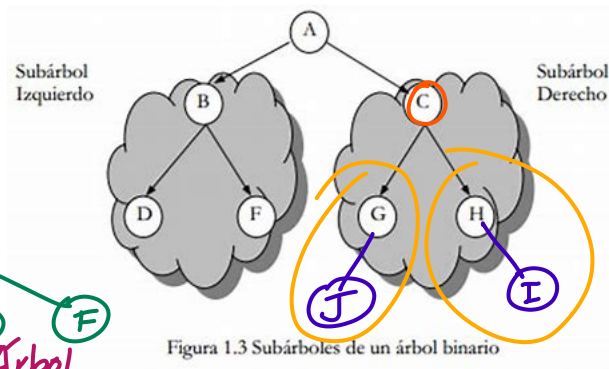
descendientes de c son d, e y f

Árbol Binario

Un árbol binario es un árbol que consta de a lo más dos hijos y estos son a su vez árboles binarios . Estos dos hijos son denominados dos subárbol izquierdo y subárbol derecho del árbol original.

Árbol Binario

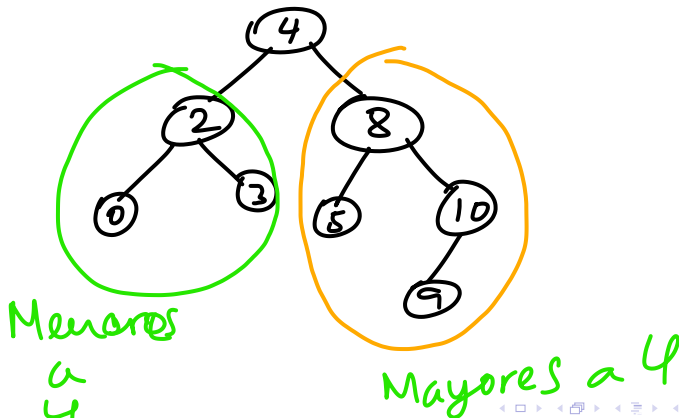
Un árbol binario es un árbol que consta de a lo más dos hijos y estos son a su vez árboles binarios. Estos dos hijos son denominados subárbol izquierdo y subárbol derecho del árbol original.



NO es Árbol
Binario

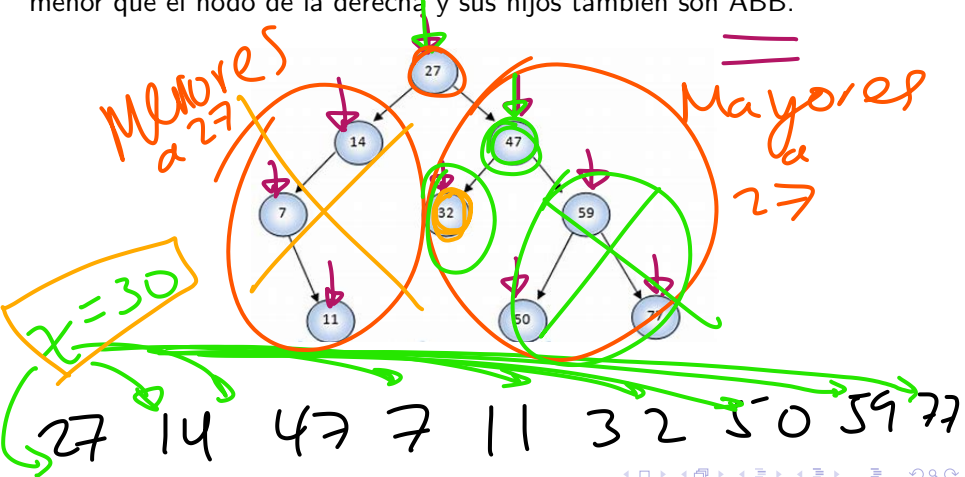
Árbol Binario de Búsqueda

Es un árbol binario en el cual el nodo de la izquierda contiene un valor menor que el nodo de la derecha y sus hijos también son ABB.



Árbol Binario de Búsqueda

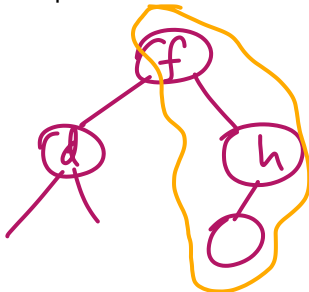
Es un árbol binario en el cual el nodo de la izquierda contiene un valor menor que el nodo de la derecha y sus hijos también son ABB.



- 1 Almacenar elementos de mayor a menor.

Aplicaciones

- 1 Almacenar elementos de mayor a menor.
- 2 Para búsqueda de palabras de un diccionario.

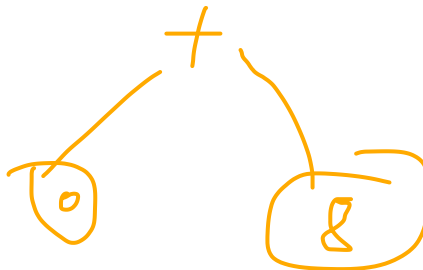


Aplicaciones

- ① Almacenar elementos de mayor a menor.
- ② Para búsqueda de palabras de un diccionario.
- ③ Almacena objetos que se identifiquen por una clave que sea ordenable.

Aplicaciones

- 1 Almacenar elementos de mayor a menor.
- 2 Para búsqueda de palabras de un diccionario.
- 3 Almacena objetos que se identifiquen por una clave que sea ordenable.
- 4 En compiladores, para la construcción del árbol sintáctico.

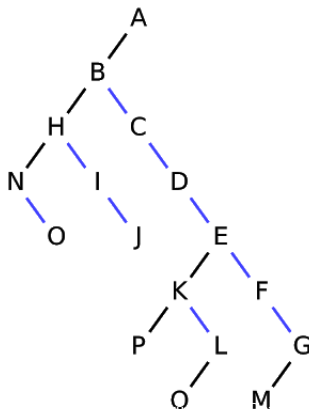
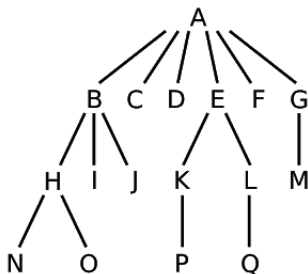


Aplicaciones

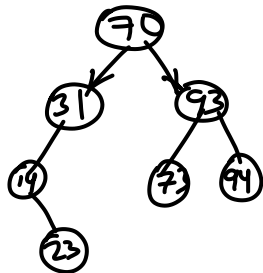
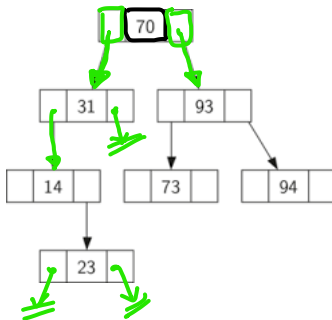
- ① Almacenar elementos de mayor a menor.
- ② Para búsqueda de palabras de un diccionario.
- ③ Almacena objetos que se identifiquen por una clave que sea ordenable.
- ④ En compiladores, para la construcción del árbol sintáctico.
- ⑤ En base de datos relacionales, para la búsqueda de un registro, etc.

Aplicaciones

- 1 Almacenar elementos de mayor a menor.
- 2 Para búsqueda de palabras de un diccionario.
- 3 Almacena objetos que se identifiquen por una clave que sea ordenable.
- 4 En compiladores, para la construcción del árbol sintáctico.
- 5 En base de datos relacionales, para la búsqueda de un registro, etc.



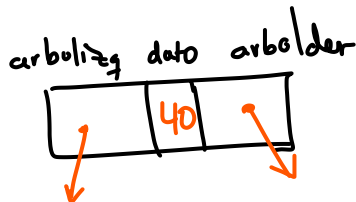
La estructura del nodo



Operaciones: insertar, buscar, eliminar, etc.

Estructura de Datos para el ABB

```
struct nodoArbol{  
    int dato;  
    nodoArbol *arbolizq;  
    nodoArbol *arbolder;  
};  
typedef struct nodoArbol pnodoArbol;
```




Clase Árbol

```
class arbol{  
private: Arbol  
    pnode praiz;  
public:  
    arbol();  
    ~arbol();  
    void insertar(int );  
    void imprimir(pnode);  
    pnode getRaiz();  
    void eliminaarbol(pnode);  
};
```

Árbol vacío:

raiz

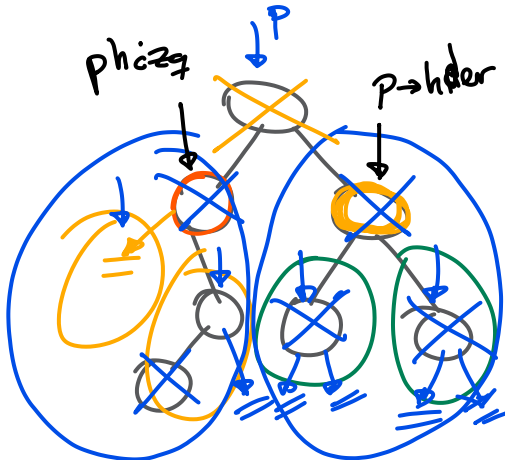


```
arbol::arbol(){  
    raiz=NULL;  
}
```


P

```
void arbol::eliminararbol(pnodo p){
    if(p!=NULL){
        eliminararbol(p->hizq);
        eliminararbol(p->hder);
        delete p;
    };
};

arbol::~arbol(){
    eliminararbol(praiz);
};
```



Raíz del árbol

```
pnode arbol::getRaiz(){  
    return praiz;  
};
```

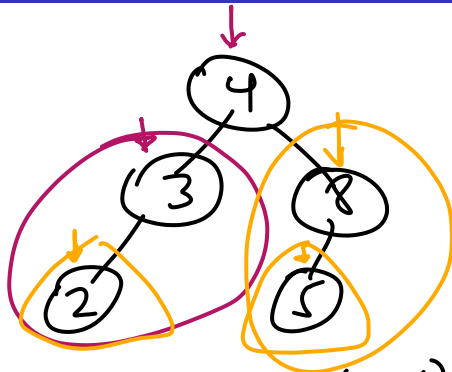
praiz → //

Insertar

```
void insertarNodo(int x){
    pnodeArbol p,q,padreq;
    char hijo;
    p=new nodoArbol;
    p->dato=x; p->arbolizq=NULL; p->arbolder=NULL; q=p-raiz;
    if(q==NULL)
        raiz=p;
    else{
        while(q!=NULL){
            padreq=q;
            if(x<q->dato) {q=q->arbolizq; hijo='i';}
            else if(x>q->dato) {q=q->arbolder; hijo='d';}
        }if(hijo=='i')
            padreq->arbolizq=p;
        else
            padreq->arbolder=p;
    }
}
```

Imprimir

```
void imprimir(pnodoArbol r){  
    if(r != NULL){  
        imprimir(r->arbolizq);  
        cout<<r->dato;  
        imprimir(r->arbolder);  
    }  
}
```



imprimir(4)

imprimir(3)

4

imprimir(8)

imprimir(2)

3

imprimir(NULL)

imprimir(5)

8

imprimir(NULL)

imprimir(NULL)

imprimi(NULL)

imprimir(NULL)

imprimi(NULL)

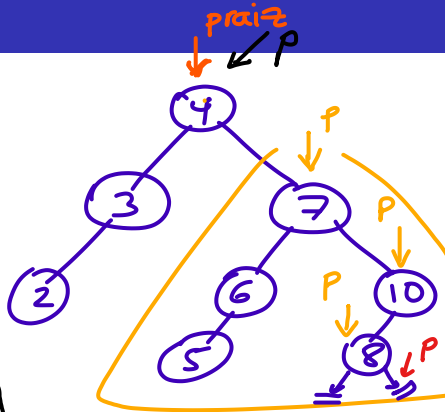
2 3 4 5 8

Buscar

praiz
p →

x=9

```
pnodoArbol buscar(int x){  
    pnodoArbol p;  
    p=praiz;  
    if(p==NULL)  
        return p;  
    else{  
        while(p!=NULL){  
            if(p->dato==x) NO  
                return p;  
            else if(x>p->dato)  
                p=p->arbolder;  
            else  
                p=p->arbolizq; }  
        return p;  
    }  
}
```



retorna NULL
→ NO lo encuentro

Dado un ABB, determine el menor elemento de este árbol.