Árbol binario de búsqueda .

Estructura de datos eficiente que permite buscar, insertar y borrar cualquier elemento o cualquier rango de elementos.

Un árbol binario de búsqueda es un árbol binario, que puede ser vacío y si no es vacío cumple las siguientes propiedades:

- Todos los elementos tienen una clave y no existen dos elementos con igual clave.
- Las claves (si existen) del subárbol izquierdo son menores que la clave en la raíz.
- Las claves (si existen) del subárbol derecho son menores que la clave en la raíz.
- Los subárboles izquierdo y derecho son también árboles binarios de búsqueda.

Árbol de binario búsqueda. Especificación

especificación ARBOL-BUSCA[A es ELEM-ORD] **es**

usa BOOL

instancia privada ARBOL-BIN[B es ELEM] donde B.elem es A.elem

tipos arbol-busca

operaciones

insertar	:	elem arbol-busca	→	arbol-busca
buscar	:	elem arbol-busca	→	arbol-busca
borrar	:	elem arbol-busca	→	arbol-busca
esta	:	elem arbol-busca	→	bool
privada min	:	arbol-busca	→	elem

variables

iz,dr,a	:	arbol-busca
x,y	:	elem

ecuaciones

min(arbol-vacio)	=	$\text{error}_{\text{arbol-busca}}$
min(plantar(iz,x,dr))	=	x ⇐ vacio?(iz)
min(plantar(iz,x,dr))	=	min(iz) ← ¬ vacio?(
insertar(y,arbol-vacio)	=	plantar(arbol-vacio,
insertar(x,plantar(iz,x,dr))	=	plantar(iz,x,dr)
insertar(y,plantar(iz,x,dr))	=	plantar(insertar(y,iz
insertar(y, plantar(iz,x,dr))	=	plantar(iz,x,insertar
buscar(x,arbol-vacio)	=	arbol-vacio
buscar(x,plantar(iz,x,dr))	=	plantar(iz,x,dr)
buscar(y,plantar(iz,x,dr))	=	buscar(y,iz) ⇐ $y <$
buscar(y, plantar(iz,x,dr))	=	buscar(y,dr) ⇐ $y >$

borrar(y,arbol-vacio)	=	arbol-vacio
borrar(x,plantar(iz,x,dr))	=	dr ⇐ vacio?(iz)
borrar(x,plantar(iz,x,dr))	=	iz ⇐ vacio?(dr)
borrar(x,plantar(iz,x,dr))	=	plantar(iz,min(dr),bor
		⇐ ¬ vacio?(iz) ∧¬ va
borrar(y,plantar(iz,x,dr))	=	plantar(borrar(y,iz),x,
borrar(y,plantar(iz,x,dr))	=	plantar(iz,x,borrar(y,d
esta(x,a)	=	¬ vacio?(buscar(x,a))

fespecificación

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
- - -
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
- - -
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Á nario. Implementación

```
te
cl
{

                                io
                                usqueda(const TipoClave &x);
                                usqueda(nodo_arbol<TipoClave> * b, const TipoClave &x);
                                Clave & x );
                                x );

  p                             iz;

                                _arbol<TipoClave> * );
};
```


Bús

```
tem                             rbolBinarioBusqueda<TipoClave>
::B                             )
{                               ,x);   }

tem                             rbolBinarioBusqueda<TipoClave>::
                                poClave> * b, const TipoClave &x)
{

                                 )

                                )
                                da(b->izq,x);
                                r,x);
}
```

Co rofundidad del árbol

Búsqueda del elemento menor

```
// Esta función es llamada con al menos un elemento en su hijo derecho
template <class TipoClave>
     TipoClave ArbolBinarioBusqueda<TipoClave>::
       Buscar_Min(nodo_arbol<TipoClave>* aux )
     {
           while (aux -> izq != NULL)
                aux = aux -> izq;
           return aux->elemento;
     }
```

Inserción de un elemento

Complejidad: $\mathcal{O}(h)$ donde h es la profundidad del árbol


```
template <class TipoClave>
 bool ArbolBinarioBusqueda<TipoClave>::insertar( const TipoClave & x )
 {     // busqueda del lugar a insertar x, q es el padre de p
    nodo_arbol<TipoClave> *p = raiz, *q=0 ;
    while( p )
    {     q = p;
          if( x == p-> elemento ) return false;
          if( x < p->elemento ) p = p->izq;
         else p = p->der;
    }

    p = new nodo_arbol<TipoClave>;
    p->izq = p->der = 0;
    p-> elemento = x;
    if (!raiz) raiz = p;
    else if ( x < q->elemento ) q->izq = p;
          else q->der = p;
    return true;
}
```

Borrado de un elemento

- Si el elemento a borrar es una hoja del árbol el campo correspondiente del padre se pone a `null` y se libera la memoria.
- Si el elemento a borrar tiene un único hijo, el hijo toma el lugar del nodo que se elimina y se libera la memoria del nodo que se elimina.
- Si el elemento a borrar tiene dos hijos, el elemento se sustituye por el mayor de los elementos del subárbol izquierdo o por el menor de los elementos del subárbol derecho. A continuación se procede a eliminar este elemento y se libera la memoria.

Complejidad: $\mathcal{O}(h)$ donde h es la profundidad del árbol


```
template <class TipoClave>
bool ArbolBinarioBusqueda<TipoClave>::borrar( TipoClave & x )
{  // busqueda del elemento x, q es el padre de p
    nodo_arbol<TipoClave> *p = raiz, *q=0 ;
    bool encontrado = false;
    while( p && encontrado == false )
    {   if ( x == p->elemento )
            encontrado = true;
        else
        {  q = p;
            if( x < p->elemento )
                p = p->izq;
            else
               p = p->der;
        }
    }
    // p apunta al elemento a borrar
    // q apunta al padre del elemento a borrar
```



```
if( !p )
    return false;          // Si no se ha e
if( p->izq == NULL && p->der == NULL ) //
{       if ( !q && p == raiz)    // si es l
        {
            raiz = NULL;
            delete p;
            return true;
        }

        if( q->izq == p )
            q->izq = NULL;
        else if( q->der == p )
                q->der = NULL;
        delete p;
        return true;          // se elimina
}
```



```
if( p->izq != NULL && p->der == NULL ) //
{       if ( !q && p == raiz)    // si es
        {  raiz = p->izq;
            delete p;
            return true;
        }
        if( q->izq == p )
            q->izq = p->izq;
        else if( q->der == p )
                q->der = p->izq;
        delete p;
        return true;          // se elimina
}
if( p->izq == NULL && p->der != NULL ) //
{       if ( !q && p == raiz)    // si es l
        {  raiz = p->der;
            delete p;
            return true;
        }
```


CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
- - -
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.

www.cartagena99.com no se hace responsable de la información contenida en el presente documento en virtud al Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.


```
er;
= p )
 p->der;

    // se elimina y finaliza

der != NULL ) // si tiene dos hijos
Buscar_Min(p->der);

 x;
```

}

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70
- - -
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Á nario balanceado.

La queda binario de n elementos puede llegar a ser n.

Los los permiten realizar las operaciones de búsqueda, inse ad $\mathcal{O}(log(n))$.

Los de búsqueda balanceados son:

-
-
-
-
-