

Estructura de Datos

Sexto encuentro

Contenidos

- Operaciones en árboles binarios.

Árbol Binario de Búsqueda

Debido a la gran diversidad de estructuras de datos soportadas sobre árboles binarios (dependiendo del objetivo con que se necesite), para definir las operaciones en los árboles binarios, nos basaremos en una estructura especial de gran utilidad práctica. Esta utilidad está dada puesto que se define un orden dentro del árbol.

Árbol Binario de Búsqueda

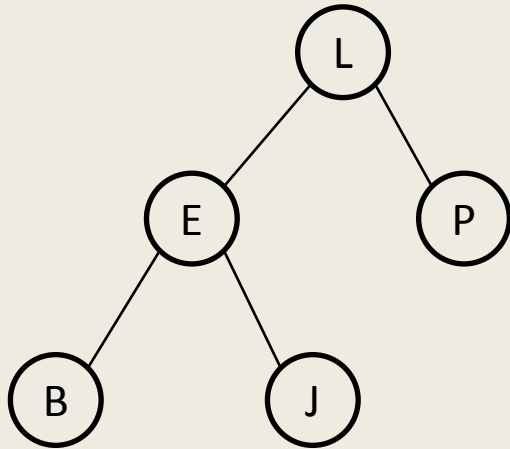
Es una estructura cuyos elementos están organizados como un árbol binario. Los nodo de este árbol están ordenados de tal forma que para cualquier nodo x del árbol, si y es un nodo cualquiera del sub-árbol izquierdo de x , entonces $valor[y] < valor[x]$ y si y es un nodo cualquiera del sub-árbol derecho de x , entonces $valor[x] < valor[y]$.

Árbol Binario de Búsqueda

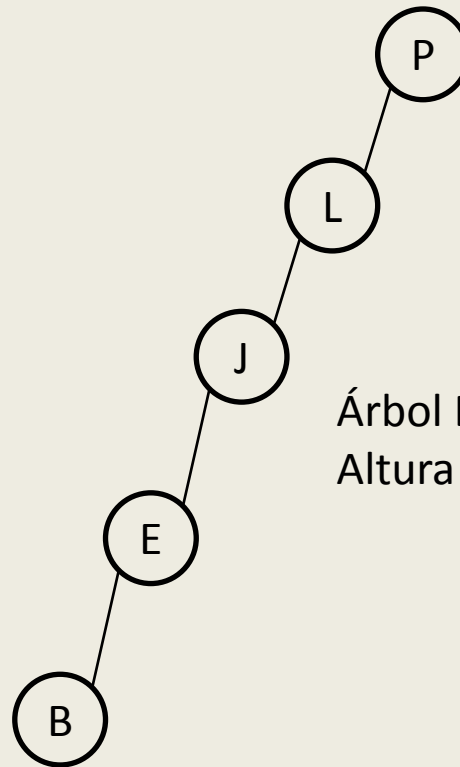
Concretamente, los elementos deben almacenarse en el árbol de tal forma que los valores en el sub-árbol izquierdo de la raíz sean menores que el valor de la raíz y los valores del sub-árbol derecho sean mayores que la raíz.

Otra consecuencia importante de esta propiedad es que un recorrido en **EnOrden** de un árbol binario de búsqueda visitará siempre los nodos del árbol en orden.

Ejemplos



Árbol Binario de Búsqueda
Altura = 2



Árbol Binario de Búsqueda
Altura = 4

Operaciones

- Buscar(k: Objeto): Devuelve el nodo en el que se encontró k o NIL en caso contrario.
- Máximo y Mínimo(): Devuelve el nodo que contiene el valor máximo (mínimo), del árbol.
- Predecesor y El:Sucesor(k: Objeto): Devuelve el nodo que contiene el elemento predecesor (sucesor) de k o NIL si no existe.
- Insertar(k: Objeto): Devuelve verdadero en caso de insertar el valor y falso en caso del que valor a insertar ya existiera en el árbol.
- El:Eliminar(k: Objeto):

Jerarquía

ÁrbolBinarioB hereda de ÁrbolBinario



Entre la clase ÁrbolBinarioB y la clase ÁrbolBinario se establece una relación de generalización.

ÁrbolBinarioB
Clase Pública

ÁrbolBinarioB(Nodo)
Constructor Público
+ Buscar(Objeto): Nodo
+ Máximo(): Nodo
+ Mínimo(): Nodo
+ Predecesor(Objeto): Nodo
+ Sucesor(Objeto): Nodo
+ Insertar(Objeto): Lógico
+ Eliminar(Objeto): Lógico

Operaciones

Constructor ÁrbolBinarioB(raíz: Nodo)

inicio

// Llamara al constructor de la clase base
base(raíz)

fin

Buscar(dato: Objeto)

inicio

este.árbol ← este.Raíz

mientras este.Árbol ≠ NIL hacer

si este.Árbol.Dato = dato entonces
devolver este.Árbol

fin_si

si dato < este.Árbol.Dato entonces
este.árbol ← este.Árbol.HijoIzquierdo

si_no

este.árbol ← este.Árbol.HijoDerecho

fin_si

fin_mientras
devolver NIL

fin

Operaciones

Máximo(): Nodo

inicio

este.árbol ← este.Raíz

mientras este.Árbol.HijoDerecho ≠ NIL hacer

 este.árbol ← este.Árbol.HijoDerecho

fin_mientras

devolver este.Árbol

fin

Mínimo(): Nodo

inicio

este.árbol ← este.Raíz

mientras este.Árbol.HijoIzquierdo ≠ NIL hacer

 este.árbol ← este.Árbol.HijoIzquierdo

fin_mientras

devolver este.Árbol

fin

Operaciones

Predecesor(dato: Objeto): Nodo

var

subÁrbol: ÁrbolBinarioB

padre: Nodo

inicio

este.árbol ← este.Buscar(dato)

si este.Árbol = NIL entonces

devolver NIL

fin_si

si este.Árbol.Hijolzquierdo ≠ NIL entonces

subÁrbol ← Nuevo ÁrboBinarioB(este.Árbol.Hijolzquierdo)

devolver subÁrbol.Máximo()

fin_si

padre ← este.Padre(este.Árbol)

mientras padre ≠ NIL y padre ≠ este.Raíz y este.Árbol = padre.Hijolzquierdo hacer

este.árbol ← este.Árbol.HijoDerecho

padre ← este.Padre(padre)

fin_mientras

devolver padre

fin

Operaciones Insertar I

Insertar(nodo: Nodo): Lógico

var

bandera: Lógico

padre: Nodo

inicio

bandera ← verdadero

este.árbol ← este.Raíz

si nodo.Dato > este.Raíz.Dato entonces

bandera ← falso

fin_si

si este.Árbol ≠ NIL entonces

este.raíz ← nodo

fin_si

mientras este.Árbol ≠ NIL hacer

si nodo.Dato = este.Árbol.Dato entonces

devolver falso

fin_si

si nodo.Dato < este.Árbol.Dato entonces

si bandera entonces

padre ← este.Árbol

este.árbol ← este.Árbol.Hijolzquierdo

bandera ← verdadero

Operaciones Insertar II

si_no

nodo.HijoDerecho ← este.Árbol

padre.HijoDerecho ← nodo

este.árbol ← este.Raíz

devolver verdadero

fin_si

si_no

si bandera entonces

nodo.Hijolzquierdo ← este.Árbol

padre.Hijolzquierdo ← nodo

este.árbol ← este.Raíz

devolver verdadero

si_no

padre ← este.Árbol

este.árbol ← este.Árbol.HijoDerecho

bandera ← falso

fin_si

fin_si

fin_mientras

si nodo.Dato < padre.Dato entonces

padre.Hijolzquierdo ← nodo

Operaciones Insertar III

si_no

padre.HijoDerecho ← nodo

fin_si

este.Árbol ← este.Raíz

devolver verdadero

fin

Estudio independiente

Desarrolle los algoritmos de los métodos ***Sucesor*** y ***Eliminar*** de la clase ÁrbolBinarioB.