Algoritmos y Estructuras de Datos

UCU

Arboles Binarios

1

Resultados Esperados del Aprendizaje



El propósito general de este módulo es presentarte estructuras básicas de listas, pilas colas; y discutir la pertinencia de su aplicación en diferentes situaciones. A la vez se pretende comenzar a analizar criterios comparativos para la selección de los métodos de representación más apropiados, como órdenes del tiempo de ejecución y recursos de memoria involucrados.

Al finalizar exitosamente esta unidad de aprendizaje serás capaz de:

- Expresar en lenguaje natural el proceso de solución de un problema planteado, considerando el contexto, precondiciones y casos particulares a ser tenidos en cuenta
- Expresar en seudocódigo estándar soluciones a problemas procedurales sencillos.
 Construir programas JAVA que implementen los TDA lista, pila y cola utilizando sus representaciones más apropiadas para resolver problemas reales.
- Comparar los costos de memoria de diferentes representaciones y elegir la más apropiada, en función de estos criterios, de acuerdo al problema específico y contexto asociado.
- Reconocer y aplicar estándares de codificación en Java
 November 1 de codificación en Ja
- Utilizar eficientemente las interfases e implementaciones de Listas provistas en la API de Colecciones de JAVA

Algoritmos y Estructuras de Datos

2

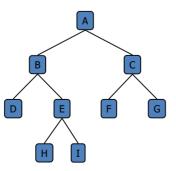
2

Arboles Binarios



- Un Arbol Binario es un Arbol con las
 - siguientes propiedades:

 Cada nodo interno tiene como máximo dos hijos (exactamente dos para árboles binarios completos)
 - Los hijos de un nodo son un par ordenado
- Los hijos de un nodo interno se denominan Hijo Izquierdo e Hijo Derecho
- Definiciones alternativas:
 - Un árbol con un sólo nodo, o un árbol cuya raíz tiene un par ordenado de hijos, cada uno de los cuales es a su vez un árbol bipario.
 - Conjunto finito de nodos, que puede estar vacío o consistir de una raíz y dos árboles binarios disjuntos, llamados subárbol izquierdo y derecho de la raíz.
- Applicaciones:
 - Expresiones aritméticasProcesos de decisión
 - Búsqueda

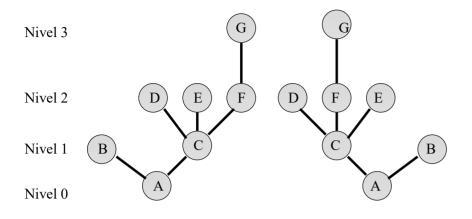


Algoritmos y Estructuras de Datos

3

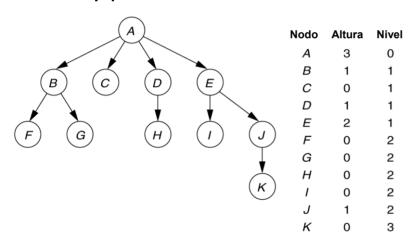
ARBOLES





Altura y profundidad o nivel





ARBOLES: Representación

- La raíz arriba, las hojas abajo.
- Se dice que cada raíz es el "padre" de las raíces de sus subárboles, los cuales son llamados "hermanos".
- Los subárboles son llamados "hijos" de su "padre".
- La raíz del árbol total no tiene padre.
- Ancestros y descendientes.

Algoritmos y Estructuras de Datos

Lleno y completo

⊕UCU

- Arbol binario Lleno
- Arbol binario completo

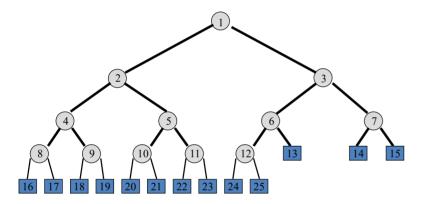
Algoritmos y Estructuras de Datos

7

⊕UCU

7

ARBOL BINARIO COMPLETO



Algoritmos y Estructuras de Datos

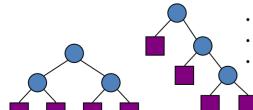
8

Propiedades de los Arboles Binarios llenos



Notación

- *n* número de nodos
- e número de nodos externos
- *i* número de nodos internos
- h altura



- Propiedades:
 - e = i + 1
 - n = 2e 1
 - $h \leq i$
 - $h \le (n-1)/2$
 - $e \le 2^h$
 - $h \ge \log_2 e$
 - $h \ge \log_2(n+1) 1$

Jaoritmos y Estructuras de Datos



Operaciones en arboles



- Operaciones básicas.
- Inserción, búsqueda, eliminación
- Recorridos

binarios

- Preorden
- Postorden
- inorden

Algoritmos y Estructuras de Datos

11

11

Recorrida en Preorden



•	En un recorrido en preorden, un nodo es visitado antes que sus descendientes	Algoritmo preOrden(v) visitar(v) Para cada hijo w de v
•	Aplicación: imprimir un documento estructurado 1	preorden (w)
	1. Introducción 3 4 6 1.1 General 1.2 Aplicación 2.1 Insertar	5 2. Métodos Casos

Algoritmos y Estructuras de Datos

Recorrida en Postorden



•	En un recorrido en postorden, un nodo es visitado después de sus descendientes	Algoritmo postOrden(v) Para cada hijo w de v postOrden (w)
•	Aplicación: calcular el espacio usado por archivos en un directorio y sus 9 subdirectorios	visitar(v)

subdirectorios

p2/

trabajos/

programas/

h1c.doc

3

DDR.java

10K

Stocks.java

20K

Robot.java

20K

Algoritmos y Estructuras de Datos

13

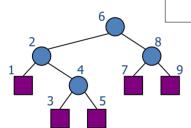
Recorrido en Inorden de un árbol binario



 En un recorrido en inorden el nodo es visitado después que su subárbol izquierdo y antes que su subárbol derecho

Algoritmo TNodoAB.InOrden
Si tiene HijoIzquierdo
HijoIzquierdo.inOrden
visitar(v)
Si tiene HijoDerecho

HijoDerecho. inOrden



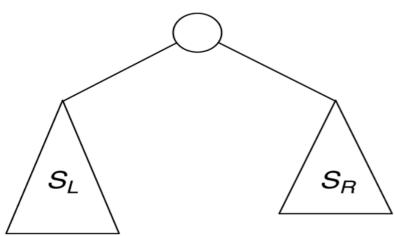
Algoritmos y Estructuras de Datos

14

14

Vista recursiva usada para calcular el tamaño de un árbol ST = SL + SR + 1

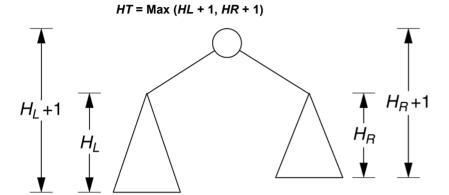




Algoritmos y Estructuras de Datos

15

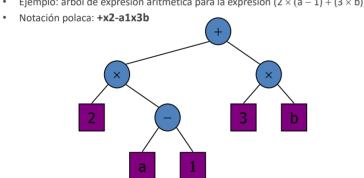
Vista recursiva usada para calcular la altura de un 😩 UCU nodo:



16

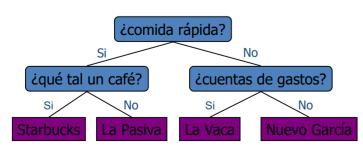
Arbol de Expresión Aritmética **⊕**UCU

- Arbol binario asociado con una expresión aritmética:
 - Nodos internos: operadores
 - Nodos externos: operandos
- Ejemplo: árbol de expresión aritmética para la expresión (2 \times (a 1) + (3 \times b))



Aplicación de Arbol Binario: Arbol de Decision UCU

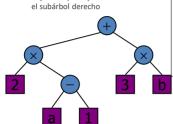
- Arbol binario asociado con un proceso de decisión
 - Nodos internos : preguntas con respuestas si / no
 - Nodos externos : decisiones
- Ejemplo: decisión de cena



Impresión de expresiones aritméticas



- Especialización del recorrido en inorden
 - Imprimir el operando u operador
 - Imprimir "(" antes de recorrer el subárbol izquierdo
 - Imprimir ")" después de recorrer el subárbol derecho



Algoritmo TNodoAB.printExpression

> Si tieneHijolzquierdo *imprimir*("(")

HijoIzquierdo.printExpression imprimir

Si tieneHijoDerecho

HijoDerecho .printExpression imprimir(")")

$$((2 \times (a - 1)) + (3 \times b))$$

Algoritmos y Estructuras de Datos

Evaluar expresiones aritméticas



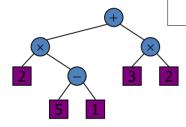
- Especialización del recorrido en postorden
 - Método recursivo que retorna el valor de un subárbol
 - Al visitar un nodo interno, combina los

Si esHoja Devolver *elemento* $x \leftarrow HijoIzquierdo .evalExpr$ $y \leftarrow HijoDerecho.evalExpr$

♦ ← operador contenido

Algoritmo TNodoAB.evalExpr

devolver $x \diamond y$



Algoritmos y Estructuras de Datos

Ejercicios Arboles Binarios



- Define árbol binario. Dá un ejemplo de organización usando esta estructura.
- Representa la siguiente expresión aritmética utilizando un árbol binario: a - (b * (c + d / (f + g)) + h * (i - j * (k + I)))

y dá un algoritmo para, utilizando este árbol, evaluar la expresión cuando las variables toman valores.

El recorrido en preorden de un cierto árbol binario produce ADFGHKLPQRWZ

y el recorrido en inorden produce

GFHKDLAWRQPZ

Dibuje el árbol binario correspondiente.

Algoritmos y Estructuras de Datos

Εj	erc	icios	Arbo	les	Bina	arios
----	-----	-------	------	-----	------	-------



Sean p(x), s(x) e i(x) las posiciones del nodo x en preorden, postorden e inorden respectivamente.

•Marque en el cuadro siguiente las posiciones que pueden ser ciertas simultáneamente.

	i(n) < i(m)	s(n) < s(m)	p(n) < p(m)
n es ancestro de m	10	100	1000
n es descendiente de m	10	100	1000
n está a la izquierda de m	10	100	1000
n está a la derecha de m	10	100	1000

Algoritmos y Estructuras de Datos

22

22

Ejercicios Arboles Binarios



- Escriba un algoritmo para contar las hojas de un árbol binario
- Dado un árbol binario de elementos de tipo entero, escriba un algoritmo que calcule la suma de todos los elementos.
- Escribir un algoritmo que determine la cantidad de nodos que se encuentran en un árbol binario en un nivel n

Algoritmos y Estructuras de Datos

23

23

ARBOL BINARIO DE BUSQUEDA



- Se dice que un árbol binario es de búsqueda si está organizado de forma que:
 - para cada nodo t_i todas las claves del subárbol izquierdo de t_i son menores que la clave de t_i y todas las claves del árbol derecho son mayores.
- Si el árbol tiene n nodos y está balanceado, su altura será de log(n)
- Una búsqueda en este caso puede tomar log(n) comparaciones o menos.

Algoritmos y Estructuras de Datos

Árbol binario de búsqueda	⊕ UCU	
TDA ArbolBinario Raiz : ElementoArbolBinario Primitivas Buscar (UnaEtiquetaqueta) : ElementoArbo	lBinario	
ArbolBinario.Buscar(UnaEtiquetaqueta)	.5	
COM resultado = nulo si Raiz <> nulo ENTONCES		
resultado = Raiz.Buscar(UnaEtiquetaqueta) Devolver resultado FIN		
Algoritmos y Estructuras de Datos	25	
,		
Arbol binario de búsqueda	⊕ UCU	
TDA Flamenta Anhal Dinania		
TDA ElementoArbolBinario Etiqueta : TipoEtiqueta		
Hijolzquierdo, Hijo Derecho: Elemento Ark	oolBinario	
Operaciones		
Insertar(unElementoArbolBinario) Buscar(UnaEtiquetaqueta)		
Eliminar(UnaEtiquetaqueta)		
Algoritmos y Estructuras de Datos	26	
TArbolBB	⊕ UCU	
public class TArbolBB {	⊜ 300	
TElementoAB raiz;		
<pre>public TArbolBB () {} // a Implementar public boolean esVacio() {} // a Implementar</pre>		
public boolean insertar(TElementoAB unElemento) {}		
public TElementoAB buscar (Comparable UnaEtiquetaqueta) {	}	
public void preOrden() {} // a Implementar	•	
public void inOrden() {}		

27

public void postOrden() $\{...\}$ // a Implementar

Algoritmos y Estructuras de Datos

TElementoAB	⊜ U0	U	
public class TElementoAB <t>{ Comparable etiqueta; TElementoAB hijolzq;</t>			
TElementoAB hijoDer; <t> datos;</t>			
// a implementar public TElementoAB <t> (Comparable UnaE</t>	tiquetaqueta, <t> unosDatos) {}</t>		
public boolean insertar(TElementoAB unEle	mento) {}		
public TElementoAB buscar(Comparable Un	naEtiquetaqueta) {}		
public void preOrden() {}			
public void inOrden() {}			
<pre>public void postOrden() {} }</pre>			
Algoritmos y E	structuras de Datos	28	

29

Arbol binario de busqueda



De TElementoAB

Buscar(UnaEtiquetaqueta): TElementoABinario

```
COM

RESULTADO = nulo

SI UnaEtiquetaqueta = Etiqueta ENTONCES

RESULTADO = THIS
SINO

SI UnaEtiquetaqueta < Etiqueta ENTONCES

SI Hijolzquierdo <> nulo ENTONCES

RESULTADO = Hijolzquierdo.Buscar(UnaEtiquetaqueta)

FINSI
SINO

SI HijoDerecho <> nulo ENTONCES

RESULTADO = HijoDerecho.Buscar(UnaEtiquetaqueta)

FINSI
FINSI
FINSI
FINSI
FINSI
devolver RESULTADO
FIN
```

Algoritmos y Estructuras de Datos

29

29

TArbolBB -Buscar una Etiqueta



```
// de TArbolBB
public TElementoAB buscar (Comparable
UnaEtiquetaqueta) {
  if (esVacio()) {
    return null;
  } else {
    return raiz.buscar(UnaEtiquetaqueta);
  }
}
```

Algoritmos y Estructuras de Datos

30

TElementoAB-Buscar Etiqueta	⊕ UCU
<pre>public TElementoAB buscar(Comparable UnaEtiquetaqueta) { if (UnaEtiquetaqueta.compareTo(etiqueta) == 0) {</pre>	
return this;	
} else {	
if (UnaEtiquetaqueta.compareTo(etiqueta) < 0) {	
if (hijolzq != null) {	
return hijolzq.buscar(UnaEtiquetaqueta);	
} else {	
return null;	
}	
} else {	
if (UnaEtiquetaqueta.compareTo(etiqueta) > 0) {	
if (hijoDer != null) {	
return hijoDer.buscar(UnaEtiquetaqueta);	
} else {	
return null;	
}	
} else {	
return null;	
}	
}	
}	
1	

Algoritmos y Estructuras de Datos

Inserción en árboles binarios



De ArbolBinario

Insertar(unElementoArbolBinario)

COM

SI Raiz = nulo ENTONCES

Raiz = unElementoArbolBinario

SINO

Raiz.Insertar(unElementoArbolBinario)

FIN

Algoritmos y Estructuras de Datos

32

Inserción en árboles binarios



De TElementoAB

Insertar(UnElementoArbolBinario)

SI Etiqueta = unElementoArbolBinario.Etiqueta ENTONCES

SALIR // ya está en el árbol FINSI

SI unElementoArbolBinario.Etiqueta < Etiqueta ENTONCES

SI Hijolzquierdo = nulo ENTONCES

HijoIzquierdo ← unElementoArbolBinario

SINO Hijolzquierdo.Insertar(unElementoArbolBinario)

SINO SI HijoDerecho = nulo ENTONCES

HijoDerecho ←unElementoArbolBinario

SINO HijoDerecho.Insertar(unElementoArbolBinario) FINSI

FINSI

FIN

Algoritmos y Estructuras de Datos

Implementación	de
Recorridos ABB	

⊕UCU

- Preorden
- Inorden
- Postorden

Algoritmos y Estructuras de Datos

34

2/

TArbolBB - Inorden

⊕UCU

```
public String inOrden(){
   if (raiz == null) return "arbol vacio";
      else return raiz.inOrden();
}
```

Algoritmos y Estructuras de Datos

35

TElementoAB - Inorden

⊕UCU

```
public String inOrden() {
    String tempStr = "";
    if (hijolzq != null) {
        tempStr = hijolzq.inOrden();
    }
    tempStr = tempStr + imprimir();
    if (hijoDer != null) {
        tempStr = tempStr + hijoDer.inOrden();
    }
    return tempStr;
}
```

Búsqueda y eliminación en árboles binario de búsqueda

UCU

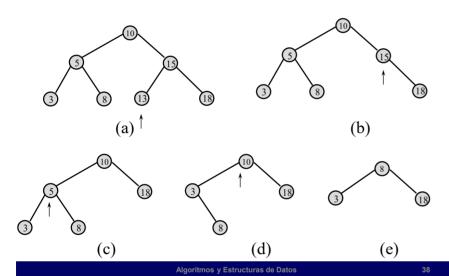
- Se busca el nodo con la etiqueta a eliminar.
- Si no existe un nodo con esa etiqueta, la eliminación es infructuosa.
- Si existe y es una hoja, se elimina el nodo directamente.
- Si no es una hoja, pero le falta alguno de sus dos sub árboles, la operación es sencilla, ya que basta con una reasignación de referencias.
- Si el nodo a eliminar es un nodo interno completo, deben sustituirse sus datos y etiqueta por el los del nodo de mayor etiqueta de su sub árbol izquierdo (su "inmediato anterior" en el orden lexicográfico dado).
- Luego se elimina ese elemento, que será una hoja, o le faltará el sub árbol derecho.

Algoritmos y Estructuras de Datos

__

ELIMINACION EN ARBOL BINARIO DE BUSQUEDA

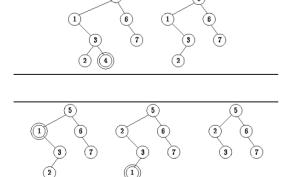




38

Eliminación

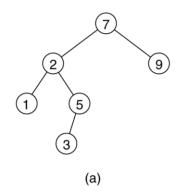


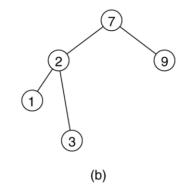


Algoritmos y Estructuras de Datos

Eliminación del nodo 5 con 1 hijo: (a) antes y (b) después







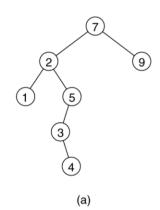
Algoritmos y Estructuras de Dato

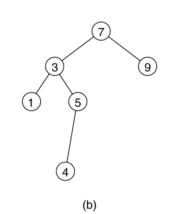
40

40

Eliminación del nodo 2 con dos hijos: (a) antes y (b) después







Algoritmos y Estructuras de Datos

41

41

Búsqueda y eliminación en árboles binarios de búsqueda

⊕UCU

ArbolBinario.Eliminar(UnaEtiqueta)

COM

SI Raiz <> nulo ENTONCES

Raiz ← Raiz.Eliminar(UnaEtiqueta)

SI NO

mensaje "árbol vacío"

FIN

Algoritmos y Estructuras de Datos

árboles binarios de búsqueda			
(1) Si UnaEtiqueta < etiqueta entonces	// si esta, está en el subárbol izquierdo)	
Si hijolzq <> nulo entonces hijolzq ← hijolzq.eliminar(UnaEtiqueta) Finsi	//actualiza el hijo, con el mismo u otro	valor	
retornar (this)	//al padre le devuelve el mismo hijo		
Finsi			
(2) Si UnaEtiqueta > etiqueta entonces Si hijoDer <> nulo entonces	// si esta, está en el subárbol derecho		
hijoDer \leftarrow hijoDer.eliminar(UnaEtiqueta)	//actualiza el hijo, con el mismo u otro	valor	
Finsi			
retornar (this)	// al padre le devuelve el mismo hijo		
Finsi			
(3) retornar quita El Nodo	// esta, hay que eliminarlo		
	// al padre le devuelve el nuevo hijo		
Fin			

Algoritmos y Estructuras de Datos

44

Búsqueda y Eliminación en árboles binarios de búsqueda

// Cuando encuentra el nodo a eliminar llama, por claridad, al método que hace el trabajo



TElementoAB.quitaElNodo: de Tipo TElementoAB;

Comienzo

(1) Si hijolzq = nulo entonces retornar hijoDer // puede retornar un nulo

(2) Si hijoDer = nulo entonces retornar hijolzq // le falta el hijo derecho retornar hijolzq

(3) // es un nodo completo elHijo ← hijolzq elPadre ← this mientras elHijo.hijoDer <> nulo hacer elPadre ← elHijo elHijo ← elHijo elHijo ← elHijo elHijo ← elHijo.hijoDer fin mientras // elHijo.hijoDer fin mientras // elHijo.hijoDer ← elHijo.hijolzq elHijo.hijolzq ← hijolzq Finsi // elHijo.hijoDer ← hijoDer retornar elHijo // elHijo quedara en lugar de this

Fin

Algoritmos y Estructuras de Datos

45

Análisis de Búsqueda e Inserción en Arbol Binario de Búsqueda



- Los algoritmos de inserción y búsqueda analizados no controlan el balanceo del árbol.
- Se desconoce a priori la forma en que el árbol ha
- Si el árbol estuviera balanceado, para encontrar una clave se precisarían aproximadamente log n comparaciones.
- El peor caso se da para n/2 comparaciones.
- El problema es encontrar la cantidad promedio de comparaciones.
- n claves, n! árboles correspondientes a n! permutaciones de claves.

Algoritmos y Estructuras de Datos

47

Ejercicios Arboles Binarios de Búsqueda



- ¿Qué es un árbol binario de búsqueda?
 Desarrolle el algoritmo de búsqueda correspondiente y evalúe su rendimiento.
- Desarrolle el algoritmo de eliminación en árbol binario de búsqueda y evalúe su rendimiento
- Muestre el resultado de insertar los elementos 3,1,4,6,9,2,5 y 7 en un árbol binario de búsqueda inicialmente vacío. Muestre después el resultado de eliminar la raíz

Algoritmos y Estructuras de Datos

48

48

Arboles Balanceados

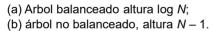


- Criterio de Adelson, Velskii y Landis:
- Un árbol está balanceado si y sólo si para cada nodo las alturas de sus dos subárboles difieren a lo sumo en 1.
 - Asegura un O log(n) en el peor caso para las tres operaciones: búsqueda, inserción y eliminación.
 - El teorema de Adelson-Velski y Landis garantiza que un árbol balanceado nunca tendrá una altura mayor que el 45 % respecto a su equivalente perfectamente balanceado.
 - Si la altura de un árbol balanceado de n nodos es hb(n), entonces
 - $\log(n+1) \le hb(n) \le 1.4404*\log(n+2) -0.328$

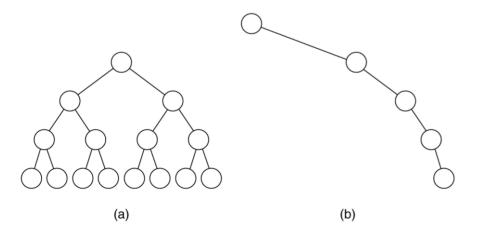
Algoritmos y Estructuras de Datos

49

40



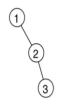




Algoritmos y Estructuras de Datos

Arboles binarios de búsqueda posibles insertando las claves 1, 2 y 3 de diferentes maneras

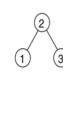


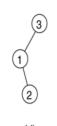


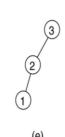
(a)







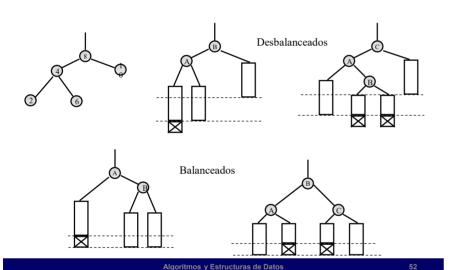




51

ARBOLES BALANCEADOS





52

Arboles Balanceados



- Criterio de Adelson, Velskii y Landis:
 - Un árbol está balanceado si y sólo si para cada nodo las alturas de sus dos subárboles difieren a lo sumo en 1.
 - Asegura un O log(n) en el peor caso para las tres operaciones: búsqueda, inserción y eliminación.
 - Se demuestra con la ayuda de los árboles de Fibonacci.

Algoritmos y Estructuras de Datos

AVL



- Una inserción o eliminación puede destruir el balance
- Se debe revisar la condición de balance y corregir si es necesario
- Después de la Inserción, sólo los nodos que se encuentran en el camino desde el punto de inserción hasta la raíz pueden tener el balance alterado
- Si se repara correctamente el equilibrio del nodo desequilibrado más profundo, se recupera el equilibrio de todo el árbol

Algoritmos y Estructuras de Datos

54

54

Diferentes condiciones de desequilibrio



- Supongamos que X es el nodo cuyo equilibrio queremos ajustar
- Se pueden dar cuatro casos:
 - Inserción en el subárbol izquierdo del hijo izquierdo de X
 - Inserción en el subárbol derecho del hijo izquierdo de X
 - Inserción en el subárbol izquierdo del hijo derecho de X
 - Inserción en el subárbol derecho del hijo derecho de X
- 1 y 4 son simétricos respecto a X
- 2 y 3 son simétricos respecto a X

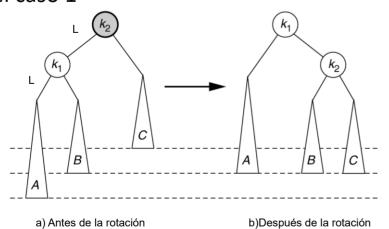
Algoritmos y Estructuras de Datos

55

55

Rotación Simple para resolver el caso 1

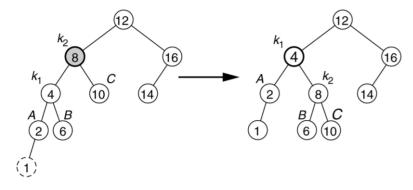




Algoritmos y Estructuras de Datos

La rotación simple balancea el árbol después de la inserción del 1.





a) Antes de la rotación

b)Después de la rotación

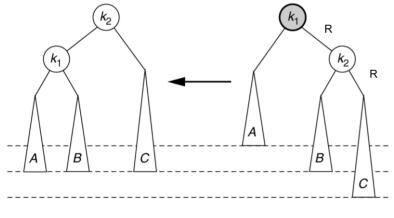
Algoritmos y Estructuras de Date

57

57

Rotación simple simétrica para resolver el caso 4





a) Antes de la rotación

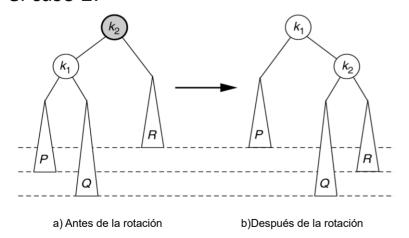
b)Después de la rotación

Algoritmos y Estructuras de Datos

58

58

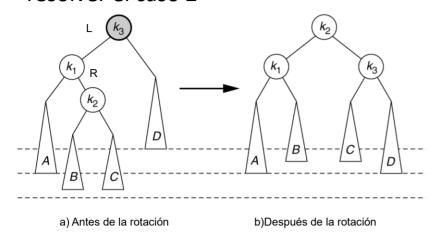
La rotación simple no resuelve **UCU** el caso 2.



Algoritmos y Estructuras de Datos

Rotación doble LR para resolver el caso 2



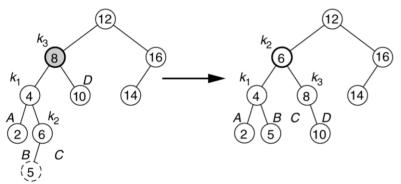


Algoritmos y Estructuras de Datos

60

60

Rotación doble restablece el balance después de la inserción del 5.



a) Antes de la rotación

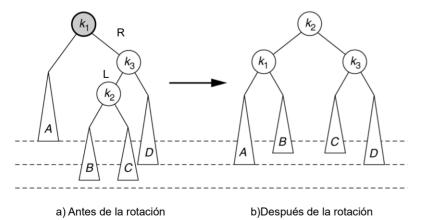
b)Después de la rotación

Algoritmos y Estructuras de Datos

61

61

Rotación doble RL para resolver **QUCU** el caso 3.



Algoritmos y Estructuras de Datos

```
CASO 1 LL
      TElementoAB rotacionLL (TElementoAB k2)
       TElementoAB k1 = k2.hijolzq;
              k2. hijolzq = k1.hijoDer;
              k1. hijoDer = k2;
              return k1;
         }
     CASO 4 RR
    TElementoAB rotacionRR(TElementoAB k1)
     TElementoAB k2 = k1.hijoDer;
            k1. hijoDer = k2.hijolzq;
            k2.hijolzq = k1;
            return k2;
63
   CASO 2 LR
                                                                                     TElementoAB rotacionLR (TElementoAB k3)
           k3. hijolzq = rotacionRR( k3. hijolzq );
           return rotacionLL( k3 );
       }
     CASO 3 RL
     TElementoAB rotacionRL (TElementoAB k1)
            k1. hijoDer = rotacionLL( k1. hijoDer );
            return rotacionRR( k1 );
       }
                                                                                     Ejercicios Arboles AVL
           ¿Qué es un Arbol Binario Balanceado – AVL? Describa las situaciones de
desbalanceo posibles (ilustre con ejemplos) y qué operaciones se deben
aplicar en cada caso para restituir la condición de AVL.
           Defina árbol balanceado. Describa una estructura de datos apta para implementarlo. Desarrolle el algoritmo de búsqueda e inserción en árbol balanceado. Ilustre el funcionamiento mediante un ejemplo. ¿Cuál es el orden del tiempo de ejecución del mismo?
           Insertar en un árbol AVL las siguientes claves, en el orden indicado,
           8, 5, 2, 1, 20, 15, 16, 17,7
mostrando la evolución del árbol. Luego eliminar las clave 15 y 7, mostrando los pasos.
           Muestre el resultado de insertar en un árbol AVL las siguientes claves, en el orden indicado, 2,1,4,5,9,3,6, 7
           mostrando la evolución del árbol. Luego muestre el resultado de eliminar las clave 4, 5 y 9, mostrando la evolución
          Insertar en un árbol AVL las siguientes claves, en el orden indicado 1,88,77,4,5,9,18,56,33,21,45,99,22 mostrando la evolución del árbol. Luego eliminar las clave 18, 77, 1 y 45 mostrando los pasos.
                                      Algoritmos y Estructuras de Datos
```

Ejercicios Arboles AVL	⊕ UCU	
 Weiss – Cuestiones Breves 18.3 y 18.4 Weiss – Problemas teóricos 18.9 18.10 		
Algoritmos y Estructuras de Datos	66	