

Computación y Estructuras Discretas I

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes



2024-2

Agenda del día

1

Árboles n-arios

- Introducción
- Árbol n-ario genérico
- Algoritmos en árboles n-arios

2

Árboles de búsqueda binaria

- Introducción
- Propiedad del árbol de búsqueda binaria
- Consultas de un árbol de búsqueda binaria
- Inserción en un árbol de búsqueda binaria
- Eliminación en un árbol de búsqueda binaria
- Ejercicios

Introducción

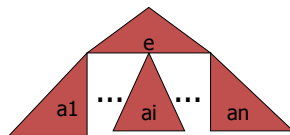
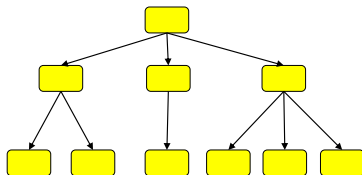
¿Qué es un árbol n-ario?

Introducción

¿Qué es un árbol n-ario?

- Es una estructura recursiva en la que cada elemento puede tener cualquier número de subárboles n-arios asociados.
- Generalización de los árboles binarios.
- En este caso el orden de los subárboles no es importante.
- No es necesario saber cuál sub árbol es el primero o el último, sino simplemente saber que es un subárbol.

Introducción



Formalismo abstracto

Introducción

¿Qué conceptos se extienden de árboles binarios?

Introducción

¿Qué conceptos se extienden de árboles binarios?

- Nodo: elemento del árbol
- Raíz: nodo inicial del árbol
- Hoja: nodo sin hijos
- Camino: nodos entre dos elementos incluyéndolos
- Rama: camino entre la raíz y una hoja
- Altura: número de nodos en la rama más larga
- Peso: número de nodos en el árbol

Introducción

- Orden de un elemento:
 - Número de subárboles asociados
 - Una hoja es un elemento de orden 0
- Orden de un árbol n-ario:
 - Máximo orden de sus elementos

Introducción

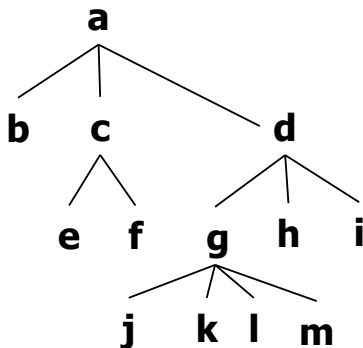
¿Cuál es el algoritmo preorden para un árbol n-ario?

Introducción

¿Cuál es el algoritmo preorden para un árbol n-ario?

- En este recorrido se incluye primero la raíz del árbol.
- Luego se recorren en preorden todos los subárboles que tiene asociados.
- El orden en el que se recorren los elementos del árbol no cumple ninguna propiedad particular.
- Pero se garantiza que en el recorrido va a aparecer una vez cada uno de los elementos del árbol.

Introducción



Introducción

¿Qué es un camino en un árbol n-ario?

Introducción

¿Qué es un camino en un árbol n-ario?

- Un camino entre dos elementos E_1 y E_2 de un árbol n-ario es una secuencia de elementos presentes en el árbol $\langle N_1, \dots, N_k \rangle$ que cumple las siguientes propiedades:
 - 1 $N_1 = E_1$
 - 2 $N_k = E_2$
 - 3 N_i es el padre de N_{i+1} .
- Dicha secuencia no necesariamente existe entre todo par de elementos del árbol.

Introducción

¿Qué pasa si existe un camino entre E_1 y E_2 ?

Introducción

¿Qué pasa si existe un camino entre E_1 y E_2 ?

- Se dice que E_1 es un ancestro de E_2 .
- Se dice que E_2 es un descendiente de E_1 .

Introducción

¿Qué pasa si existe un camino entre E_1 y E_2 ?

- Se dice que E_1 es un ancestro de E_2 .
- Se dice que E_2 es un descendiente de E_1 .

¿Cómo se define la longitud de un camino?

Introducción

¿Qué pasa si existe un camino entre E_1 y E_2 ?

- Se dice que E_1 es un ancestro de E_2 .
- Se dice que E_2 es un descendiente de E_1 .

¿Cómo se define la longitud de un camino?

- Se define como el número de elementos de la secuencia menos 1.

Introducción

¿Qué es una rama?

Introducción

¿Qué es una rama?

- Una rama es un camino que lleva de la raíz a una hoja del árbol.
- Eso quiere decir que en un árbol existe el mismo número de ramas que de hojas.
- La altura de un árbol es la longitud de la rama más larga del árbol más 1.

Introducción

¿Qué diferencia se tiene a la hora de insertar y eliminar con relación a los árboles binarios ordenados?

Introducción

¿Qué diferencia se tiene a la hora de insertar y eliminar con relación a los árboles binarios ordenados?

- Para poder agregar o suprimir un elemento se debe indicar el punto del árbol en el cual se desea realizar la operación.
 - Como no hay un orden, no sabríamos dónde insertar un nuevo elemento.
 - Se debe dar el padre al que le va a insertar o eliminar un nodo hijo.

Agenda del día

1 Árboles n-arios

- Introducción
- Árbol n-ario genérico
- Algoritmos en árboles n-arios

2 Árboles de búsqueda binaria

- Introducción
- Propiedad del árbol de búsqueda binaria
- Consultas de un árbol de búsqueda binaria
- Inserción en un árbol de búsqueda binaria
- Eliminación en un árbol de búsqueda binaria
- Ejercicios

Árbol n-ario genérico

¿Cómo sería la implementación genérica de un árbol n-ario?

Árbol n-ario genérico

¿Cómo sería la implementación genérica de un árbol n-ario?

```
public class NodoArbolNArio<T> {  
    private T elem;  
    private List<NodoArbolNArio<T>> hijos;  
    ...  
}  
  
public class ArbolNArio<T> {  
    private NodoArbolNArio<T> raiz;  
    ...  
}
```


Agenda del día

- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - Inserción en un árbol de búsqueda binaria
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

Algoritmos en árboles n-arios

¿Cómo es la estructura de un algoritmo para árboles n-arios?

Algoritmos en árboles n-arios

¿Cómo es la estructura de un algoritmo para árboles n-arios?

- Similar a árboles binarios.
 - Primer nivel:
 - Trata el caso de un árbol vacío.
 - Delega la responsabilidad.
 - Segundo nivel:
 - Planteamiento recursivo
 - Se tienen n avances posibles en la recursión
 - Se requiere un ciclo para iterar sobre cada avance.

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo de búsqueda?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo de búsqueda?

- Implementamos el algoritmo en el primer nivel
 - En la clase ArbolNArio
- Implementamos el algoritmo en el segundo nivel
 - En la clase NodoArbolNArio

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo en el primer nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo en el primer nivel?

```
public T buscar(T elemento) {  
    if (raiz == null)  
        return null;  
    else  
        raiz.buscar(elemento);  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo en el segundo nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo en el segundo nivel?

```
public T buscar(T elemento) {  
    if (elem.equals(elemento))  
        return elem;  
    else if (esHoja())  
        return null;  
    else {  
        for (int i=0; i<hijos.size(); i++) {  
            T temp = hijos.get(i).buscar(elemento);  
            if (temp != null)  
                return temp;  
        }  
        return null;  
    }  
}  
  
public boolean esHoja() {  
    return (hijos.size() == 0);  
}
```

Algoritmos en árboles n-arios

¿Qué otros algoritmos tenemos?

Algoritmos en árboles n-arios

¿Qué otros algoritmos tenemos?

- Algoritmos para calcular propiedades.
 - Dar el peso
 - Dar la altura
 - ...
- La mayoría de estos algoritmos siguen el mismo esquema de solución
 - Generalización del patrón de descenso recursivo.

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darPeso()` en el primer nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darPeso()` en el primer nivel?

```
public int darPeso() {  
    if (raiz == null)  
        return 0;  
    else  
        raiz.darPeso();  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darPeso()` en el segundo nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darPeso()` en el segundo nivel?

```
public int darPeso() {  
    if (esHoja()) {  
        return 1;  
    }  
    else {  
        int pesoAcum = 1;  
        for (int i=0; i<hijos.size(); i++) {  
            pesoAcum += hijos.get(i).darPeso();  
        }  
        return peso;  
    }  
}  
  
public boolean esHoja() {  
    return (hijos.size() == 0);  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darNroHojas()` en el primer nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darNroHojas()` en el primer nivel?

```
public int darNroHojas() {  
    if (raiz == null)  
        return 0;  
    else  
        raiz.darNroHojas();  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darNroHojas()` en el segundo nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darNroHojas()` en el segundo nivel?

```
public int darNroHojas() {  
    if(esHoja()) {  
        return 1;  
    }  
    else {  
        int numHojas = 0;  
        for(int i=0; i<hijos.size(); i++) {  
            numHojas += hijos.get(i).darNroHojas();  
        }  
        return numHojas;  
    }  
}  
  
public boolean esHoja() {  
    return (hijos.size() == 0);  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darAltura()` en el primer nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darAltura()` en el primer nivel?

```
public int darAltura() {  
    if (raiz == null)  
        return 0;  
    else  
        raiz.darAltura();  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darAltura()` en el segundo nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `darAltura()` en el segundo nivel?

```
public int darAltura() {  
    if (esHoja()) {  
        return 1;  
    }  
    else {  
        int maxAltura = 0;  
        for (int i=0; i<hijos.size(); i++) {  
            NodoArbolNArio<T> hijo = hijos.get(i);  
            int tempAltura = hijo.darAltura();  
            if (tempAltura > maxAltura) {  
                maxAltura = tempAltura;  
            }  
        }  
        return maxAltura+1;  
    }  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `insertarElemento(T padre, T elemento)` en el primer nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `insertarElemento(T padre, T elemento)` en el primer nivel?

```
public void insertarElemento(T elementoPadre, T elemento ) throws
    Exception {
    if (raiz == null) {
        NodoArbolNArio<T> nuevo = new NodoArbolNArio<T>();
        nuevo.setElem(elemento);
        raiz = nuevo;
    }
    else {
        NodoArbolNArio<T> padreTemp = buscarNodo(elementoPadre);
        if (padreTemp == null)
            throw new Exception("Padre no existe");
        else
            padreTemp.insertarElemento(elemento);
    }
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `insertarElemento(T elemento)` en el segundo nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `insertarElemento(T elemento)` en el segundo nivel?

```
public void insertarElemento(T elemento) {  
    NodoArbolNArio<T> nuevo = new NodoArbolNArio<T>();  
    nuevo.setElem(elemento);  
    hijos.add(nuevo);  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `eliminarElemento(T elemento)` en el primer nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `eliminarElemento(T elemento)` en el primer nivel?

```
public void eliminarElemento(T elemento) throws Exception {  
    if (raiz == null)  
        throw Exception("El arbol esta vacio");  
    else if (elemento.equals(raiz.getElem())) {  
        if (raiz.esHoja())  
            raiz = null;  
        else  
            throw new Exception("Imposible eliminar");  
    }  
    else {  
        NodoArbolNArio<T> padre = buscarPadre(elemento);  
        if (padre == null) {  
            throw new Exception("Nodo invalido");  
        }  
        else  
            padre.eliminarElemento(elemento);  
    }  
}
```

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `eliminarElemento(T elemento)` en el segundo nivel?

Algoritmos en árboles n-arios

¿Cómo sería el algoritmo `eliminarElemento(T elemento)` en el segundo nivel?

```
public void eliminarElemento(T elemento) throws Exception {  
    for(int i=0;i<hijos.size();i++) {  
        NodoArbolNArio<T> hijo = hijos.get(i);  
        if(hijo.getElem().equals(elemento) && hijo.esHoja())  
            hijos.eliminar(i)  
        else  
            throw new Exception("No eliminable");  
    }  
}
```

Algoritmos en árboles n-arios

¿Cuáles son los patrones principales para los algoritmos sobre árboles n-arios?

Algoritmos en árboles n-arios

¿Cuáles son los patrones principales para los algoritmos sobre árboles n-arios?

- Descenso recursivo
- Recorrido total

Algoritmos en árboles n-arios

```
if( condición1 )
{
    // Solución directa 1
}


else if( condición2 )
{
    // Solución directa 2
}


else
{
    // Inicialización de las variables en las que se van a
    // acumular/componer las respuestas parciales


    for( int i = 0; i < subarboles.size() && !condicion; i++ )
    {
        Nodo hijo = ( Nodo )subarboles.get( i );


        // Avance de la recursividad sobre el subárbol "hijo"


        // Acumulación/composición de la respuesta
    }
}
```

 El algoritmo tiene una o varias salidas de la recursividad, en las cuales es posible dar una solución directa al problema.

 Puesto que no se conoce a priori el número de subárboles presentes, es necesario avanzar con un ciclo que se desplaza sobre ellos con la variable "i".

 En el esqueleto suponemos que el vector de subárboles se llama "subarboles" y que cada elemento del árbol es de la clase "Nodo".

 Dentro del ciclo hacemos la llamada recursiva sobre cada uno de los subárboles y acumulamos o componemos la respuesta con las obtenidas anteriormente.

 El ciclo puede terminar porque se hizo la llamada recursiva sobre todos los subárboles o porque se cumple una condición que refleja que el problema ya se ha resuelto. Esto último muchas veces se reemplaza por la instrucción de retorno dentro del ciclo.

Algoritmos en árboles n-arios

```
// Inclusión de la raíz en el recorrido total  
  
for( int i = 0; i < subarboles.size( ); i++ )  
{  
    Nodo hijo = ( Nodo )subarboles.get( i );  
    // Recorrido total sobre el subárbol "hijo"  
}
```



El objetivo de los algoritmos que siguen este patrón es recorrer cada uno de los elementos del árbol haciendo una operación cualquiera sobre cada uno de ellos.

Antes de entrar al ciclo, el método incluye el elemento de la raíz en el recorrido.



En el ciclo nos vamos desplazando con la variable "i" sobre cada uno de los subárboles.

Agenda del día

- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - Inserción en un árbol de búsqueda binaria
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

Introducción

¿Por qué son importantes los árboles binarios?

Introducción

¿Por qué son importantes los árboles binarios?

- Operaciones básicas como insertar, borrar y buscar, toman un tiempo proporcional a la altura del árbol.
- Para un árbol binario completo con n nodos, las operaciones básicas toman $\Theta(\log n)$
- Si el árbol se construye como una cadena lineal de n nodos, tomarían $\Theta(n)$

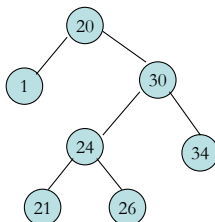
Introducción

¿Qué es un árbol de búsqueda binaria?

Introducción

¿Qué es un árbol de búsqueda binaria?

- Es un árbol binario en el cual se cumple que para cada nodo x :
 - Los nodos del subárbol izquierdo son menores o iguales a x .
 - Los nodos del subárbol derecho son mayores o iguales a x .



Agenda del día

- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - Inserción en un árbol de búsqueda binaria
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

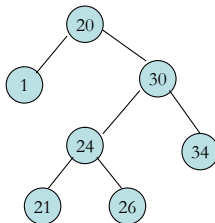
Propiedad del árbol de búsqueda binaria

¿Qué propiedad debe cumplir el árbol de búsqueda binaria?

Propiedad del árbol de búsqueda binaria

¿Qué propiedad debe cumplir el árbol de búsqueda binaria?

- Sea x un nodo del árbol.
- Si y es un nodo en el subárbol izquierdo de x , entonces $key[y] \leq key[x]$.
- Si y es un nodo en el subárbol derecho de x , entonces $key[y] \geq key[x]$.



Propiedad del árbol de búsqueda binaria

¿Cuál es otra característica del árbol de búsqueda binaria?

Propiedad del árbol de búsqueda binaria

¿Cuál es otra característica del árbol de búsqueda binaria?

Si son recorridos en inorden, producen una lista de las claves ordenada ascendentemente.

INORDER-TREE-WALK(x)

if $x \neq \text{nil}$

then INORDER-TREE-WALK(left[x])

print key[x]

INORDER-TREE-WALK(right[x])

Propiedad del árbol de búsqueda binaria

¿Cuál es la complejidad del `INORDER-TREE-WALK (x)` ?

Propiedad del árbol de búsqueda binaria

¿Cuál es la complejidad del `INORDER-TREE-WALK (x)` ?

$\Theta(n)$

Agenda del día

- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - Inserción en un árbol de búsqueda binaria
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

Consultas de un árbol de búsqueda binaria

¿Cuáles son las consultas para un árbol de búsqueda binaria?

Consultas de un árbol de búsqueda binaria

¿Cuáles son las consultas para un árbol de búsqueda binaria?

- Búsqueda de una clave.
- Mínimo
- Máximo
- Sucesor de un nodo
- Predecesor de un nodo.

Consultas de un árbol de búsqueda binaria

¿Cuáles son las consultas para un árbol de búsqueda binaria?

- Búsqueda de una clave.
- Mínimo
- Máximo
- Sucesor de un nodo
- Predecesor de un nodo.

¿Cuál es la complejidad de estas consultas?

Consultas de un árbol de búsqueda binaria

¿Cuáles son las consultas para un árbol de búsqueda binaria?

- Búsqueda de una clave.
- Mínimo
- Máximo
- Sucesor de un nodo
- Predecesor de un nodo.

¿Cuál es la complejidad de estas consultas?

- Cada una de estas operaciones se puede hacer en $O(h)$ donde h es la altura del árbol.

Consultas de un árbol de búsqueda binaria

¿Cómo sería la búsqueda de un nodo con clave k dado un árbol con apuntador a la raíz x ?

Consultas de un árbol de búsqueda binaria

¿Cómo sería la búsqueda de un nodo con clave k dado un árbol con apuntador a la raíz x ?

TREE-SEARCH(x, k)

if $x = \text{nil}$ or $k = \text{key}[x]$

then return x

if $k < \text{key}[x]$

then return TREE-SEARCH(left[x], k)

else return TREE-SEARCH(right[x], k)

Consultas de un árbol de búsqueda binaria

¿Y la misma búsqueda, pero iterativa?

Consultas de un árbol de búsqueda binaria

¿Y la misma búsqueda, pero iterativa?

ITERATIVE-TREE-SEARCH(x, k)

while $x \neq \text{nil}$ and $k \neq \text{key}[x]$

do if $k < \text{key}[x]$

then $x \leftarrow \text{left}[x]$

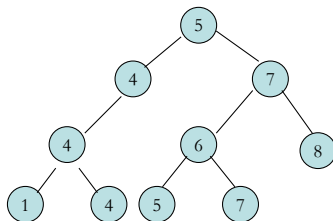
else $x \leftarrow \text{right}[x]$

Consultas de un árbol de búsqueda binaria

¿En un árbol de búsqueda binaria dónde se ubica el elemento mínimo?

Consultas de un árbol de búsqueda binaria

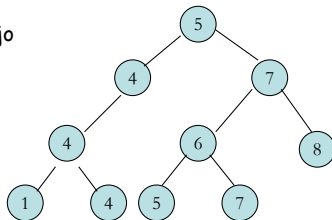
¿En un árbol de búsqueda binaria dónde se ubica el elemento mínimo?



Consultas de un árbol de búsqueda binaria

¿En un árbol de búsqueda binaria dónde se ubica el elemento mínimo?

Idea: seguir los apuntadores al hijo izquierdo desde la raíz hasta que se encuentre nil

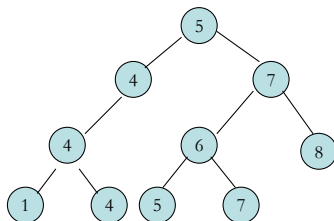


Consultas de un árbol de búsqueda binaria

¿En un árbol de búsqueda binaria dónde se ubica el elemento mínimo?

TREE-MINIMUN(x)

```
while left[x] ≠ nil  
  do x ← left[x]  
return x
```

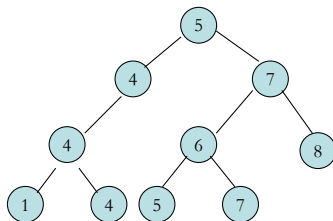


Consultas de un árbol de búsqueda binaria

¿En un árbol de búsqueda binaria dónde se ubica el elemento máximo?

Consultas de un árbol de búsqueda binaria

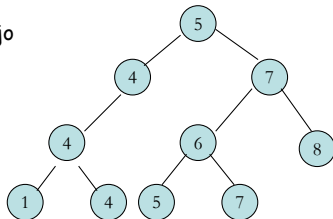
¿En un árbol de búsqueda binaria dónde se ubica el elemento máximo?



Consultas de un árbol de búsqueda binaria

¿En un árbol de búsqueda binaria dónde se ubica el elemento máximo?

Idea: seguir los apuntadores al hijo derecho desde la raíz hasta que se encuentre nil



Consultas de un árbol de búsqueda binaria

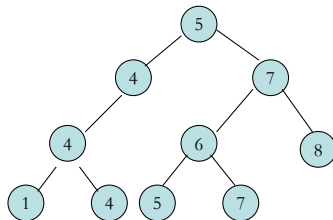
¿En un árbol de búsqueda binaria dónde se ubica el elemento máximo?

TREE-MAXIMUM(x)

while right[x] \neq nil

do $x \leftarrow$ right[x]

return x



Consultas de un árbol de búsqueda binaria

Dado un nodo x donde $key[x] = k$, ¿cuál es el predecesor de x ?

Consultas de un árbol de búsqueda binaria

Dado un nodo x donde $key[x] = k$, ¿cuál es el predecesor de x ?

El predecesor de x es el nodo y tal que $key[y]$ es la mayor clave, menor que $key[x]$.

Consultas de un árbol de búsqueda binaria

Dado un nodo x donde $key[x] = k$, ¿cuál es el sucesor de x ?

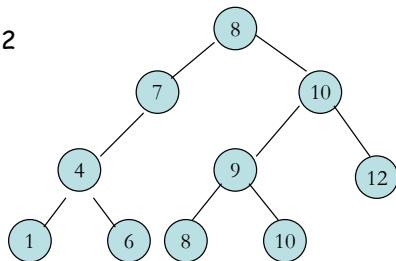
Consultas de un árbol de búsqueda binaria

Dado un nodo x donde $key[x] = k$, ¿cuál es el sucesor de x ?

El sucesor de x es el nodo y tal que $key[y]$ es la menor clave, mayor que $key[x]$.

Consultas de un árbol de búsqueda binaria

Cuál es el sucesor de 7, 9, 10 y 12



Consultas de un árbol de búsqueda binaria

¿Cómo se obtiene el sucesor de x en un árbol de búsqueda binaria?

Consultas de un árbol de búsqueda binaria

¿Cómo se obtiene el sucesor de x en un árbol de búsqueda binaria?

TREE-SUCCESSOR(x)

if $\text{right}[x] \neq \text{nil}$

then return **TREE-MINIMUM**($\text{right}[x]$)

$y \leftarrow p[x]$

while $y \neq \text{nil}$ and $x = \text{right}[y]$

do $x \leftarrow y$

$y \leftarrow p[y]$

return y

Agenda del día

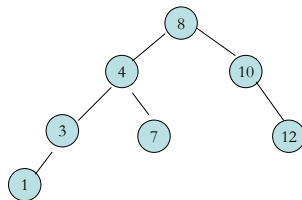
- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - **Inserción en un árbol de búsqueda binaria**
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

Inserción en un árbol de búsqueda binaria

TREE-INSERT(z)

```
y ← nil
x ← root[T]
while x ≠ nil
  do y ← x
    if key[z] < key[x]
      then x ← left[x]
    else x ← right[x]
p[z] ← y
if y = nil
  then root[T] ← z
else if key[z] < key[y]
  then left[y] ← z
else right[y] ← z
```



Explique el código para el caso de **TREE-INSERT**(z), donde $\text{key}[z]=5$

Agenda del día

- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - Inserción en un árbol de búsqueda binaria
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

Eliminación en un árbol de búsqueda binaria

```

TREE-DELETE(z)
  if left[z]=nil or right[z]=nil
    then y ← z
    else y ← TREE-SUCCESSOR(z)
  if left[y] ≠ nil
    then x ← left[y]
    else x ← right[y]
  if x ≠ nil
    then p[x] ← p[y]
  if p[y] = nil
    then root[T] ← x
    else if y = left[p[y]]
      then left[p[y]] ← x
      else right[p[y]] ← x
  if y ≠ z
    then key[z] ← key[y]
  return y

```

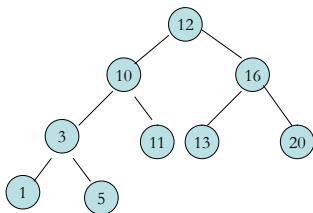
Eliminación en un árbol de búsqueda binaria

Caso 1:

Borrar z y z no tiene hijos.

TREE-DELETE(T, z), donde $\text{key}[z]=5$

Qué se debe hacer?



Eliminación en un árbol de búsqueda binaria

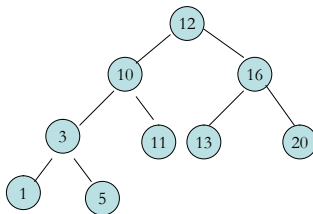
Caso 1:

Borrar z y z no tiene hijos.

TREE-DELETE(T, z), donde $\text{key}[z]=5$

El padre de z debe ahora apuntar a nil

$p[z] \leftarrow \text{nil}$



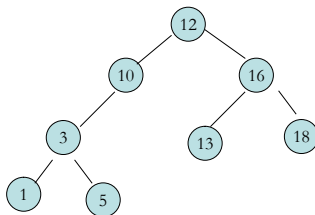
Eliminación en un árbol de búsqueda binaria

Caso 2:

Borrar z y z tiene un solo hijo

TREE-DELETE(T, z), donde $\text{key}[z]=10$

Qué se debe hacer?



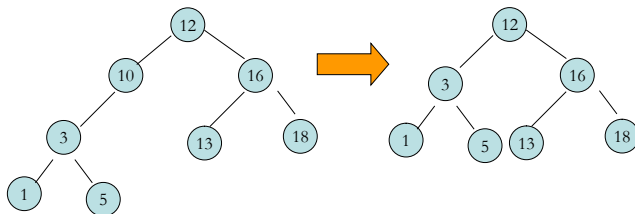
Eliminación en un árbol de búsqueda binaria

Caso 2:

Borrar z y z tiene un solo hijo

$\text{TREE-DELETE}(T, z)$, donde $\text{key}[z]=10$

Se separa z del árbol



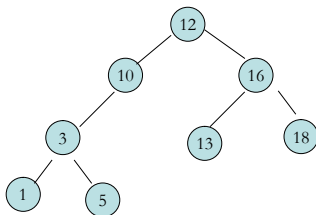
Eliminación en un árbol de búsqueda binaria

Caso 3:

Borrar z y z tiene dos hijos

TREE-DELETE(T, z), donde $\text{key}[z]=12$

Qué se debe hacer?



Eliminación en un árbol de búsqueda binaria

Caso 3:

Borrar z y z tiene dos hijos

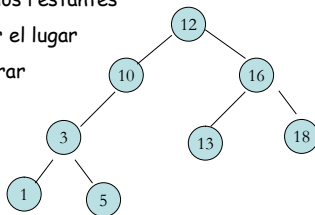
TREE-DELETE(T, z), donde $\text{key}[z]=12$

Qué se debe hacer?

Cuál de los nodos restantes

debería ocupar el lugar

del nodo a borrar



Eliminación en un árbol de búsqueda binaria

Caso 3:

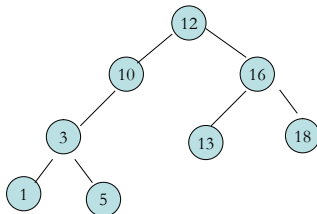
Borrar z y z tiene dos hijos

TREE-DELETE(T, z), donde $\text{key}[z]=12$

Se separa(elimina) su sucesor y del árbol

y se reemplaza su contenido con el

de z



Eliminación en un árbol de búsqueda binaria

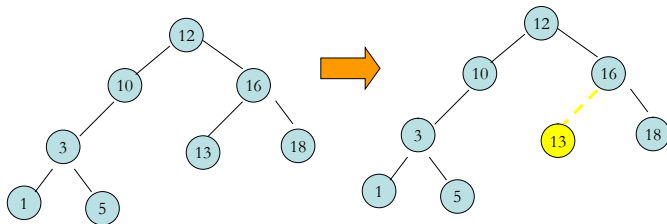
Caso 3:

Borrar z y z tiene dos hijos

TREE-DELETE(T, z), donde $\text{key}[z]=12$

Se separa(elimina) su sucesor y del árbol

y se reemplaza su contenido con el
de z



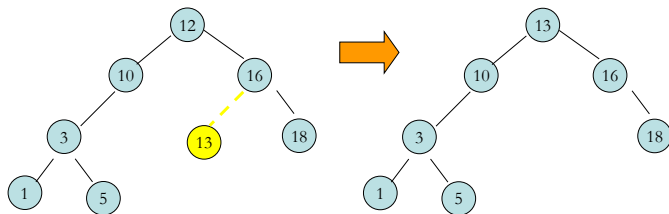
Eliminación en un árbol de búsqueda binaria

Caso 3:

Borrar z y z tiene dos hijos

TREE-DELETE(T, z), donde $\text{key}[z]=12$

Se separa(elimina) su sucesor y del árbol
y se reemplaza su contenido con el
de z



Agenda del día

- 1 **Árboles n-arios**
 - Introducción
 - Árbol n-ario genérico
 - Algoritmos en árboles n-arios

- 2 **Árboles de búsqueda binaria**
 - Introducción
 - Propiedad del árbol de búsqueda binaria
 - Consultas de un árbol de búsqueda binaria
 - Inserción en un árbol de búsqueda binaria
 - Eliminación en un árbol de búsqueda binaria
 - Ejercicios

Ejercicios

Ejercicio

En grupos de máximo 5 integrantes por breakout room deben:

- *Implementar un árbol n -ario genérico.*
- *Implementar un árbol binario de búsqueda genérico.*