Árboles

Árboles ordenados

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

24 de julio de 2021

Árboles binarios ordenados

1. Árboles binarios ordenados

2. Árboles B

Temas

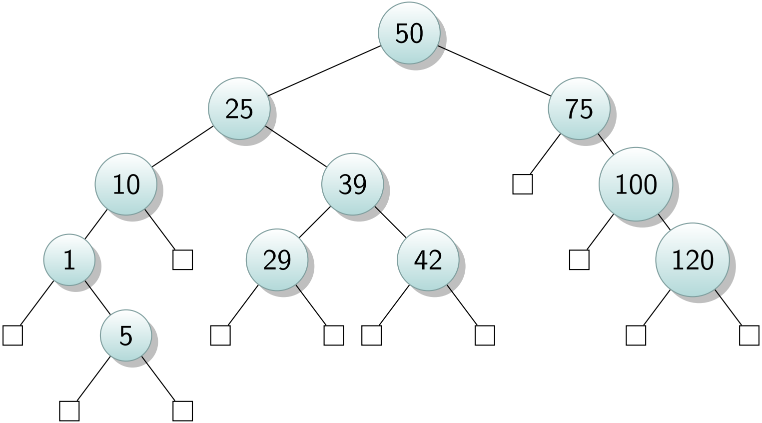
1. Árboles binarios ordenados
   - Definición
   - Insertar
   - Remover
   - Complejidades

Árboles binarios ordenados

En un *árbol binario ordenado* los elementos almacenados en cada nodo obedecen la siguiente condición:
Dado un nodo raíz r, su subárbol izquierdo T_L y su subárbol derecho T_R:

$$\forall x \in T_L, x \leqslant r$$
$$\forall x \in T_R, x > r$$

Árboles binarios ordenados



Pregunta: ¿Qué tipo de recorrido imprimirá todos los números en orden?

Temas

1. Árboles binarios ordenados
   - Definición
   - Insertar
   - Remover
   - Complejidades

Algoritmo para insertar un nodo

Algoritmo 1 Inserta al árbol.

```
function AGREGA(árbol, dato)
    if esVacío(árbol) then
        árbol.raíz ← new NODO(dato)
    else
        AGREGA(árbol.raíz, dato)
```

Algoritmo 2 Inserta por nodo.

```
function AGREGA(nodo, dato)
    if dato ⩽ nodo.dato then
        if nodo.hijoI < ∅ then
            nodo.hijoI ← new NODO(dato)
        else
            AGREGA(nodo.hijoI, dato)
    else
        Lo mismo, pero del lado derecho.
```

Ejercicio de inserción

Insertar: 50,25,10,39,29,75,1,5,100,42,120

Temas

1. Árboles binarios ordenados
   - Definición
   - Insertar
   - Remover
   - Complejidades

Algoritmo para remover un dato

Algoritmo 3 Remover dato.

```
function REMUEVE(árbol, dato)
    nodo ← ENCUENTRA(árbol, dato)
    if nodo = ∅ then return
    while ¬ ESHOJA(nodo) do
        if nodo.hijoI ≠ ∅ then
            nodoN ← MAYOR(nodo.hijoI)
        else                                    ▷ nodo.hijoD ≠ ∅
            nodoN ← MENOR(nodo.hijoD)
        INTERCAMBIA(nodo.dato, nodoN.dato)
        nodo ← nodoN
    DESCONECTA(nodo)
```

Temas

1. Árboles binarios ordenados
   - Definición
   - Insertar
   - Remover
   - Complejidades

Complejidades

Insertar Caso promedio: $\mathcal{O}(\log(n))$, peor caso: $\mathcal{O}(n)$, ¿mejor caso $\mathcal{O}(1)$?

Buscar Caso promedio: $\mathcal{O}(\log(n))$, peor caso: $\mathcal{O}(n)$, mejor caso $\mathcal{O}(1)$.

Remover Caso promedio: $\mathcal{O}(\log(n))$, peor caso: $\mathcal{O}(n)$.

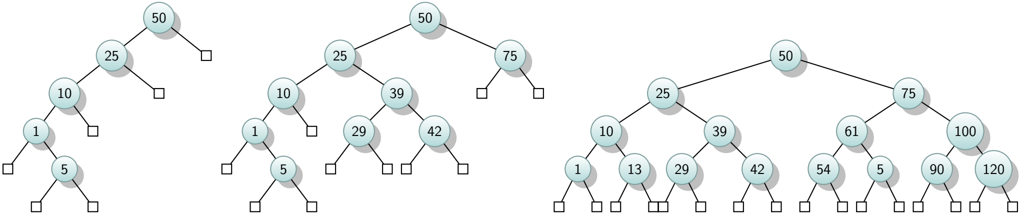


Figura: **Izquierda:** Árbol degenerado. **Medio:** árbol desbalanceado. **Derecha:** árbol completo.

Árboles B

1. Árboles binarios ordenados

2. Árboles B

Árboles B

Un árbol B tiene las propiedades siguientes Cormen y col. 2009:

1. Cada nodo x tiene los campos siguientes:
   1. $n[x]$, el número de llaves almacenadas actualmente en el nodo x,
   2. las $n[x]$ llaves mismas, almacenadas en orden no descendiente, de tal modo que $llave_1[x] \leqslant llave_2[x] \leqslant ...llave_{n[x]}[x]$,
   3. $hoja[x]$, un valor booleano que es $verdadero$ si x es una hoja y $falso$ si x es un nodo interno.
2. Cada nodo x también contiene $n[x] + 1$ referencias $c_1[x], c_2[x], ..., c_{n[x]+1}[x]$ a sus hijos. Los nodos hoja no tienen hijos, por lo que sus campos c_i está indefinidos.

Árboles B II

3. Las llaves $llave_i[x]$ separan los rangos de llaves almacenadas en cada subárbol: si k_i es cualquier llave almacenada en el subárbol con raíz $c_i[x]$, entonces

$$k_1 \leqslant llave_1[x] \leqslant k_2 \leqslant llave_2[x] \leqslant \ldots \leqslant llave_{n[x]}[x] \leqslant k_{n[x]+1}. \tag{1}$$

4. Todas las hojas tienen la misma profundidad, que es el altura del árbol h.

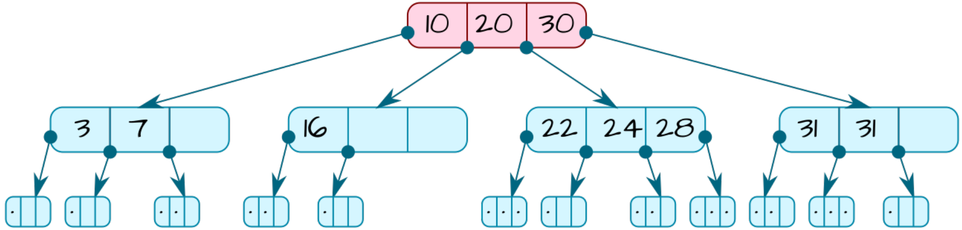


Figura: Árbol B con $t = 2$. Cada nodo contiene máximo 3 llaves y mínimo 1.

Árboles B III

5. Existen cotas inferiores y superiores al número de llaves que puede contener cada nodo. Estas cotas se pueden expresar en términos de un entero fijo $t \geqslant 2$ denominado el *grado mínimo* del árbol-B:
   1. Cada nodo, aparte de la raíz, debe tener al menos $t - 1$ llaves. Cada nodo interno, aparte de la raíz, tiene por lo tanto al menos t hijos. Si el árbol no está vacío, la raíz debe tener al menos una llave.
   2. Cada nodo puede contener máximo $2t - 1$ llaves. Por lo tanto, un nodo interno puede tener a lo más $2t$ hijos. Decimos que un nodo está $lleno$ si contiene exactamente $2t - 1$ llaves.

Bibliografía I

- Cormen, Thomas H. y col. (2009). *Introduction to Algorithms*. 3rd. The MIT Press.
- Preiss, Bruno (1999). *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. John Wiley & Sons.
- Vargas Villazón, América, Jorge Lozano Moreno y Guillermo Levine Gutiérrez, eds. (1998). *Estructuras de datos y Algoritmos*. John Wiley & Sons, 438 pp.

Licencia