

Árboles

Emmanuel Cruz Hernández
`emmanuel_cruzh@ciencias.unam.mx`

6 de diciembre de 2021

- 1 Árboles generales
 - Operaciones
 - Algoritmos básicos
- 2 Árboles binarios
 - Operaciones
- 3 Árboles binarios de búsqueda
 - Operaciones
- 4 Árboles binarios completos
- 5 Bibliografía

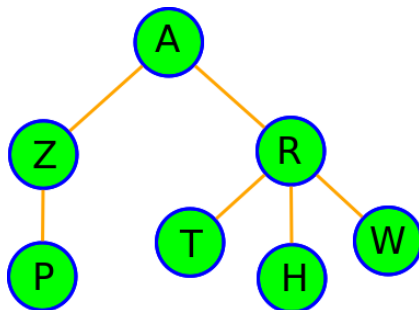
Un árbol es una estructura de datos que organiza sus datos de forma jerárquica

Cuenta con las siguientes características:

- Dinámica
- Homogénea
- No lineal
- Secuencial

Origen del concepto

La terminología principal que se da a esta estructura tiene origen de la estructura de los árboles genealógicos, en los cuales existen los conceptos de *padre*, *hijos*, *abuelos*, etc.



Definición formal

Un árbol T es un conjunto de nodos que almacenan elementos de forma que los nodos tienen una relación padre-hijo que satisface las siguientes propiedades:

- Si T no está vacío, existe un nodo raíz que no tiene padre.
- Cada nodo v de T diferente de la raíz tiene un nodo padre único w ; cada nodo con padre de w es hijo de w .
- Un árbol vacío es un árbol con raíz nula.

- Dos nodos que son hijos del mismo padre son hermanos.
- Un nodo v es externo si v no tiene hijos. Los nodos externos también se conocen como hojas.
- Un nodo v es interno si tiene uno o más hijos.

Operaciones sobre árboles generales

- $\text{root}()$: devuelve la raíz del árbol.
- $\text{parent}(v)$: devuelve el nodo padre del nodo v .
- $\text{children}(v)$: devuelve una lista con los hijos del nodo v . Devuelve la lista vacía si v es una hoja.
- $\text{isRoot}(v)$: verifica si el nodo v es la raíz del árbol.
- $\text{isLeaf}(v)$: verifica si v es una hoja del árbol.
- $\text{isInternal}(v)$: verifica si v es un nodo interno del árbol.

Operaciones de árboles

Se pueden consultar las operaciones de forma visual, en el siguiente enlace:

<https://docs.google.com/presentation/d/17jSxTDS46WZ7sUvJU57t4wuH-w0jqfpCvXsJhvMMfE/edit?usp=sharing>

Para obtener información específica sobre un árbol se requiere de ciertos algoritmos para obtener el resultado. Algunas de estas operaciones son:

- Profundidad de un nodo.
- Altura de un nodo.
- Recorrido en árboles.
 - Preorden
 - Posorden

Profundidad y altura de un nodo

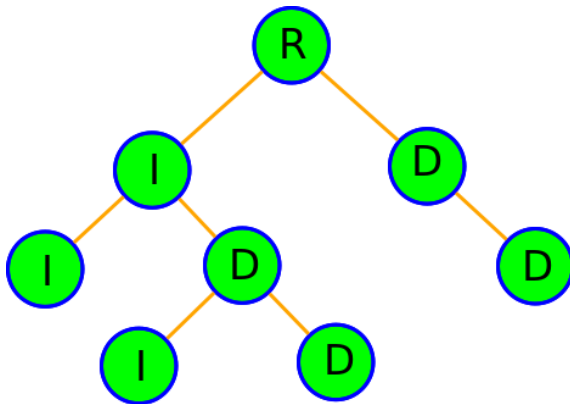
- Profundidad(p): Es la cantidad de ramas que hay entre el nodo raíz y el nodo p .
- Altura(p): Es la cantidad de ramas en el camino más largo que hay entre el nodo p y una hoja.

- Preorden(p):
 - 1 Se visita el nodo p .
 - 2 Para cada hijo c en el nodo p , se aplica preorden.
- Posorden(p):
 - 1 Para cada hijo c en el nodo p , se aplica posorden.
 - 2 Se visita el nodo p .

Un árbol binario es un árbol con ciertas restricciones que se muestran a continuación.

- Cada nodo tiene a lo más dos hijos.
- Cada hijo de un nodo es categorizado como hijo derecho o hijo izquierdo.
- Un hijo izquierdo precede a un hijo derecho en el orden de hijos de un nodo.

Representación gráfica



Operaciones sobre árboles binarios

- $\text{addRoot}(e)$: crea una nueva raíz para el árbol, almacenando el elemento e . Ocurre un error si el árbol no es vacío.
- $\text{addLeft}(p, e)$: crea un nuevo hijo izquierdo en el nodo p , almacenando el elemento e . Ocurre un error si p ya tiene hijo izquierdo.

Operaciones sobre árboles binarios

- $\text{addRight}(p, e)$: crea un nuevo hijo derecho en el nodo p , almacenando el elemento e . Ocurre un error si p ya tiene hijo derecho.
- $\text{set}(p, e)$: reemplaza el elemento almacenado en el nodo p por el elemento e .
- $\text{remove}(p)$: remueve el nodo p , reemplazándolo por un hijo de p (si tiene). Ocurre un error si p tiene dos hijos.

Operaciones sobre árboles binarios

Se pueden consultar las operaciones de forma visual, en el siguiente enlace:

https://docs.google.com/presentation/d/1lfZCcZ8GDtMUKSCcysfG8ShojkW82XKHNB_zmEwmFyQ/edit?usp=sharing

Complejidad en tiempo

Método	Complejidad
<code>addRoot(e)</code>	$O(1)$
<code>addLeft(p, e)</code>	$O(1)$
<code>addRight(p, e)</code>	$O(1)$
<code>set(p, e)</code>	$O(1)$
<code>remove(p)</code>	$O(1)$

- Preorden:
 - 1 Se visita el nodo raíz.
 - 2 Se aplica preorden del hijo izquierdo.
 - 3 Se aplica preorden del hijo derecho.
- Inorden:
 - 1 Se aplica inorden del hijo izquierdo.
 - 2 Se visita el nodo raíz.
 - 3 Se aplica inorden del hijo derecho.
- Posorden:
 - 1 Se aplica posorden del hijo izquierdo.
 - 2 Se aplica posorden del hijo derecho.
 - 3 Se visita el nodo raíz.

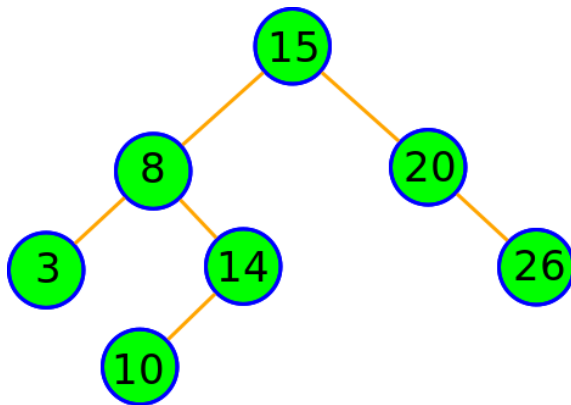
Árboles binarios de búsqueda

Sea S un conjunto cuyos elementos únicos tienen una relación de orden.

Un árbol de búsqueda binaria para S es un árbol binario propio T tal que, para cada nodo interno p de T :

- El nodo p almacena un elemento de S , denotado como $e(p)$.
- Los elementos almacenados en el subárbol izquierdo de p son menores a $e(p)$.
- Los elementos almacenados en el subárbol derecho de p son mayores a $e(p)$.

Representación gráfica



Operaciones sobre árboles binarios de búsqueda

- $\text{retrieve}(k)$: recupera el objeto con clave k o nulo si ni existe tal objeto.
- $\text{insert}(e, k)$: inserta e en el árbol binario.
- $\text{delete}(k)$: elimina y regresa el objeto con la clave k . Manda un error si no existe el objeto.
- $\text{findMin}(p)$: regresa el objeto con clave mínima en el nodo p .
- $\text{finMax}(p)$: regresa el objeto con clave máxima en el nodo p .

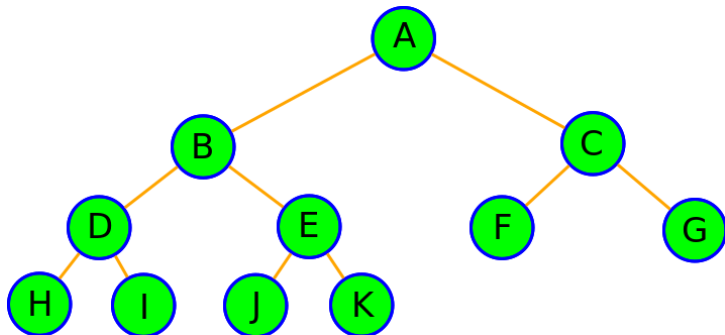
Operaciones sobre árboles binarios de búsqueda

Se pueden consultar las operaciones de forma visual, en el siguiente enlace:

<https://docs.google.com/presentation/d/1HmNzesj-fDTbKLVSRg9mb0oZ5ZDH4IkSAP4J3SYP184/edit?usp=sharing>

Un árbol binario completo es un árbol binario que está completamente lleno, con la posible excepción del nivel inferior que se llena de izquierda a derecha.

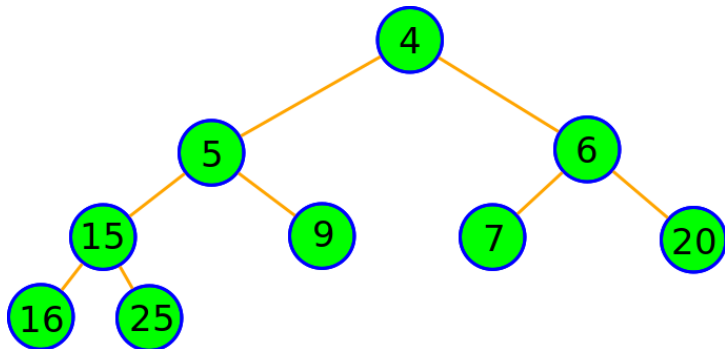
Representación gráfica



Heaps minimales

Un heap minimal es un árbol completo que cumple con la característica de tener todos los elementos menores en los niveles superiores del árbol.

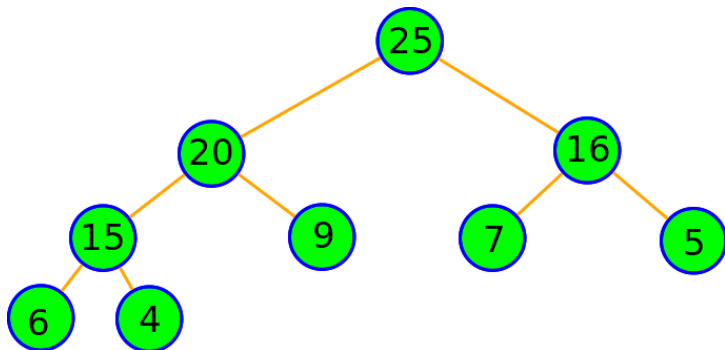
Representación gráfica



Heaps maximales

Un heap maximal es un árbol completo que cumple con la característica de tener todos los elementos mayores en los niveles superiores del árbol.

Representación gráfica






Operaciones sobre heaps minimales

Se pueden consultar las operaciones de forma visual, en el siguiente enlace:

<https://docs.google.com/presentation/d/12k0wiCt-YC9C9JvBw8bmDc0SjYr0G8gjAxfZDZ3hB9o/edit?usp=sharing>

Método	Complejidad
<code>insert(<i>e</i>, <i>k</i>)</code>	$O(\log n)$
<code>remove(<i>k</i>)</code>	$O(\log n)$

-  GOODRICH, M.T., TAMASSIA, R. Y GOLDWASSER, M.H., *Data Structures and Algorithms in Java*, Wiley, Sexta Edición, 2014.
-  WEISS, M. A., *Data Structures and Algorithm Analysis in Java*, Pearson, Tercera Edición, 2012.
-  M. ZERÓN, C., *Notas de Estructuras de Datos 2018-1*