Árbol binario de búsqueda

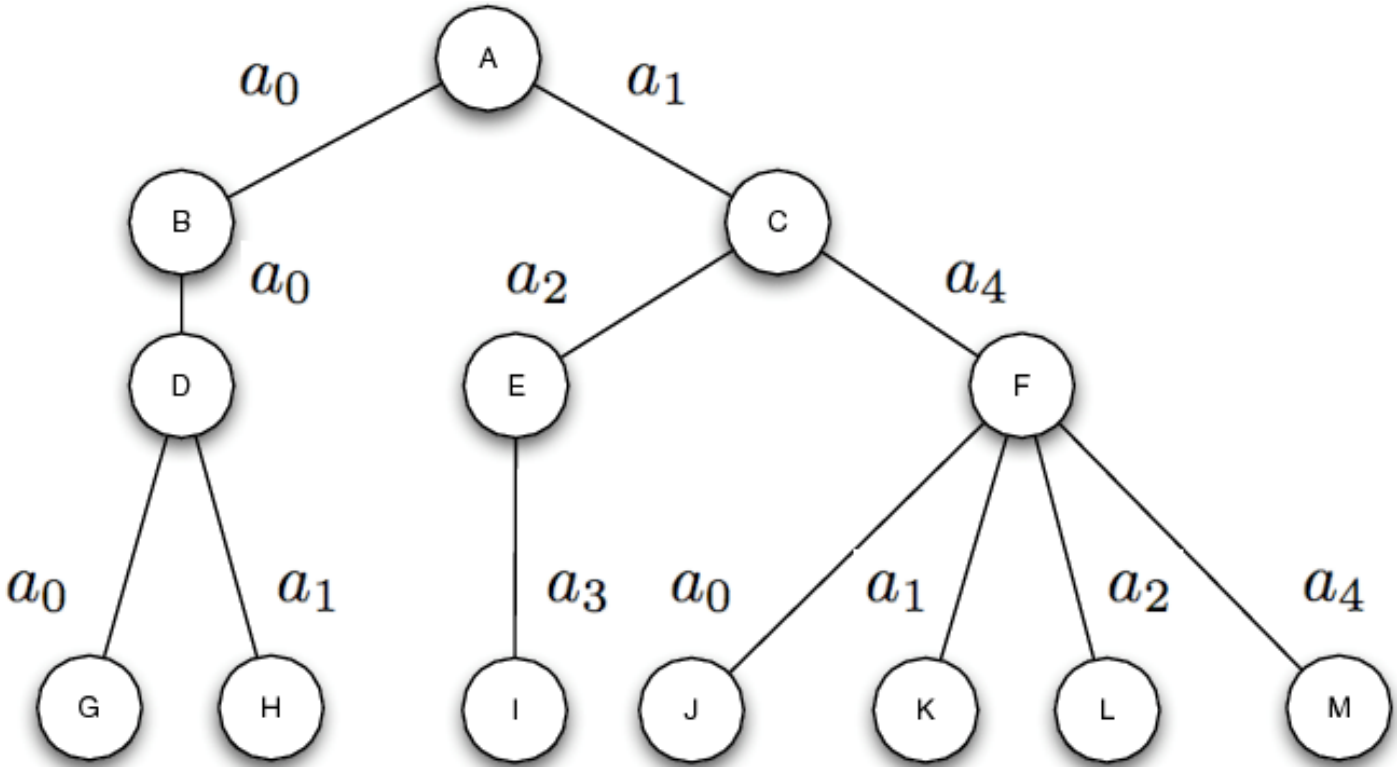
(BST)

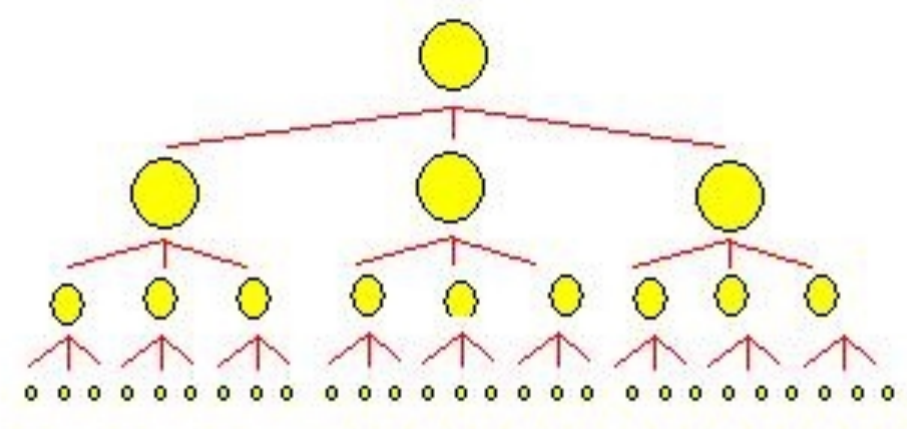
Arboles

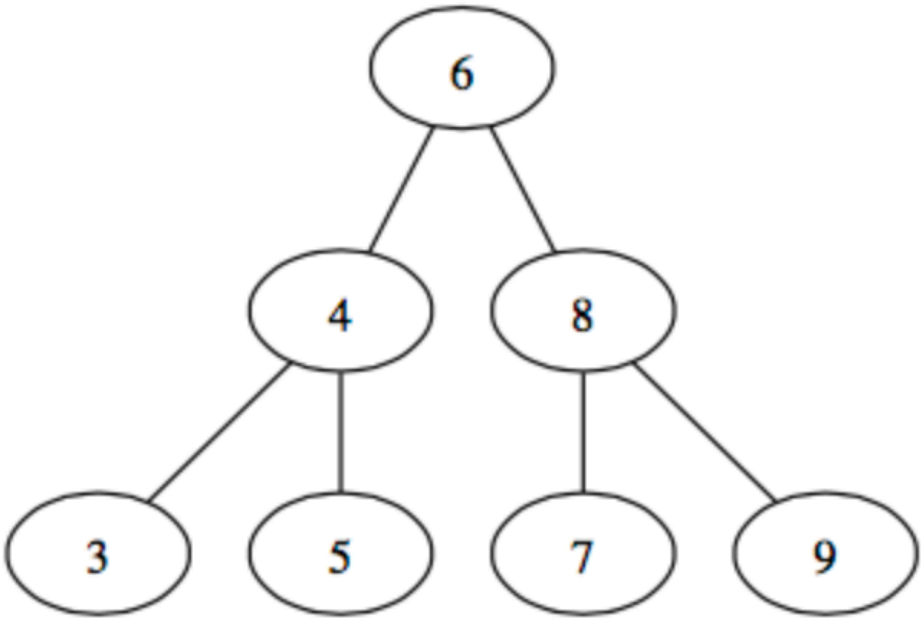
Arboles.

En general, en un árbol, un nodo puede tener cualquier número de hijos.

Sin embargo, un nodo, solo puede tener un padre.


A
B
C
D
E
F
G
H
I
J
K
L
M
a_0
a_1
a_0
a_2
a_4
a_0
a_1
a_3
a_0
a_1
a_2
a_4

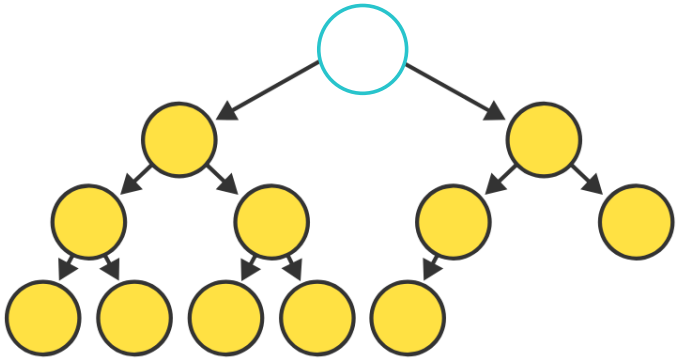



6
4
8
3
5
7
9

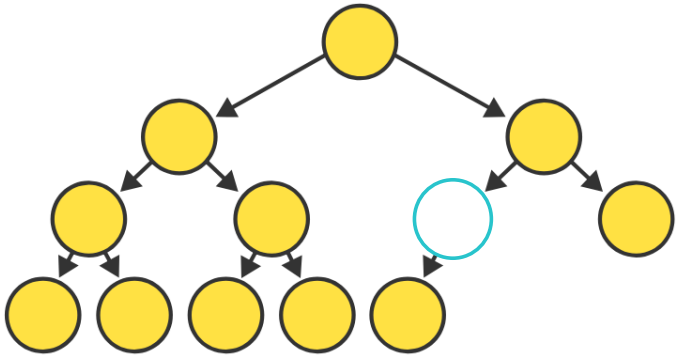
Factor de ramificación (branching factor)

El número de hijos de un nodo es conocido como: factor de ramificación

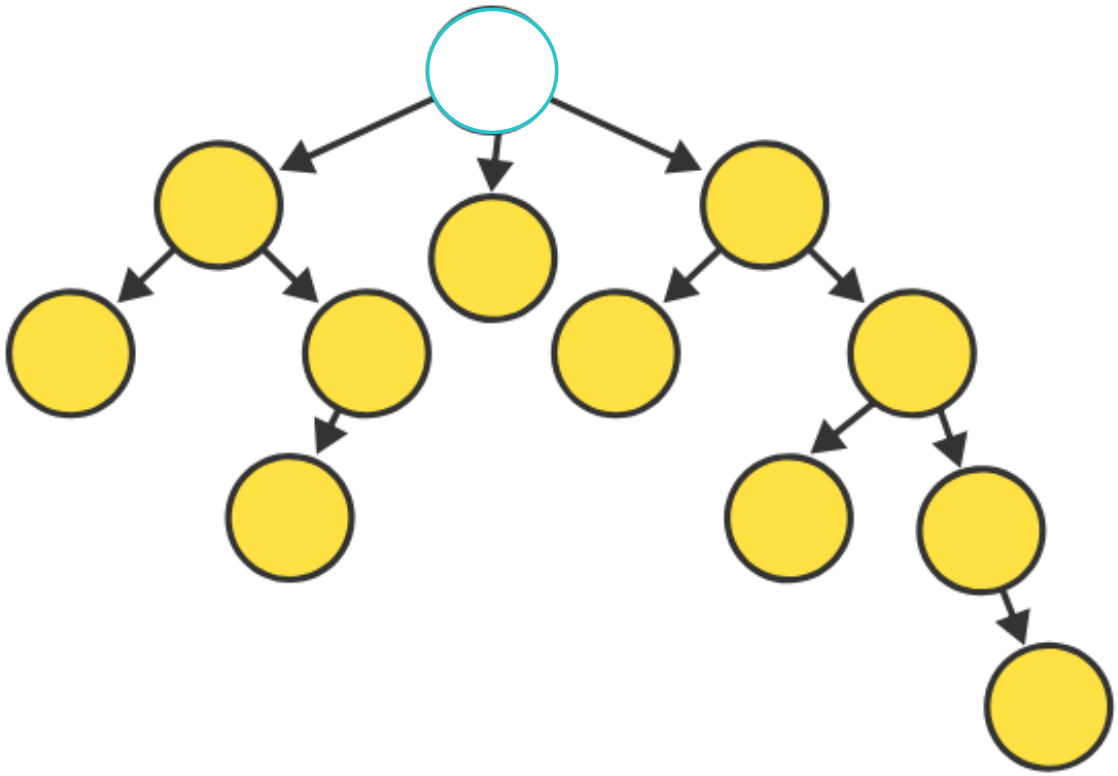
¿Cuál es el factor de ramificación del nodo marcado?



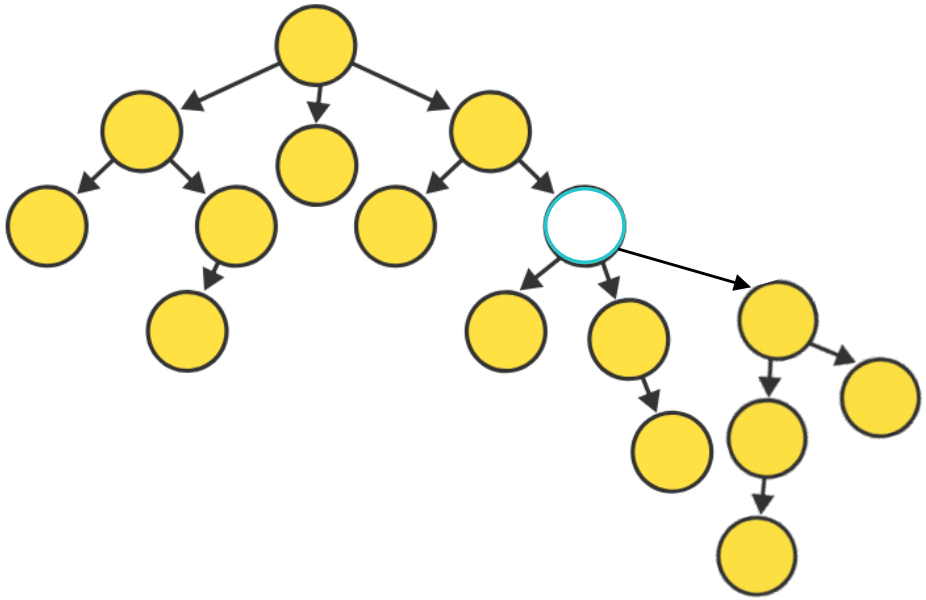
¿Cuál es el factor de ramificación del nodo marcado?



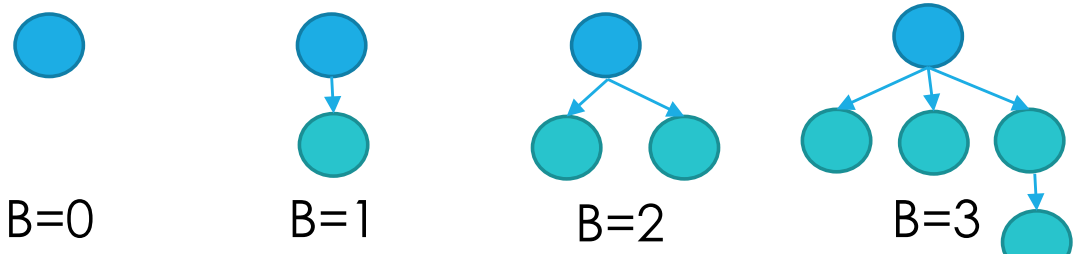
¿Cuál es el factor de ramificación del nodo marcado?

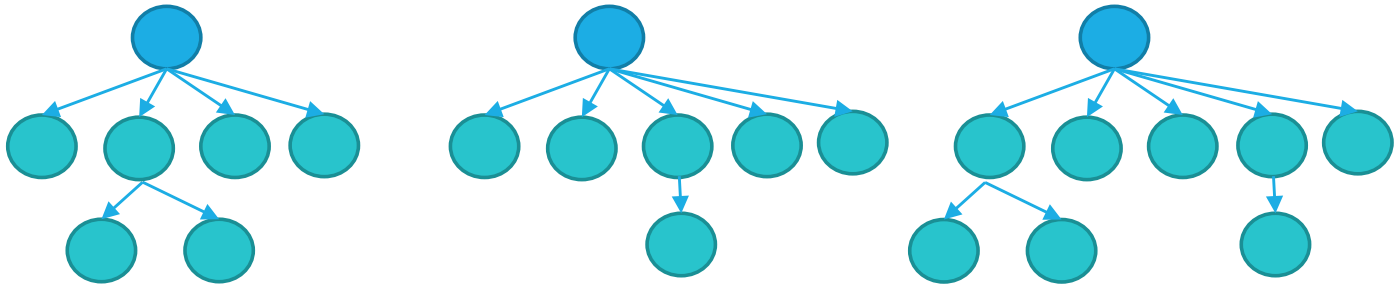


¿Cuál es el factor de ramificación del nodo marcado?



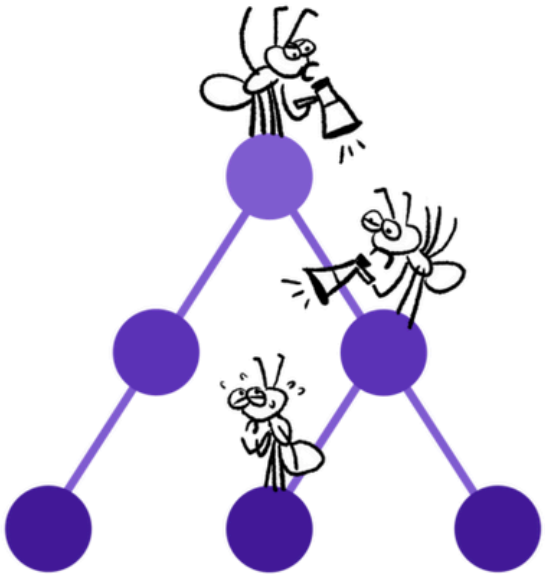
Factor de ramificación





En un árbol binario

El factor de ramificación máximo es 2



Conceptos

Profundidad

Altura de nodo

Altura de árbol

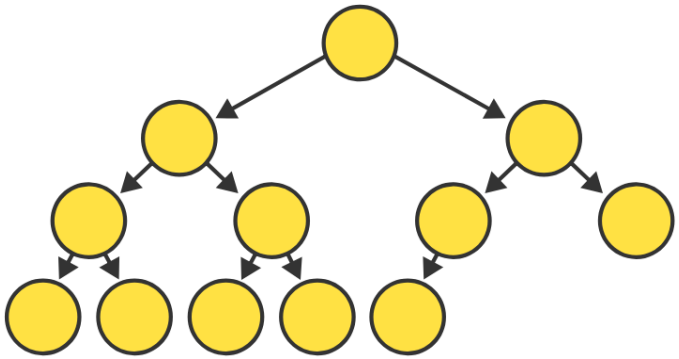
Profundidad

Es una propiedad de cada nodo de un árbol.

Es el numero de enlaces (la distancia) desde la raíz al nodo.

La raíz tiene profundidad 0

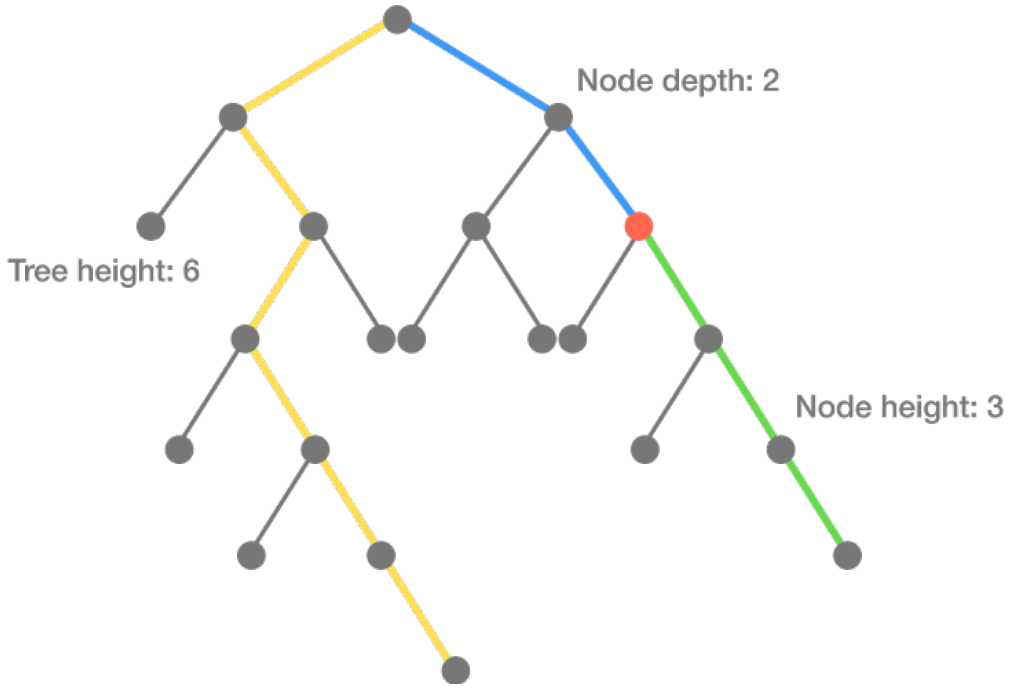
Sus hijos 1

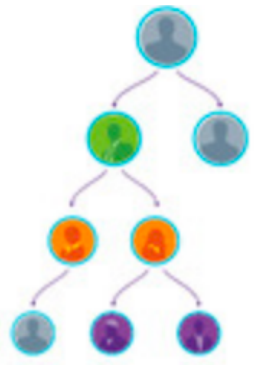


Altura de un árbol

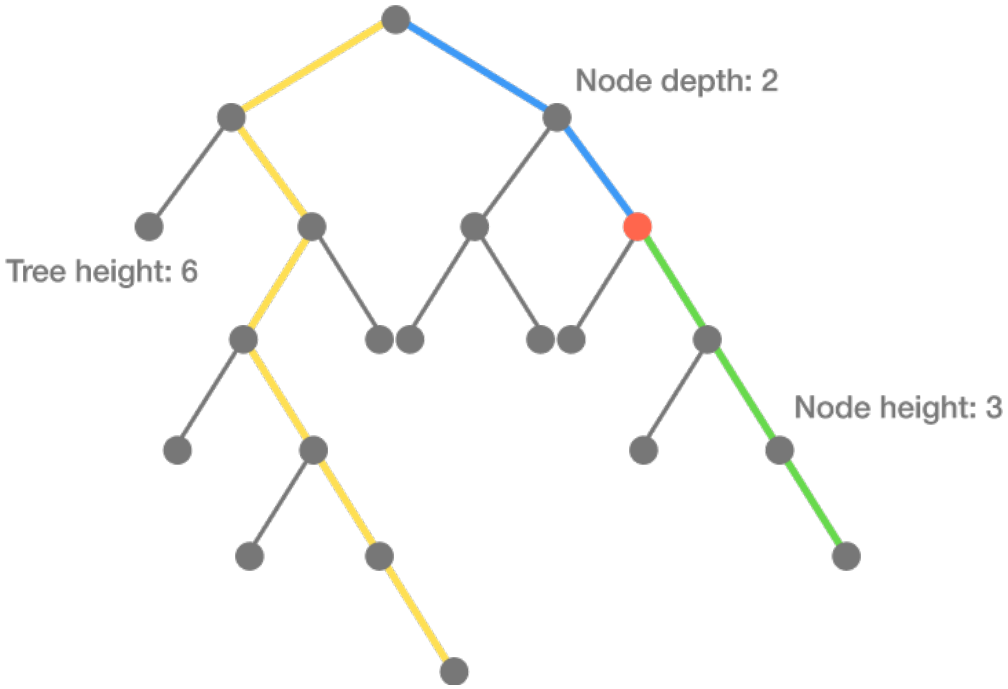
La altura de un árbol es la profundidad de su nodo mas profundo

La altura de un nodo es la mayor cantidad de enlaces de ese nodo a una hoja.





¿Cuál es el máximo numero de nodos que un ab de altura 5 puede tener?

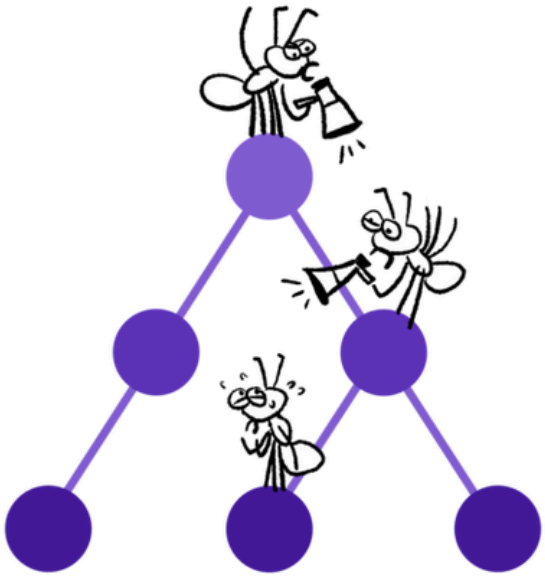


63 nodos

Altura	Nodos en el nivel	Nodos total
0	1	1
1	2	3
2	4	7
3	8	15
4	16	31
5	32	63
6	64	127
7	128	255

ABB - BST

Permite representar los datos de forma jerárquica y ordenada.



Objetivo

Encontrar un elemento *X* en tiempos sublineales a *m*

m= número de elementos

Características:

Un ABB es un AB que cumple con la propiedad de orden

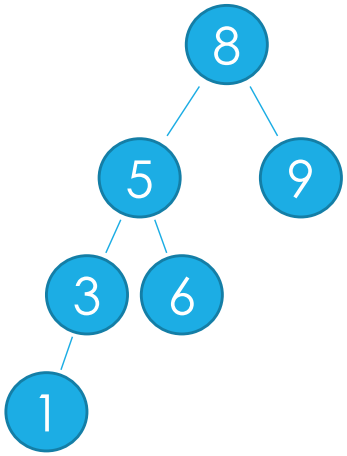
Propiedad de orden:

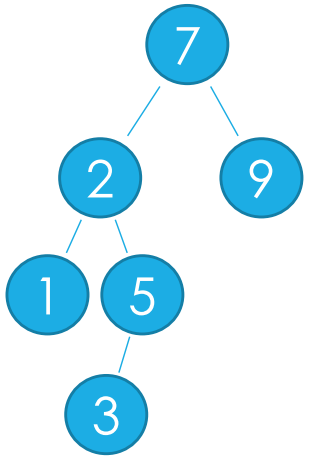
Todos los datos del subárbol izquierdo son **menores** o iguales que la raíz.

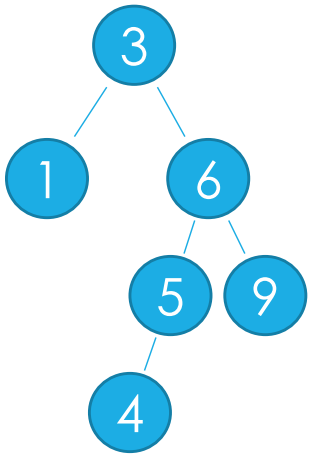
Todos los datos del subárbol derecho son **mayores** que la raíz.

Los subárboles izquierdo y derecho también son ABB

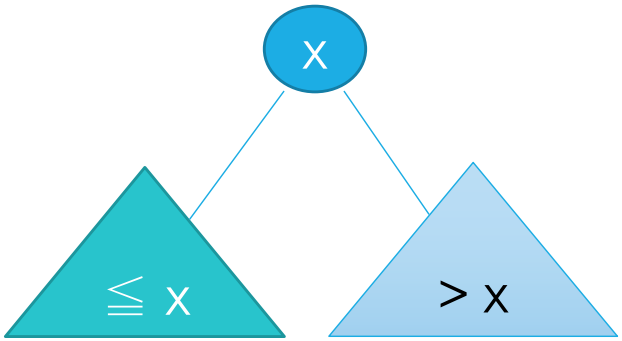
Ejemplo







De forma general…



Definición formal

Para todo nodo T del árbol debe cumplirse que todos los valores de los nodos del subárbol izquierdo de T deben ser menores o iguales al valor del nodo T.

De forma similar, todos los valores de los nodos el subárbol derecho de T deben ser mayores o iguales al valor del nodo T

Propiedades

Cada nodo tiene máximo dos hijos.

Todo nodo tiene un padre, excepto la raíz.

Los nodos del sub árbol izquierdo son menores (o iguales) a la raíz

Los nodos del sub árbol derecho son mayores a la raíz.

El sucesor de un nodo es el **menor** de todos aquellos que son **mayores** al nodo a borrar

Propiedades

Un recorrido in-orden por el árbol recorre los elementos en orden de menor a mayor.

El elemento mínimo es el primer nodo sin hijo izquierdo en un descenso por hijos izquierdos desde la raíz.

El elemento máximo es el primer nodo sin hijo derecho en un descenso por hijos derechos desde la raíz.

Para buscar un elemento se parte de la raíz y se desciende escogiendo el subárbol izquierdo si el valor buscado es menor que el del nodo o el subárbol derecho si es mayor.

Para insertar un elemento se busca en el árbol y se inserta como nodo hoja en el punto donde debería encontrarse.

Para borrar un elemento, se adaptan los enlaces si tiene 0 o 1 hijo. Si tiene dos hijos se intercambia por su sucesor.

Altura de un árbol de búsqueda

Si se entiende como ***altura de un nodo***:

la longitud del camino más largo que
comienza en el nodo y termina en una hoja

La altura de un árbol se define como:

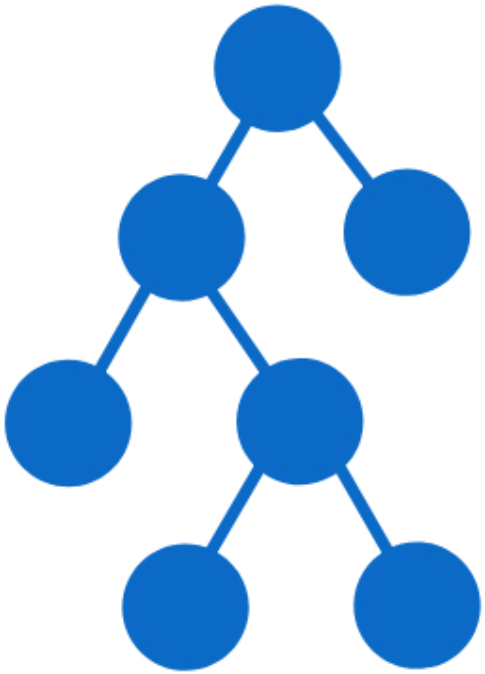
la altura de la raíz.

La altura de un árbol determina:

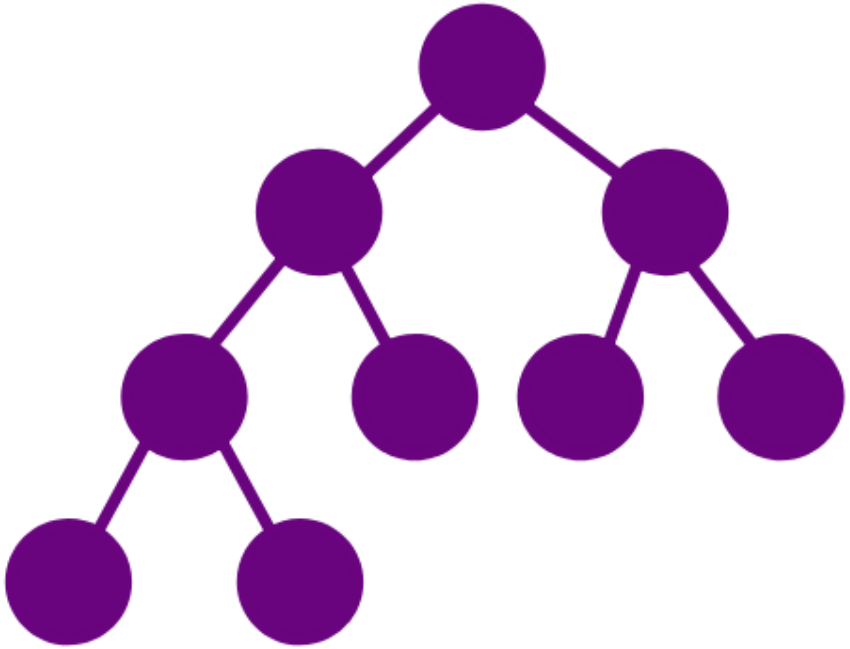
la **eficiencia** de la mayoría de las operaciones definidas sobre él.

En un árbol binario de búsqueda la principal estrategia para minimizar la altura de un árbol es:

Elegir la **mediana** como raíz.



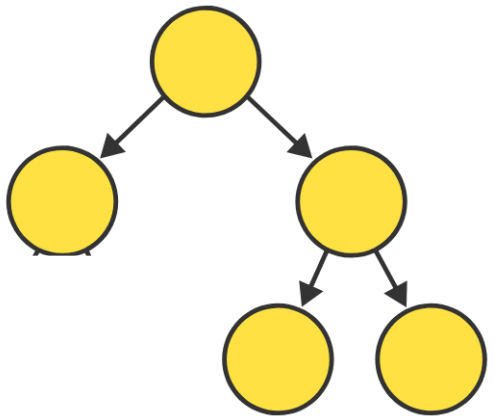
Árbol Lleno



Árbol Completo

Árbol lleno

Un árbol binario lleno, es aquel en que cada nodo, tiene exactamente cero o dos hijos.

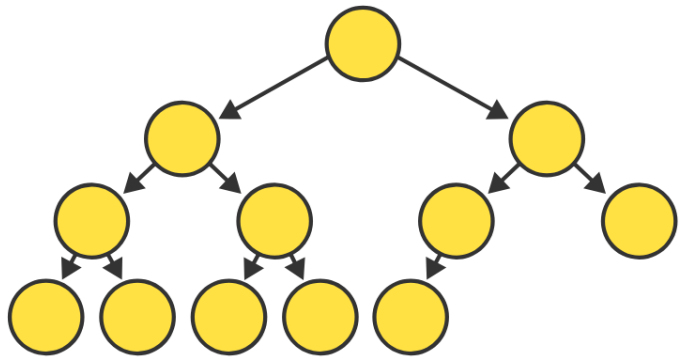


Árbol completo

Un árbol binario completo, es aquel en que todos los niveles (a excepción del último), están llenos (todos los nodos tienen dos hijos).

En el último nivel los nodos están lo mas a la izquierda posible (se llena de izquierda a derecha)

Un árbol puede estar lleno sin estar completo, y un árbol puede estar completo sin estar lleno.

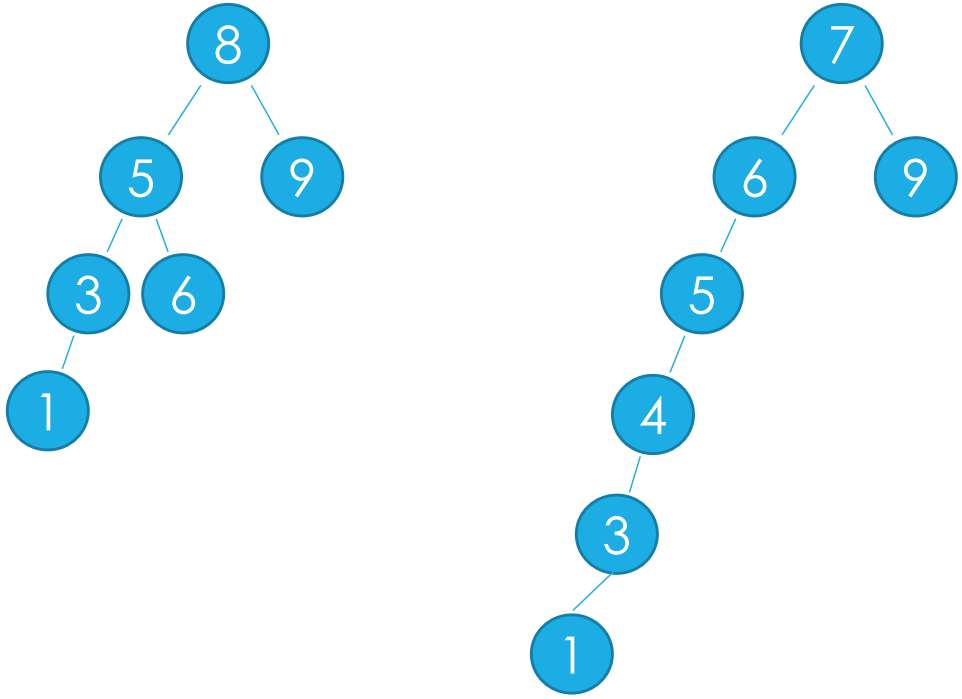


CONSIDERAR QUE:

La construcción del ABB determina su **forma**

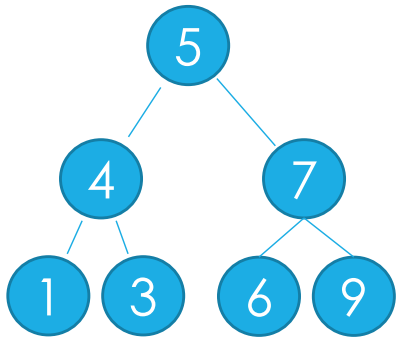
La forma del ABB afecta **directamente** el costo de las operaciones

Ejemplo

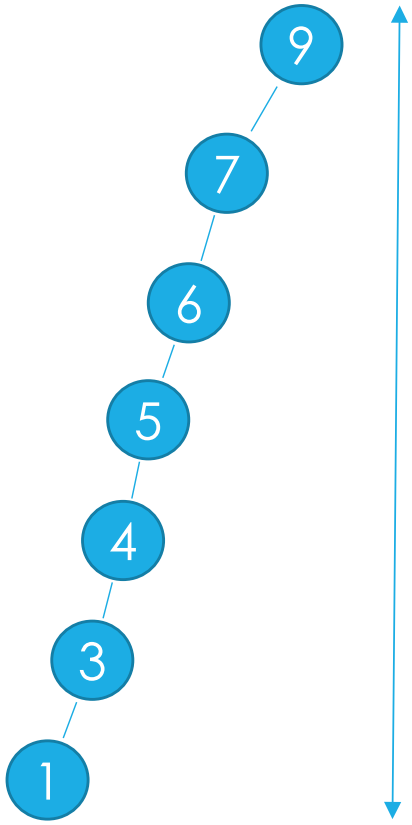


Ejemplo

Altura
O ($log_2(N)$)



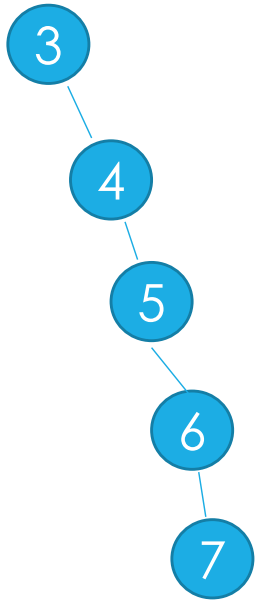
ARBOL LLENO



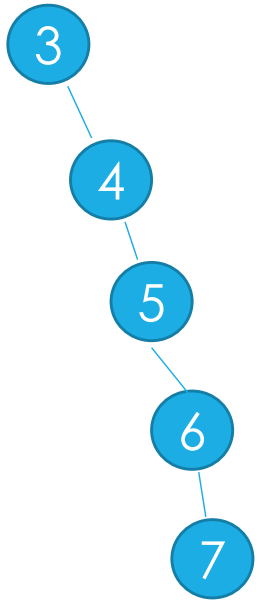
Altura
O(N)

ARBOL DEGENERADO

Insertar (3,4,5,6,7)

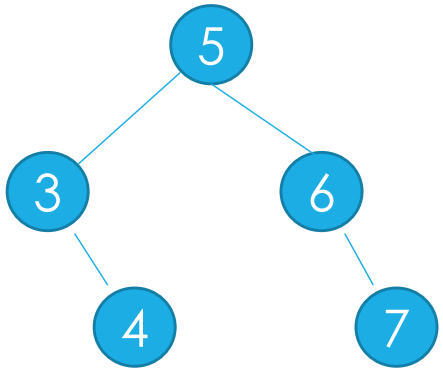


Insertar (3,4,5,6,7)



Insertar (3,4,5,6,7)

Si la raíz es la mediana: 5



Búsqueda (x, arbol)

Si x == arbol->valor // raíz

- ¡found!

Si x<= raíz

- Si arbol->izquierdo !=NULL
  - Búsqueda (x, árbol->izquierdo)
- Sino
  - //No esta en el arbol

Sino si arbol->derecho !=NULL

- Búsqueda (x, árbol->derecho)

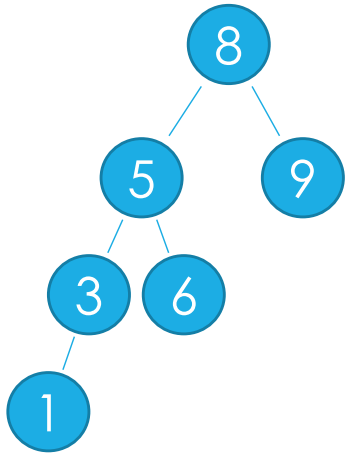
Si no

- //No esta en el arbol

La búsqueda termina cuando se encuentra el elemento o cuando no hay mas nodos que visitar

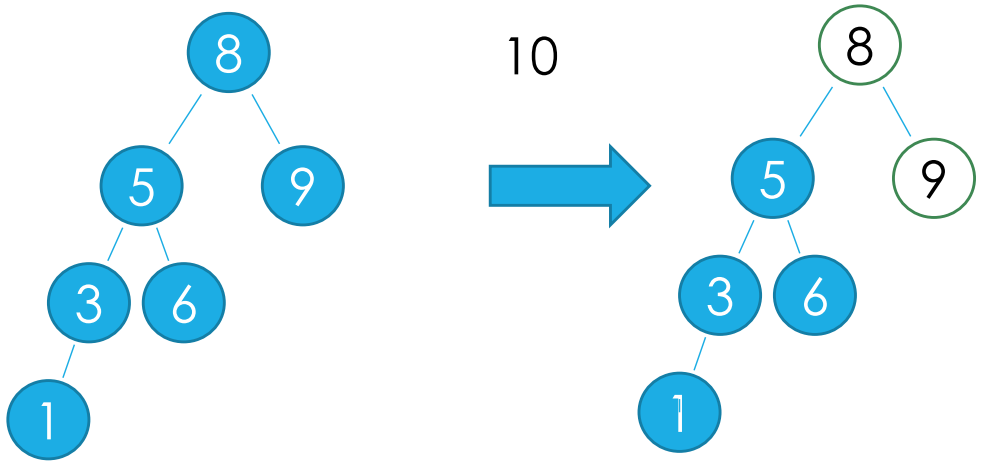
Inserción

1.- Encontrar el lugar de inserción
2.- Crear nueva hoja



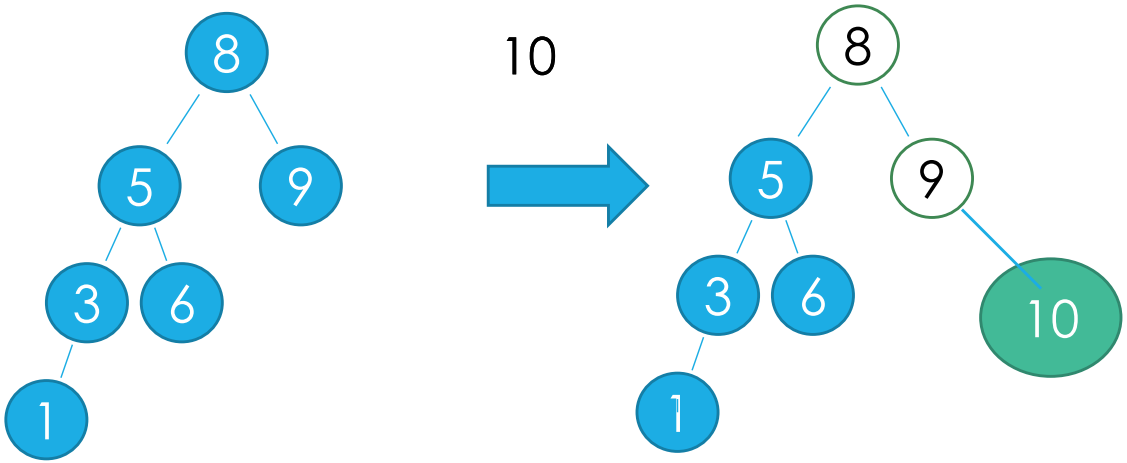
Inserción

1.- Encontrar el lugar de inserción
2.- Crear nueva hoja



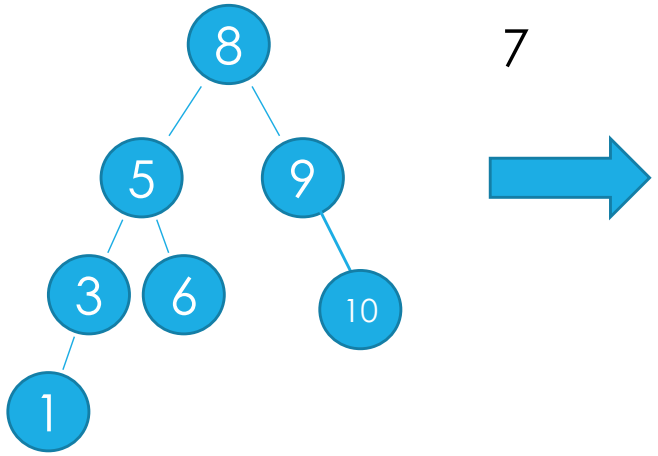
Inserción

1.- Encontrar el lugar de inserción
2.- Crear nueva hoja

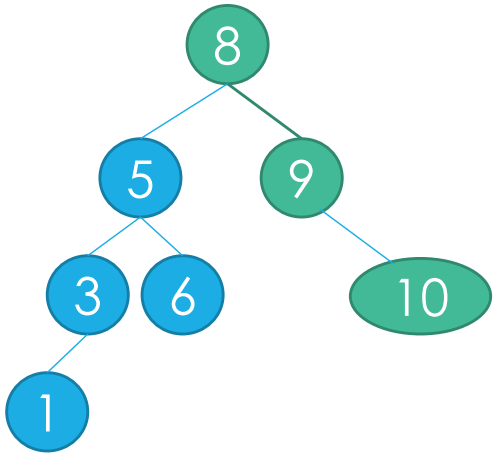


Inserción

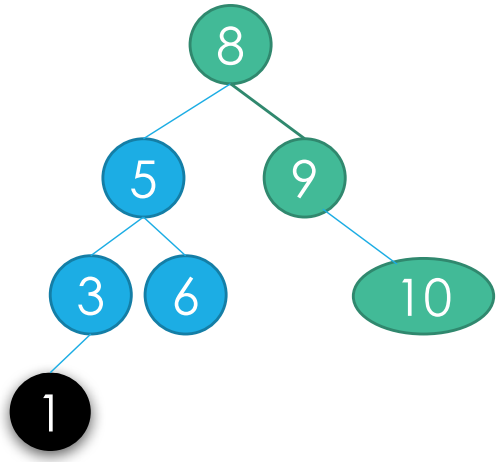
1.- Encontrar el lugar de inserción
2.- Crear nueva hoja



Obtener mínimo

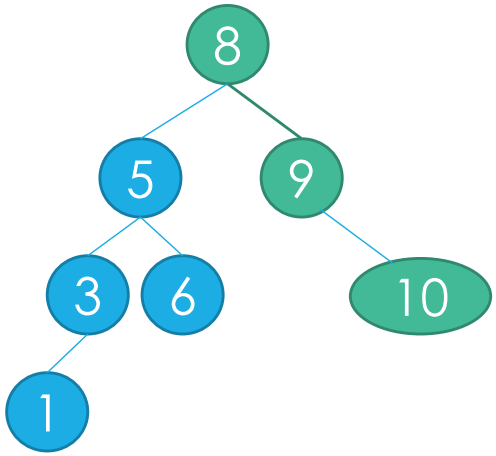


Obtener mínimo

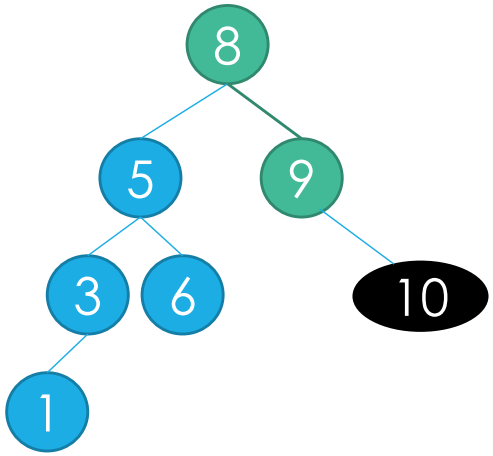


El mínimo elemento se encuentra en el nodo situado más a la izquierda

Obtener máximo



Obtener máximo



El máximo elemento se encuentra en el nodo situado más a la derecha

Borrado

Buscarlo

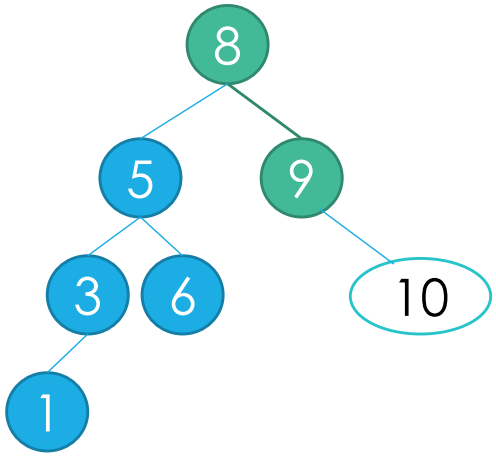
Una vez encontrado -> eliminarlo

Casos:

- Es una hoja (no tiene hijos)
- El nodo solo tiene un hijo
- El nodo tiene ambos hijos

Borrado de una hoja

Borrar (10)

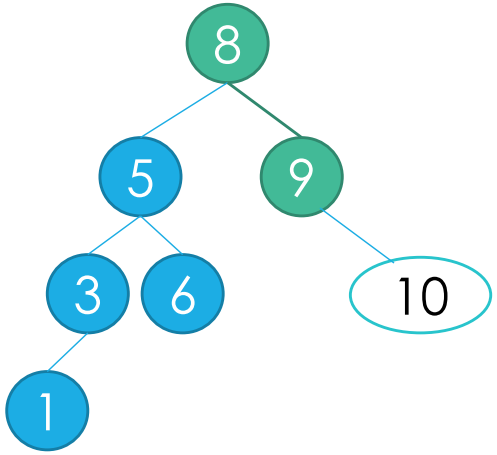


Borrado de una hoja

Borrar (10)

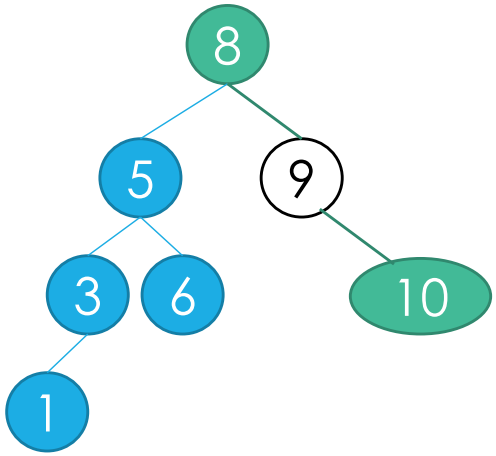
Simplemente eliminar nodo

Eliminar referencia desde el padre



Borrado de nodo con 1 hijo

Borrar (9)

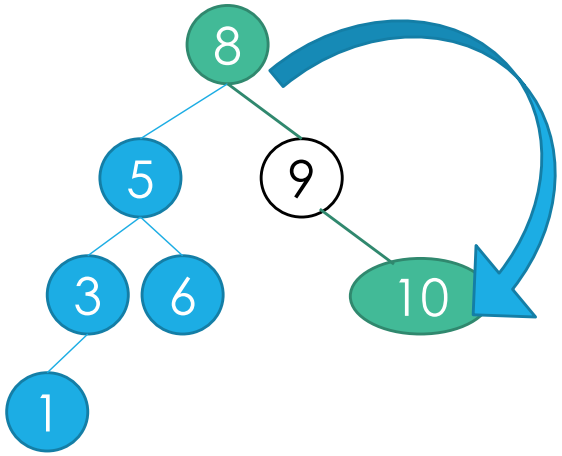


Borrado de nodo con 1 hijo

Borrar (9)

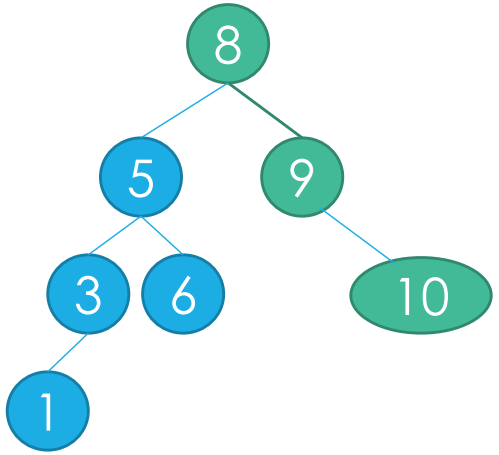
El abuelo adopta al hijo

Eliminar nodo



Borrado de nodo con 2 hijos

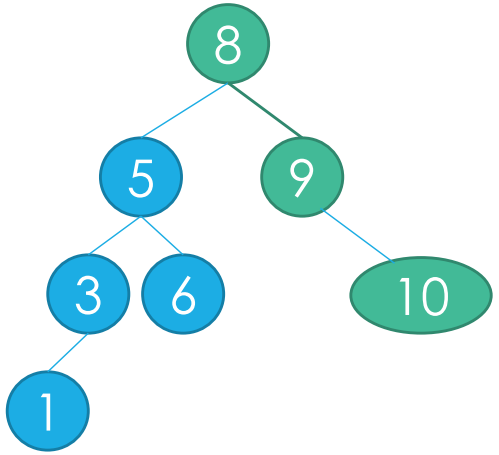
Borrar(5)
Sustituir el nodo a borrar con su **sucesor**
Eliminar el nodo *sucesor*



¿Quién es el sucesor?

Es el menor dato de todos aquellos que son mayores de nodo a borrar.

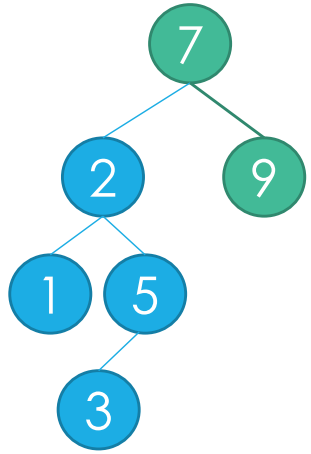
Sucesor (5) = 6



¿Quién es el sucesor?

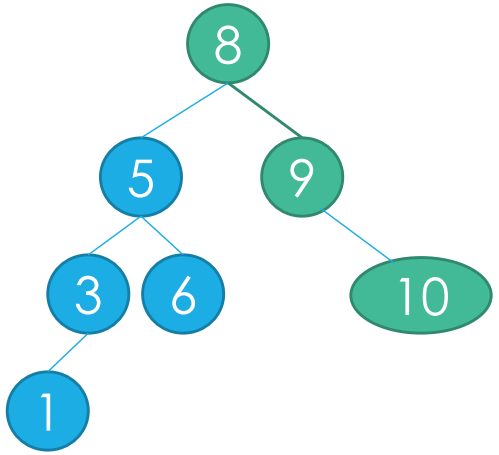
Es el **menor** de todos aquellos que son **mayores** al nodo a borrar.

Sucesor (7)= 9
Sucesor (2)= 3



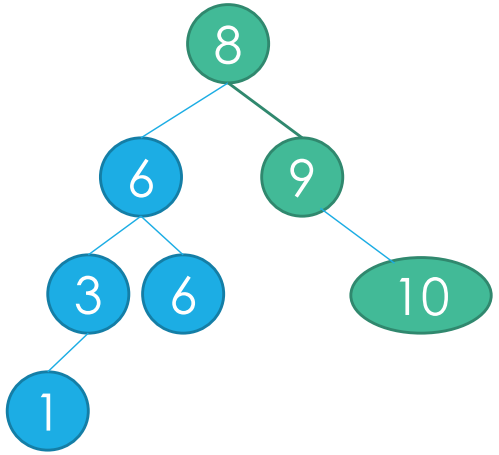
Borrado de nodo con 2 hijos

Borrar(5)
Sustituir el nodo a borrar con su **sucesor**
Eliminar el nodo *sucesor*



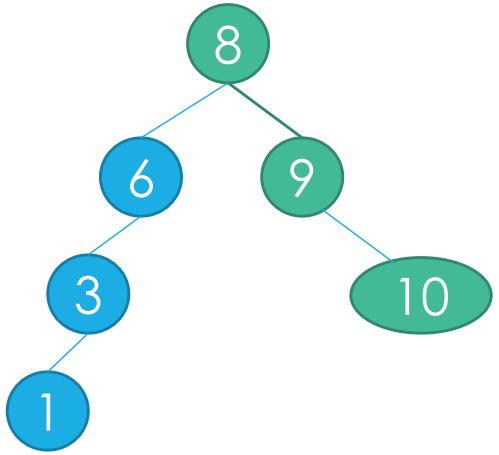
Borrado de nodo con 2 hijos

Borrar(5)
Sustituir el nodo a borrar con su **sucesor**
Eliminar el nodo *sucesor*



Borrado de nodo con 2 hijos

Borrar(5)
Sustituir el nodo a borrar con su **sucesor**
Eliminar el nodo *sucesor*



Representación

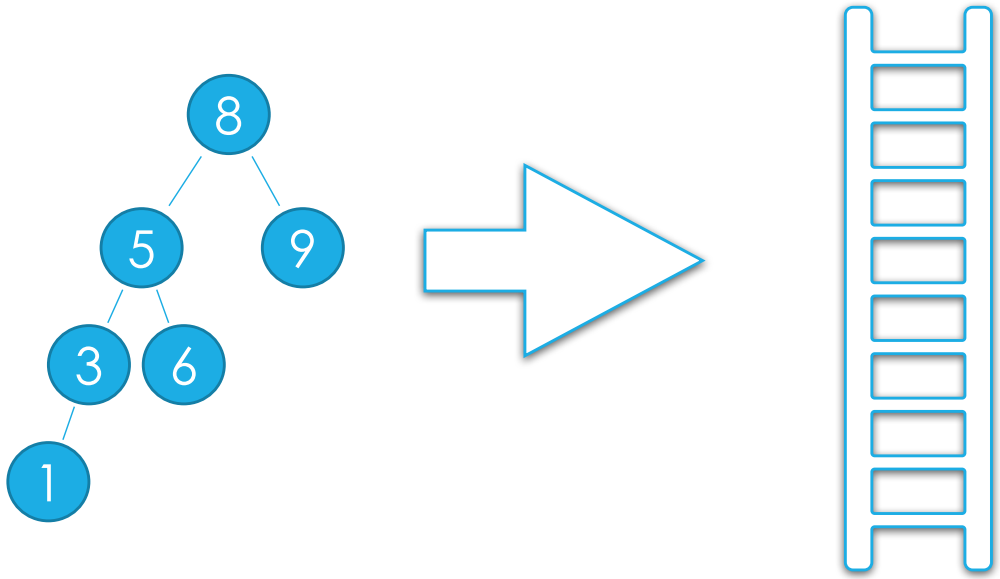
DE LA ESTRUCTURA DE DATOS ABB (BST)

Representación / Implementación

Arreglos

Nodos enlazados

Con arreglos



Implementación en arreglos

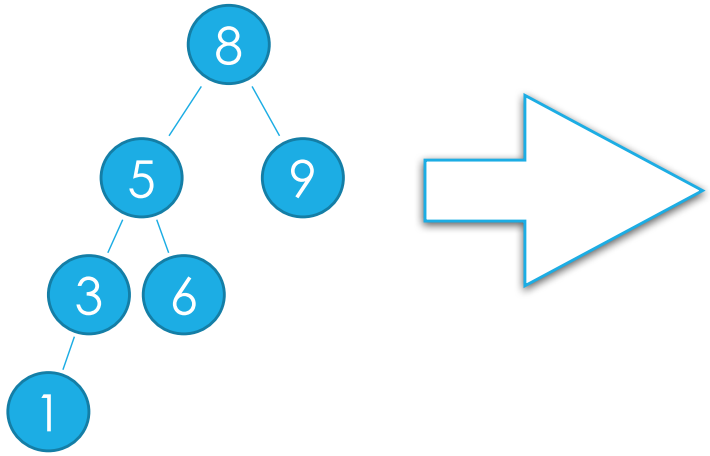
Los elementos se almacenan en un arreglo.

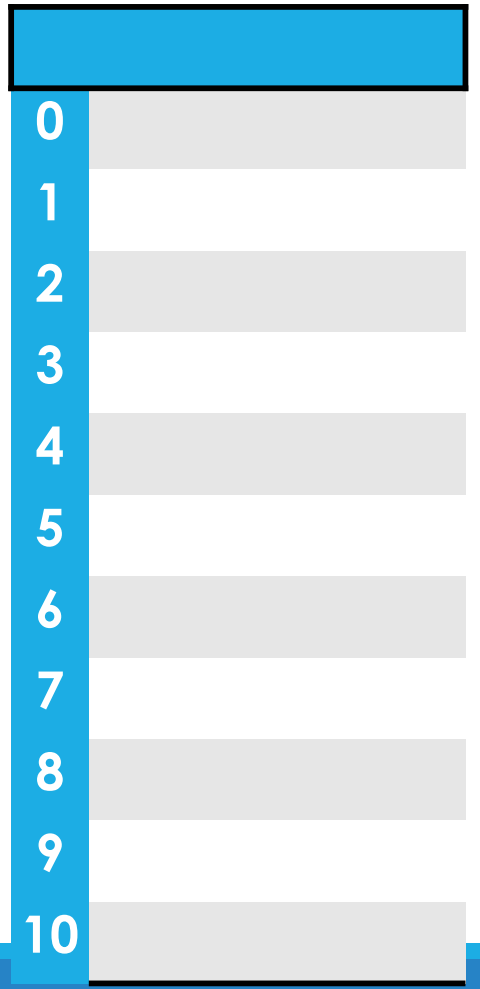
Cada nodo se representa por su posición en el vector, en donde:

	Posición
Raiz	0
Padre (nodo)	(nodo - 1)/2
HijoIzq(nodo)	2 * nodo + 1
HijoDer(nodo)	2 * nodo + 2

Con arreglos

	Posición
Raiz	0
Padre (nodo)	(nodo - 1)/2
HijoIzq(nodo)	2 * nodo + 1
HijoDer(nodo)	2 * nodo + 2





1) Diseño de la estructura

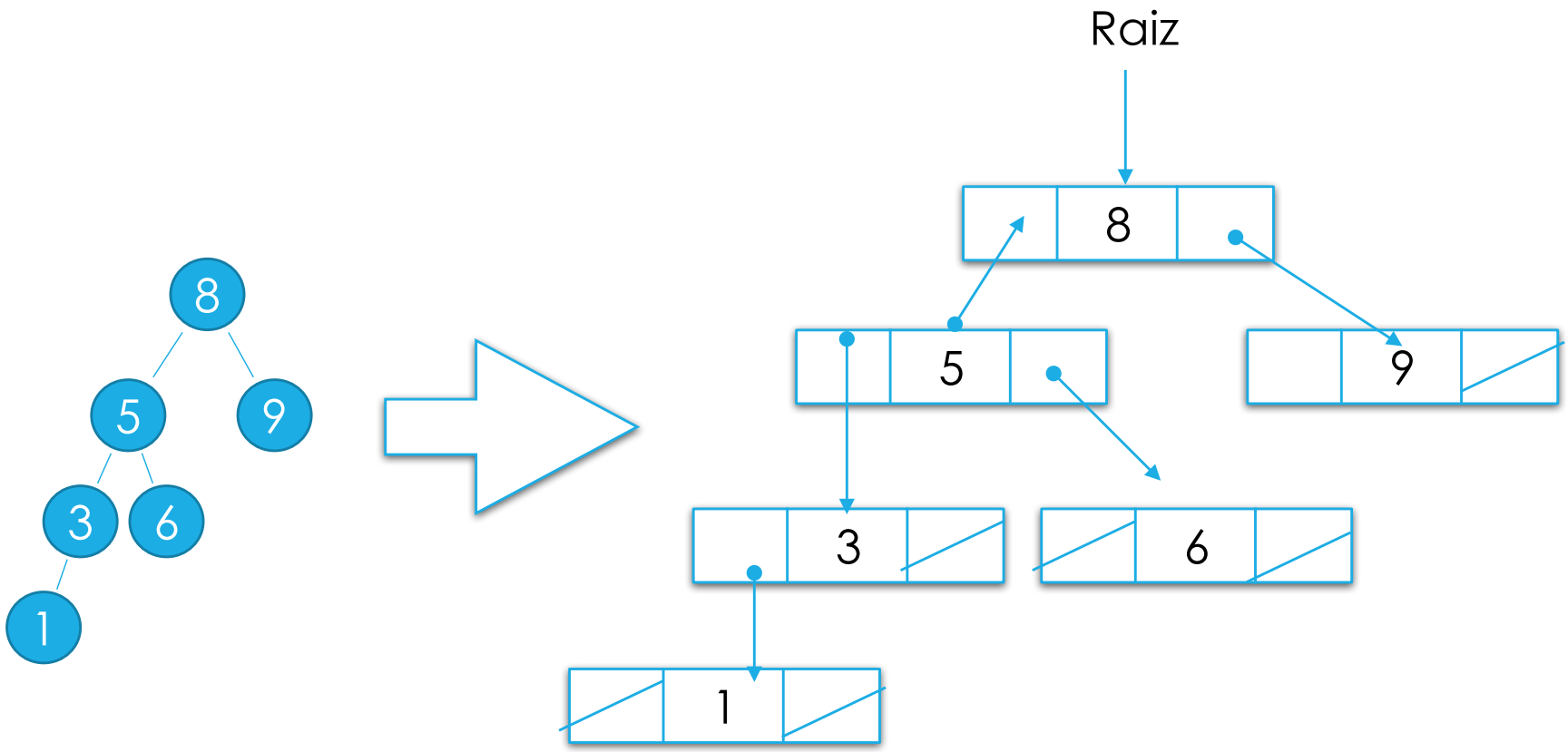
```
#define MAX 100


int arbol[MAX];
```

Funciones básicas

	Recibe	Regresa	
Insertar	Valor	posicion	Ó -1 si no hay espacio
Eliminar	Valor	posición	Ó -1 si no existe
Máximo	Nada	Valor	
Mínimo	Nada	Valor	
Buscar	Valor	Posición	Ó -1 si no existe


MANOS A
LA OBRA

Con nodos enlazados



1) Diseñar la estructura

```
typedef struct snodo
{
    int valor;
    struct snodo izq,
                 der;
}tnodo;
```

```
tnodo raiz=NULL;
```

Funciones básicas

	Recibe	Regresa	
Insertar	Valor	Bandera	1 - se insertó 0 - no hay lugar
Eliminar	Valor	Bandera	1 - se eliminó 0 - no existe
Máximo	Nada	Valor	
Mínimo	Nada	Valor	
Buscar	Valor	apuntador	Ó - NULL si no existe


MANOS A
LA OBRA

Claves no únicas

¿Su implementación permite claves duplicadas?

¿De no ser así, que tiene que modificar para que se acepten?

¿Qué sucede al eliminar una clave duplicada?

¿Qué podemos hacer para diferenciar una clave de otra?

Consulta de claves en un intervalo

¿