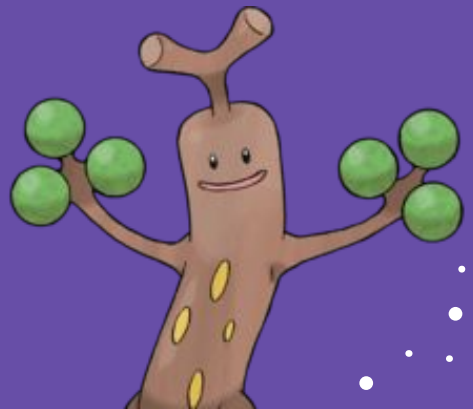
ARBOLES y ABB



Algo2Mendez - 2do cuatrimestre 2021
Izquierdo Stephanie & Sibikowski Nicolás

Temario

- Que es un árbol?
  - Qué características tienen?
- Qué son los árboles Binarios?
  - Como funcionan?
- Que son los ABBs?
  - En qué se diferencian con los árboles binarios?

Origen

- ## Que es un Árbol?

  Se trata de una colección de nodos, que a su vez, pueden estar conectados a otros múltiples nodos. Un árbol consiste de un nodo principal r, llamado raiz, y cero o muchos subárboles no vacíos [T1, T2,..., Tn], cada uno de ellos con su raíz conectada mediante un vértice al nodo raíz r.

- ## Porque se crea?

  Nacen de la necesidad de representar una jerarquía en la estructura de los datos, asi como tambien de querer optimizar la búsqueda lineal de una lista.
  Con el árbol se intenta hacer algo similar a la busqueda binaria, tratando de darnos un acceso a los datos en un mejor tiempo.

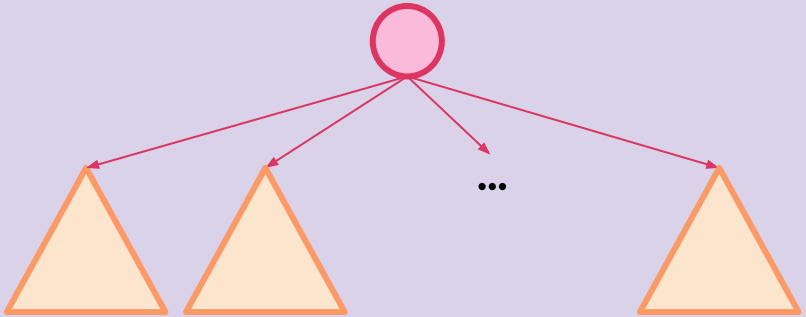
Tipos de Árboles

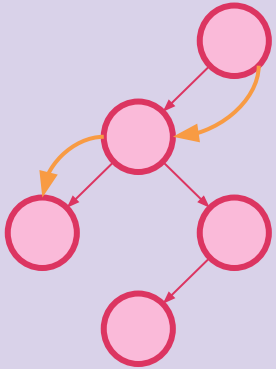
- Arboles N-Arios o generales
- Arboles binarios
- Árboles binarios de búsqueda
- AVL
- Árbol Rojo-Negro
- Arboles B, B+, B*
- Otros

Estructura

Recursividad

- Condicion de corte
- Llamada a la funcion

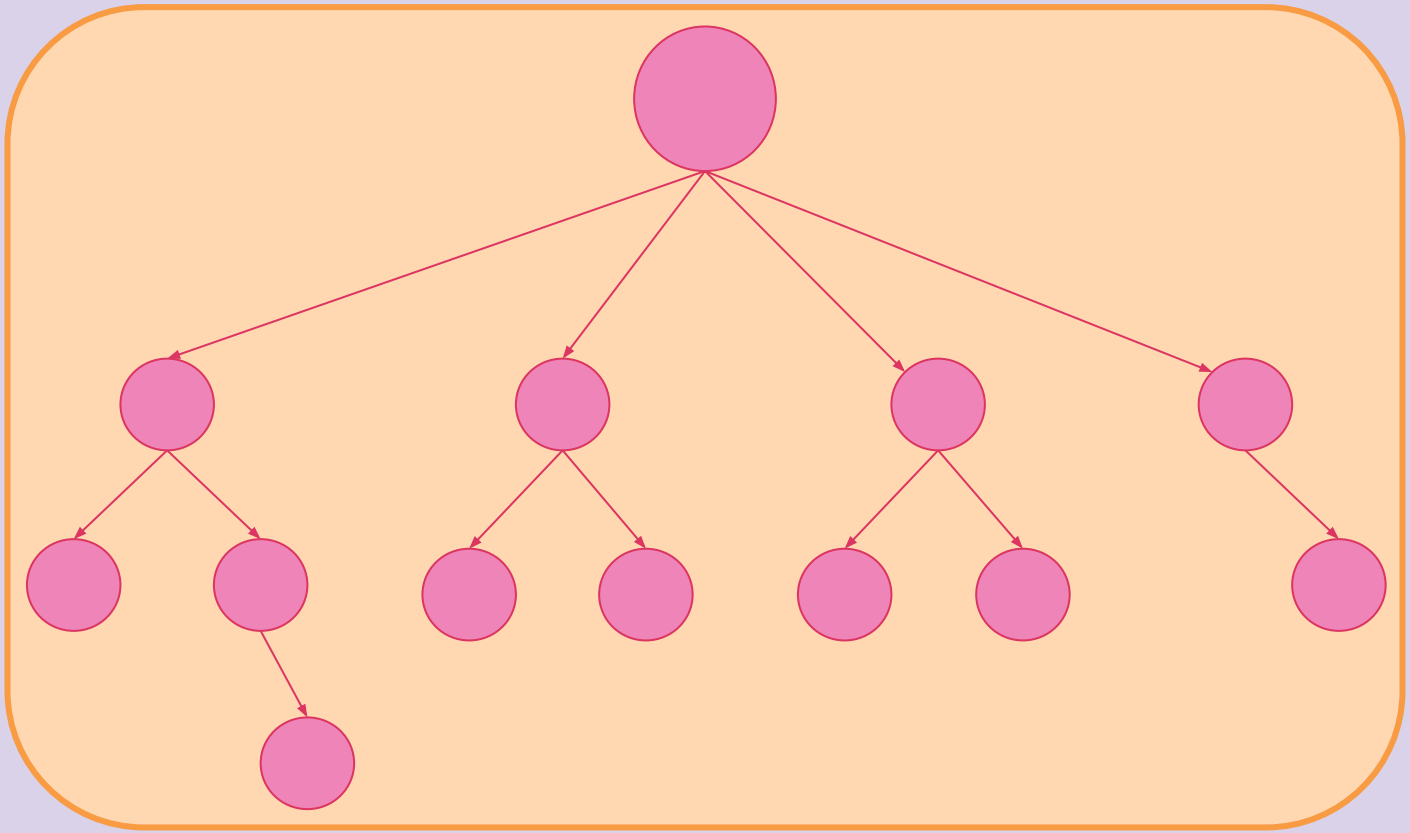




Definiciones

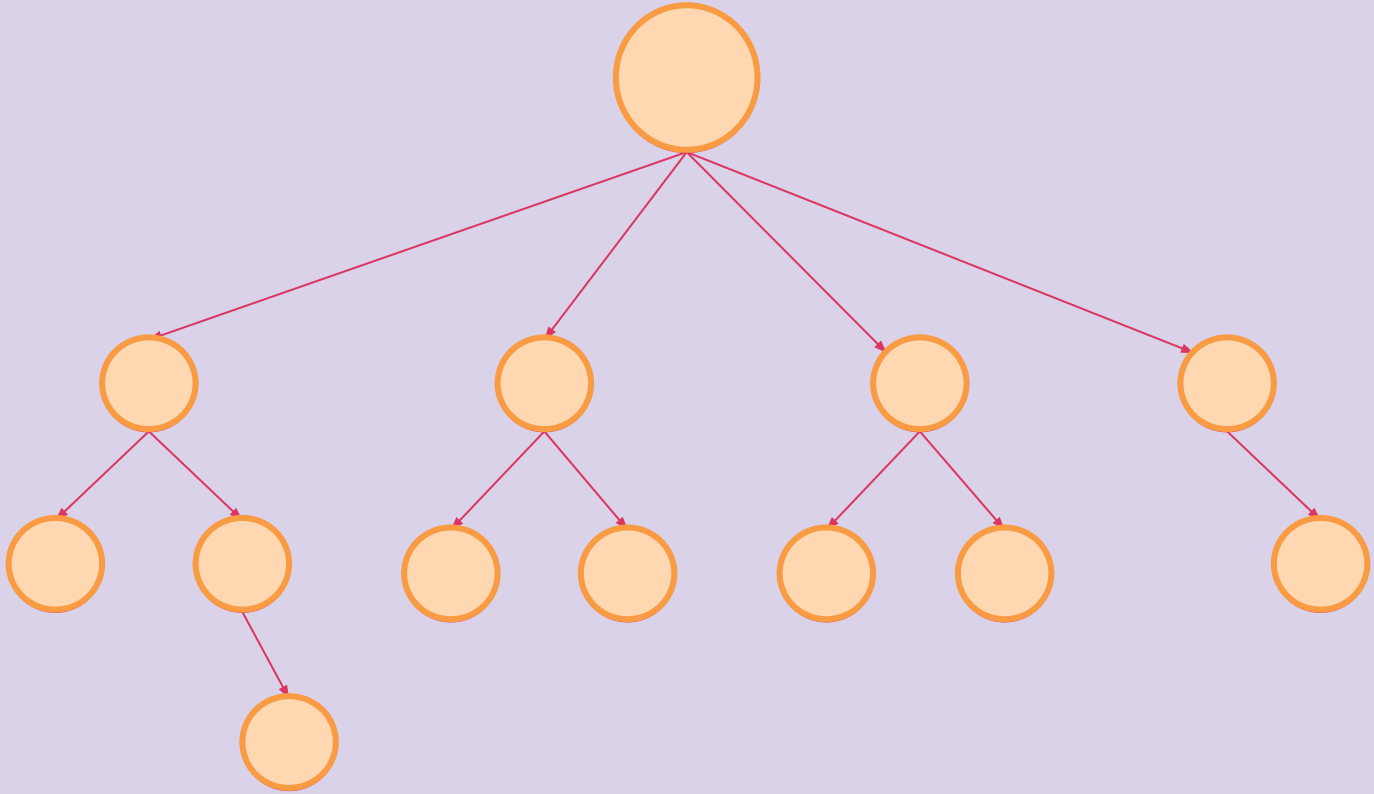
Arbol

Colección de nodos



Nodo

Son los elementos del árbol

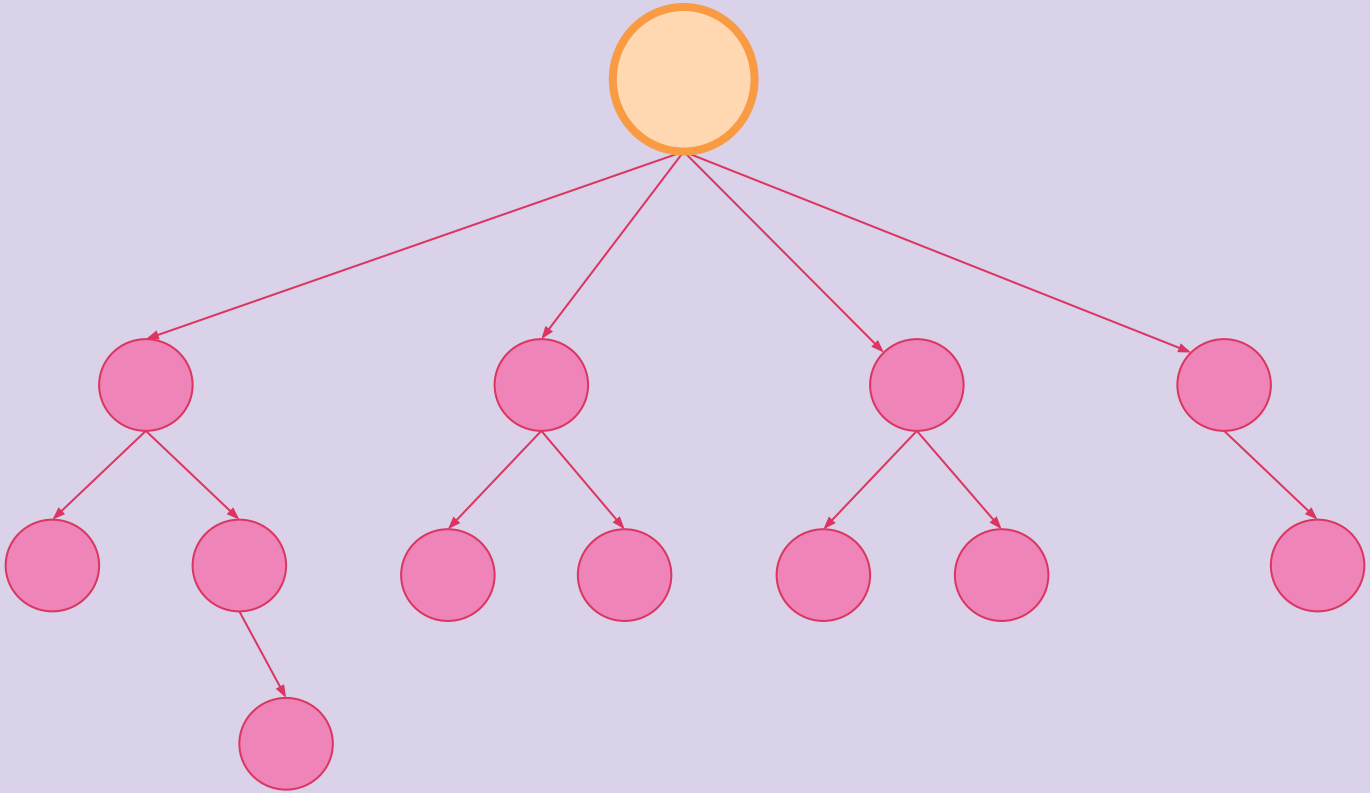


Observación

Cada nodo en un árbol tiene cero o más nodos conectados debajo de él pero solo uno superiormente

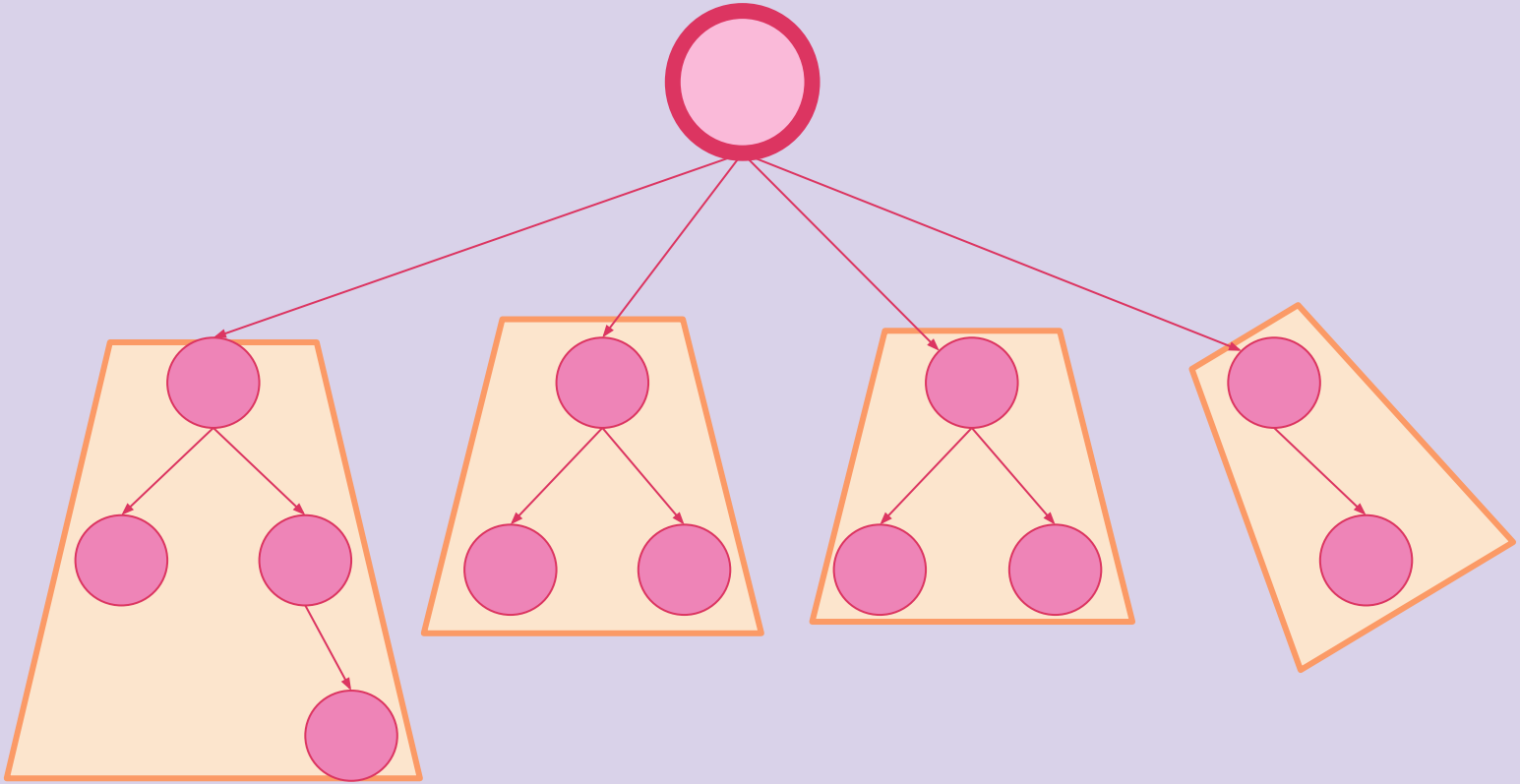
Raiz

Es el elemento en el 1er nivel del árbol



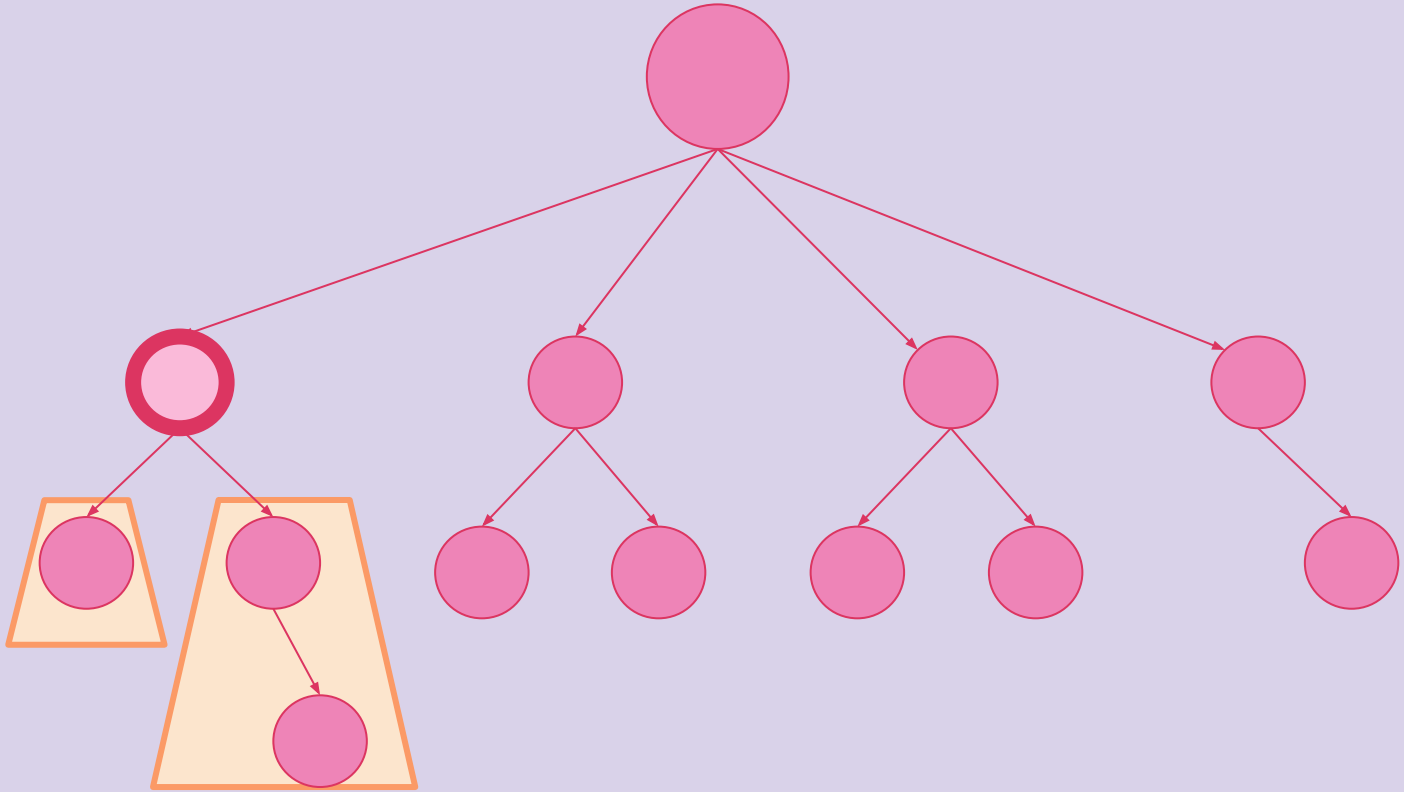
Subarbol

Parandome desde un nodo, puedo ver distintos árboles debajo de él



Subarbol

Parandome desde un nodo, puedo ver distintos árboles debajo de él

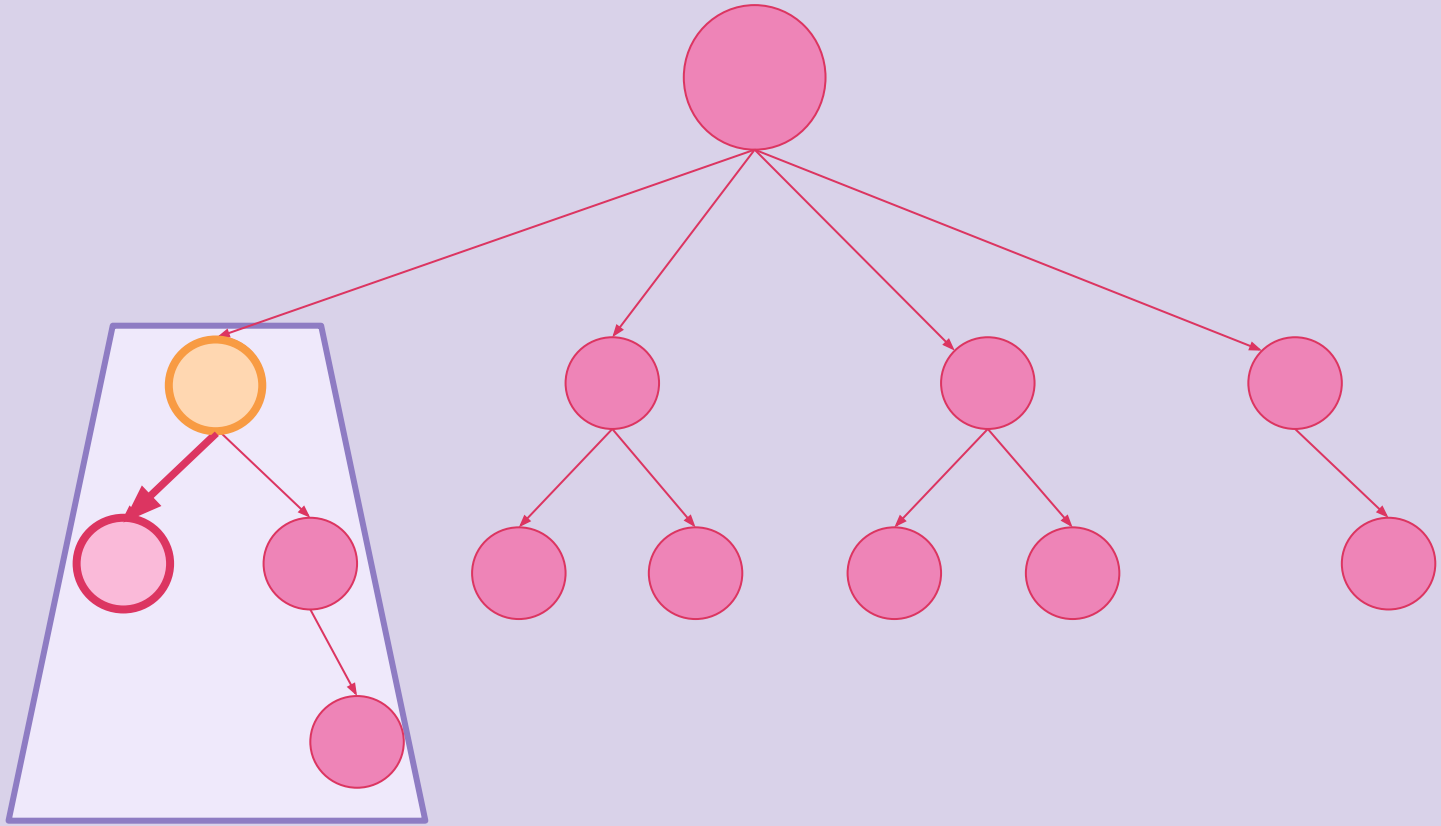


Nodos padres

Desde un nodo, el nodo inmediatamente superior conectado es el padre

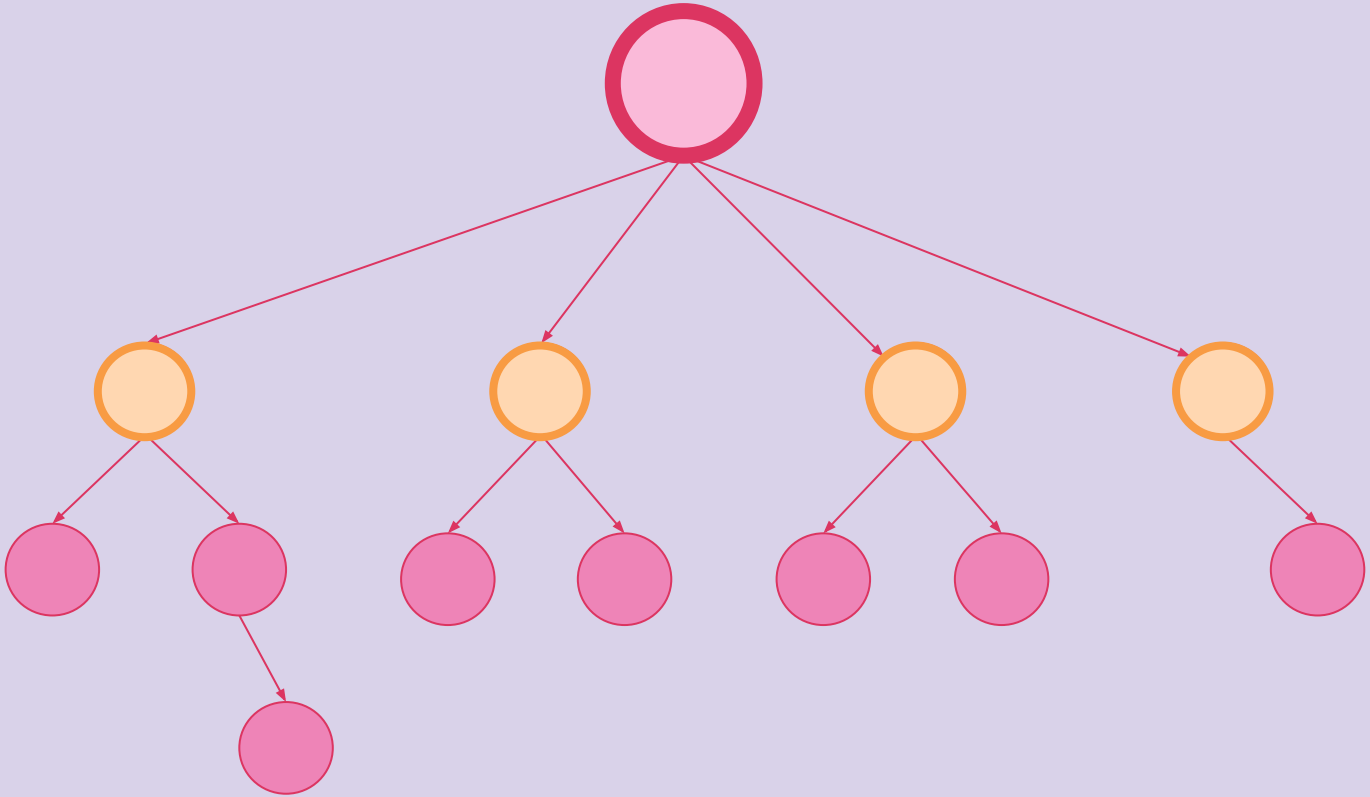
Nodos padres

Desde un nodo, el nodo inmediatamente superior conectado es el padre



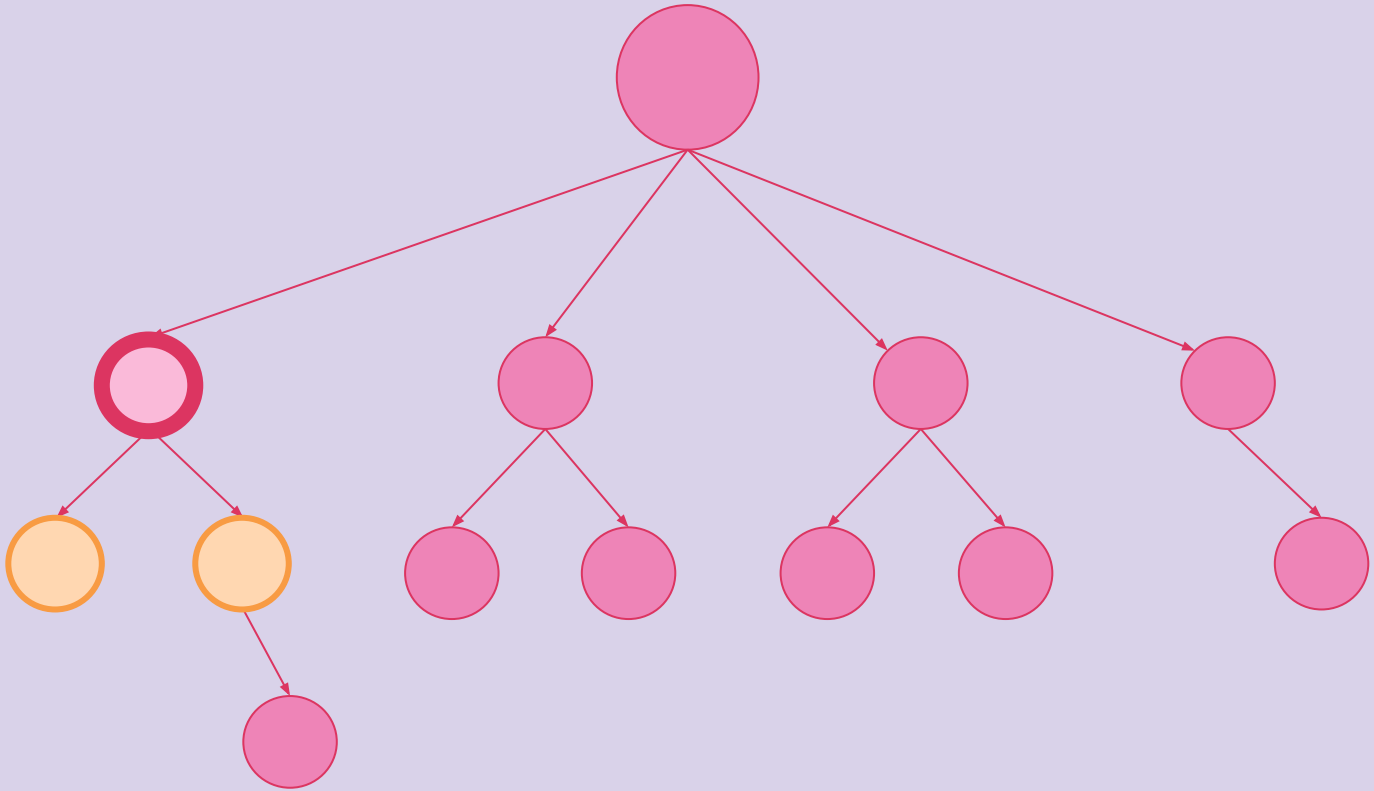
Nodos Hijos

Los nodos conectados en un nivel inferior son los hijos del nodo en el que estoy parado



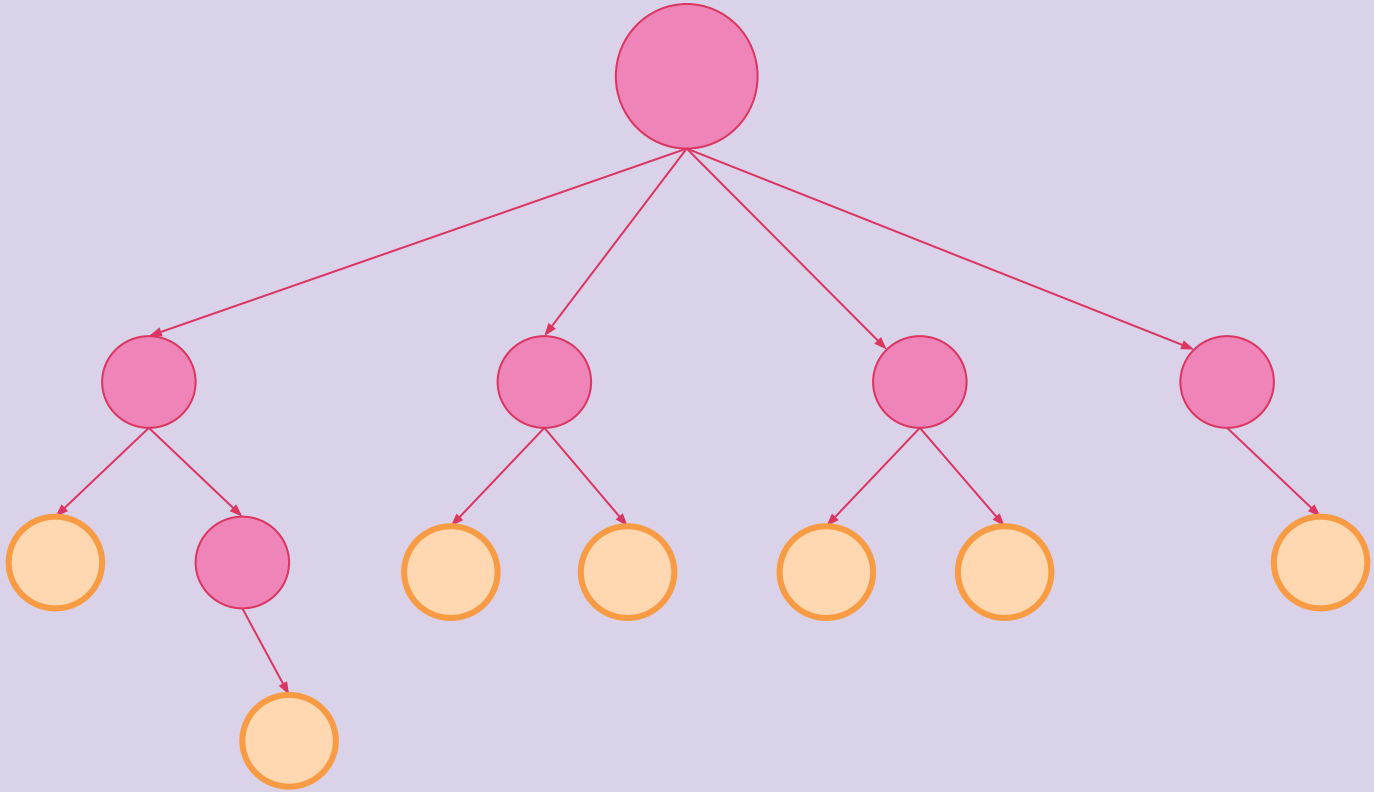
Nodos Hijos

Los nodos conectados en un nivel inferior son los hijos del nodo en el que estoy parado

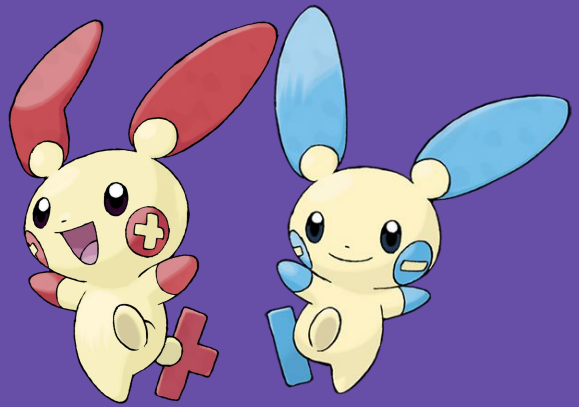


Hojas

Nodos que no tienen hijos



Arbol Binario



Definicion

- Estos árboles están íntimamente relacionados a las operaciones de búsqueda, con el objetivo de aproximarse a la búsqueda binaria
- El nodo raíz está solamente conectado a dos subárboles, lo cual nos permite determinar la noción de izquierda y derecha.

Operaciones

- Crear
- Destruir
- Insertar
- Borrar
- Buscar
- Vacio
- Recorrer

Recorridos

Una de las operaciones más importantes que se realizan con los árboles binarios, es el recorrido de estos.

Recorrer un árbol significa de alguna forma pasar por cada uno de los nodos.

Imaginemos que llamamos N al nodo actual, D al subárbol derecho, e I al subárbol izquierdo.

3 2 1

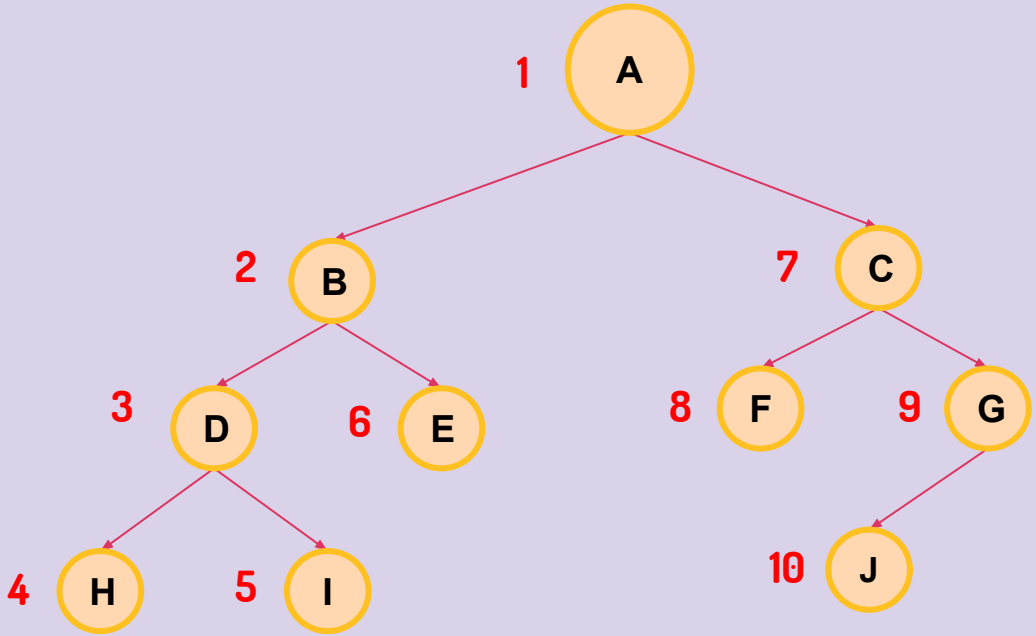
3! = 6 combinaciones!

Tenemos 6 posibles recorridos!

NID	IND	IDN	NDI	DNI	DIN
Preorder	Inorder	Postorder			

Recorrido preorden

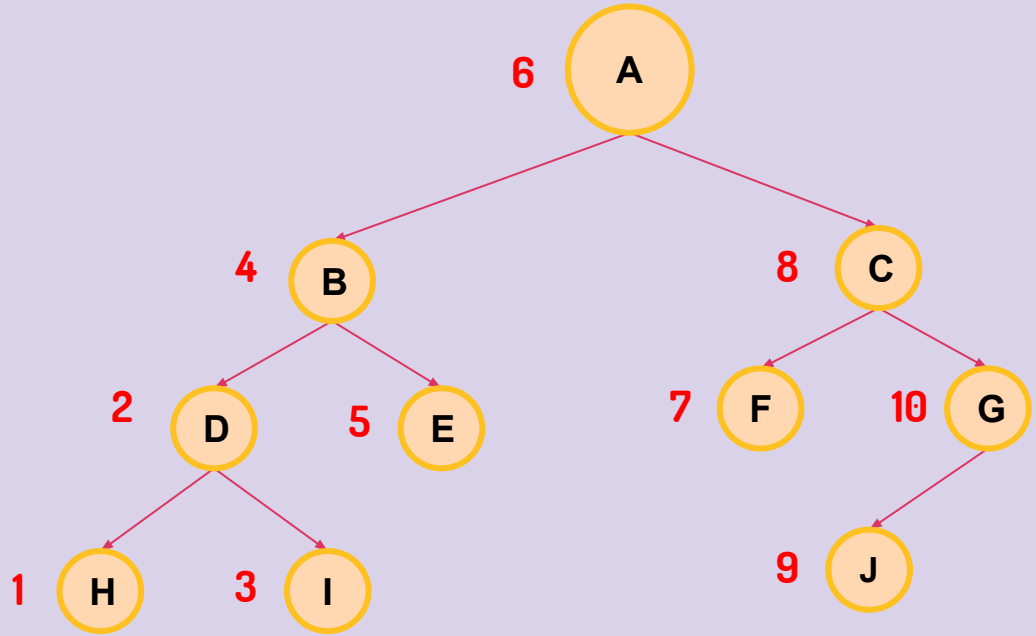
- **Preorden:** Primero se visita en nodo actual luego el subárbol izquierdo y luego el derecho.



[A, B, D, H, I, E, C, F, G, J]

Recorrido inorden

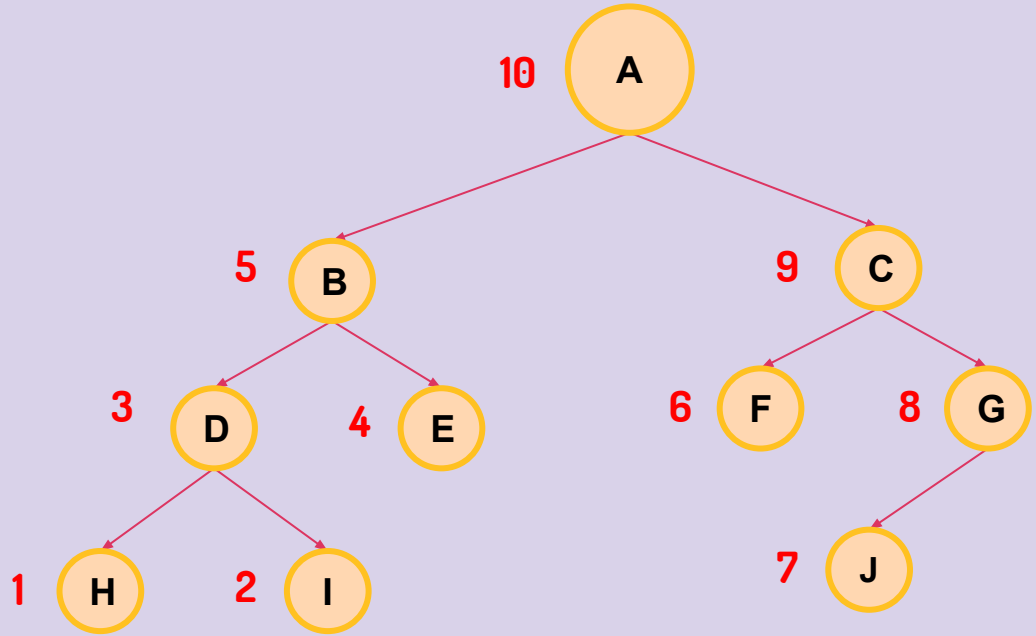
- **Inorden:** Primero se visita el subárbol izquierdo, luego el nodo actual y por último el subárbol derecho.



[H, D, I, B, E, A, F, C, J, G]

Recorrido postorden

- **Postorden:** Primero se visita el subárbol izquierdo, luego al sub-árbol derecho y por último al nodo actual.

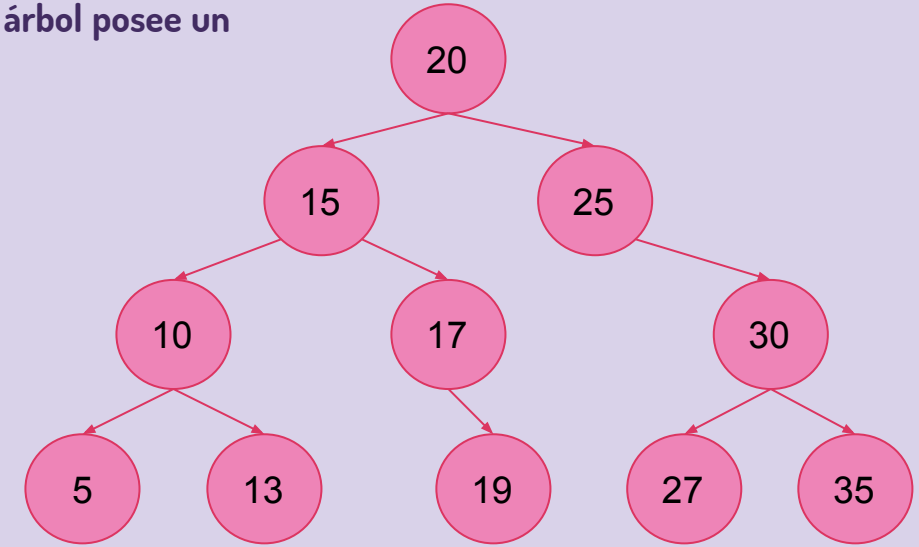


[H, I, D, E, B, F, J, G, C, A]

Árbol Binario de Búsqueda

Definición

Tenemos un orden por ende tenemos una forma de comparar los elementos para poder definir este orden y cada nodo del árbol posee un valor o una clave única.



Caracteristicas

- Las claves mayores se insertan en los subárboles derechos
- Las claves menores se insertan en los subárboles izquierdos
- Ambos subárboles también son árboles de búsqueda binaria

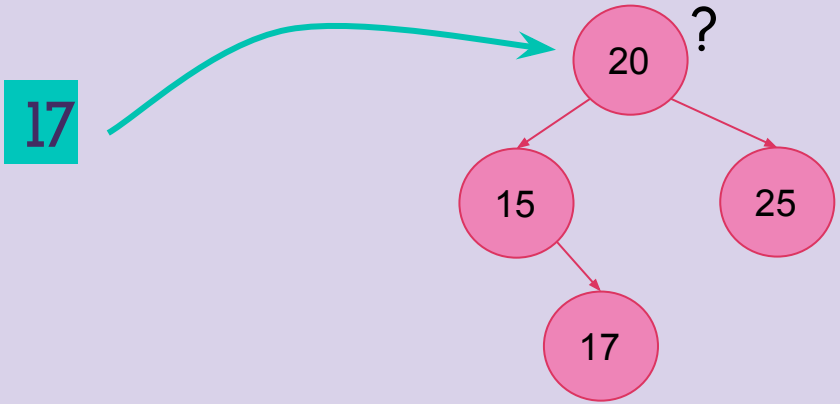
Busqueda

La búsqueda de un elemento comienza en el nodo raíz y sigue estos pasos:

1. La clave buscada se compara con la clave del nodo raíz.

2. Si las claves son iguales, la búsqueda se detiene.

3. Si la clave buscada es mayor que la clave raíz, la búsqueda se reanuda en el sub-árbol derecho. Si la clave buscada es menor que la clave raíz, la búsqueda se reanuda en el sub-árbol izquierdo.



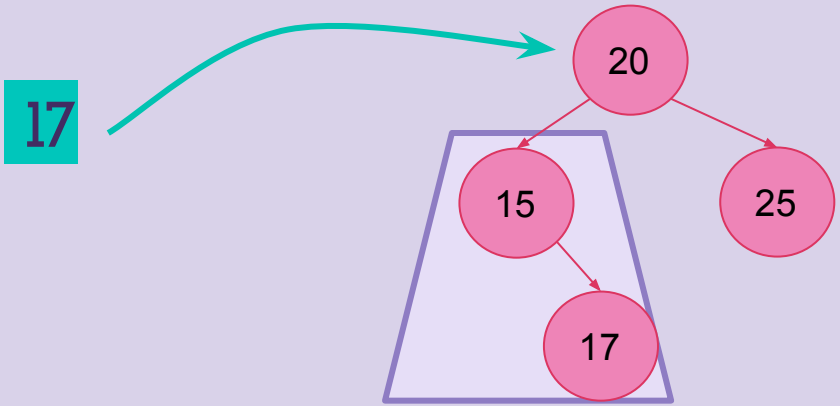
Busqueda

La búsqueda de un elemento comienza en el nodo raíz y sigue estos pasos:

1. La clave buscada se compara con la clave del nodo raíz.

2. Si las claves son iguales, la búsqueda se detiene.

3. Si la clave buscada es mayor que la clave raíz, la búsqueda se reanuda en el sub-árbol derecho. Si la clave buscada es menor que la clave raíz, la búsqueda se reanuda en el sub-árbol izquierdo.



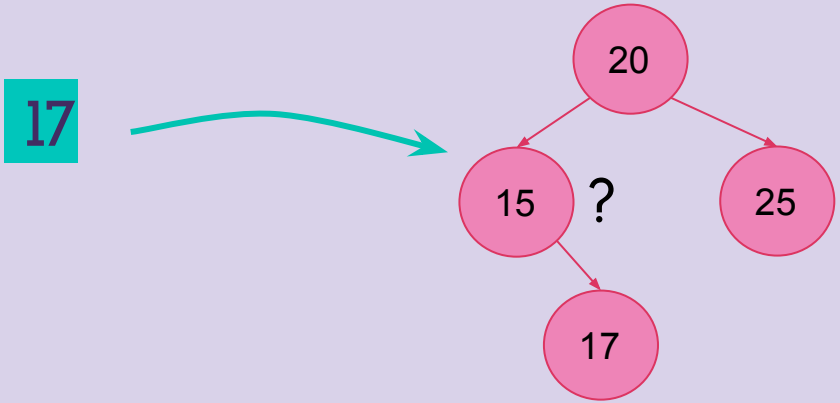
Busqueda

La búsqueda de un elemento comienza en el nodo raíz y sigue estos pasos:

1. La clave buscada se compara con la clave del nodo raíz.

2. Si las claves son iguales, la búsqueda se detiene.

3. Si la clave buscada es mayor que la clave raíz, la búsqueda se reanuda en el sub-árbol derecho. Si la clave buscada es menor que la clave raíz, la búsqueda se reanuda en el sub-árbol izquierdo.



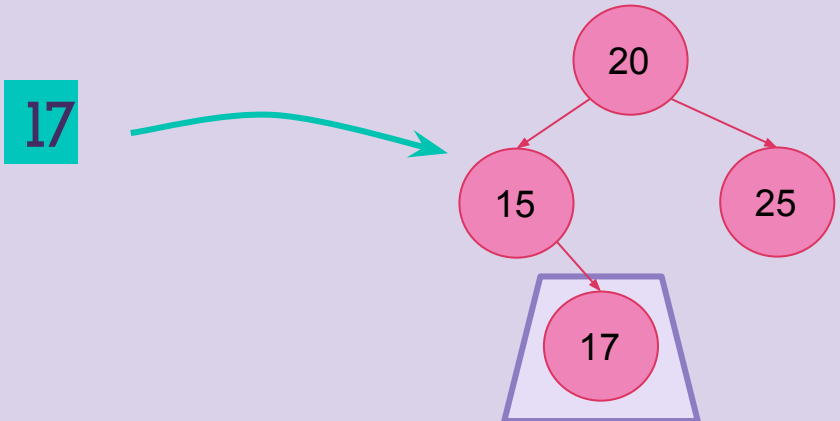
Busqueda

La búsqueda de un elemento comienza en el nodo raíz y sigue estos pasos:

1. La clave buscada se compara con la clave del nodo raíz.

2. Si las claves son iguales, la búsqueda se detiene.

3. Si la clave buscada es mayor que la clave raíz, la búsqueda se reanuda en el sub-árbol derecho. Si la clave buscada es menor que la clave raíz, la búsqueda se reanuda en el sub-árbol izquierdo.



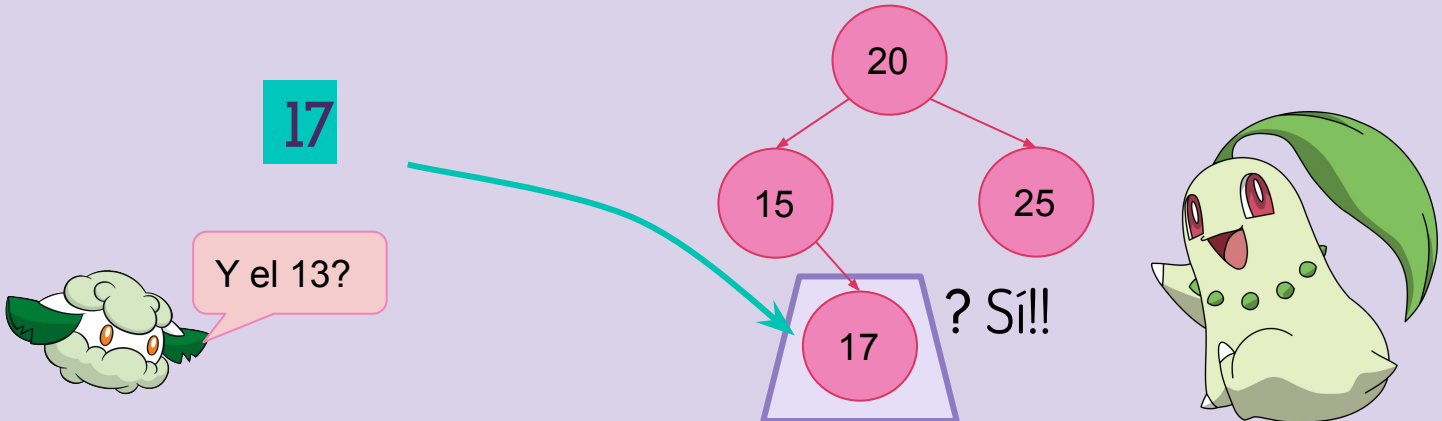
Busqueda

La búsqueda de un elemento comienza en el nodo raíz y sigue estos pasos:

1. La clave buscada se compara con la clave del nodo raíz.

2. Si las claves son iguales, la búsqueda se detiene.

3. Si la clave buscada es mayor que la clave raíz, la búsqueda se reanuda en el sub-árbol derecho. Si la clave buscada es menor que la clave raíz, la búsqueda se reanuda en el sub-árbol izquierdo.



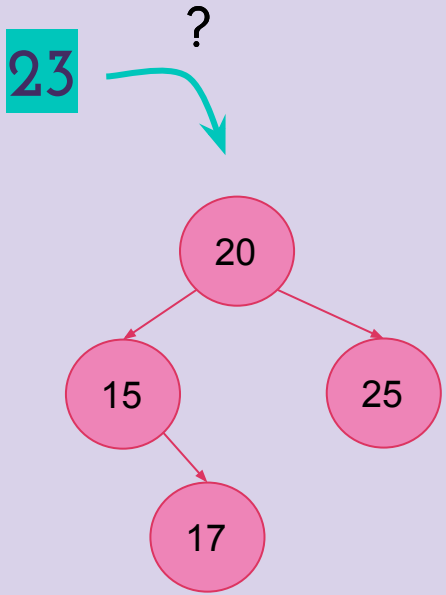
¿Complejidad?

Insertar

1. Comparo la clave del elemento a insertar con la clave del nodo raíz. Si es mayor avanzo hacia el subárbol derecho, si es menor hacia el izquierdo.

2. Repetir el paso 1 hasta encontrar un elemento con clave igual o llegar al final del sub-árbol donde debo insertar el nuevo elemento.

3. Cuando se llega al final creo un nuevo nodo, asignando null a los punteros izquierdo y derecho del mismo. Luego coloco el nuevo nodo como hijo izquierdo o derecho del anterior según sea el valor de la clave.

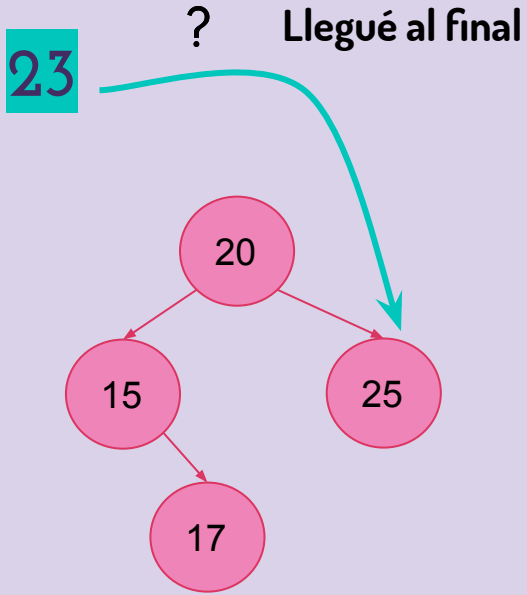


Insertar

1. Comparo la clave del elemento a insertar con la clave del nodo raíz. Si es mayor avanzo hacia el subárbol derecho, si es menor hacia el izquierdo.

2. Repetir el paso 1 hasta encontrar un elemento con clave igual o llegar al final del sub-árbol donde debo insertar el nuevo elemento.

3. Cuando se llega al final creo un nuevo nodo, asignando null a los punteros izquierdo y derecho del mismo. Luego coloco el nuevo nodo como hijo izquierdo o derecho del anterior según sea el valor de la clave.

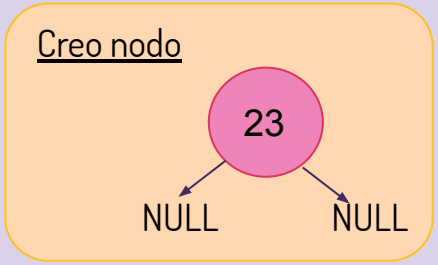


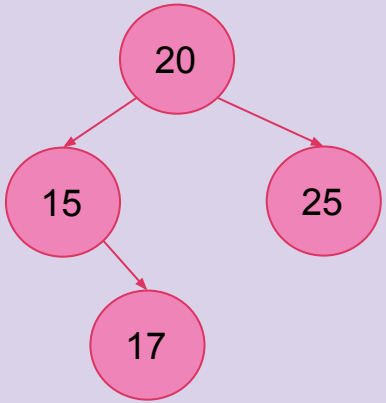
Insertar

1. Comparo la clave del elemento a insertar con la clave del nodo raíz. Si es mayor avanzo hacia el subárbol derecho, si es menor hacia el izquierdo.

2. Repetir el paso 1 hasta encontrar un elemento con clave igual o llegar al final del sub-árbol donde debo insertar el nuevo elemento.

3. Cuando se llega al final creo un nuevo nodo, asignando null a los punteros izquierdo y derecho del mismo. Luego coloco el nuevo nodo como hijo izquierdo o derecho del anterior según sea el valor de la clave.



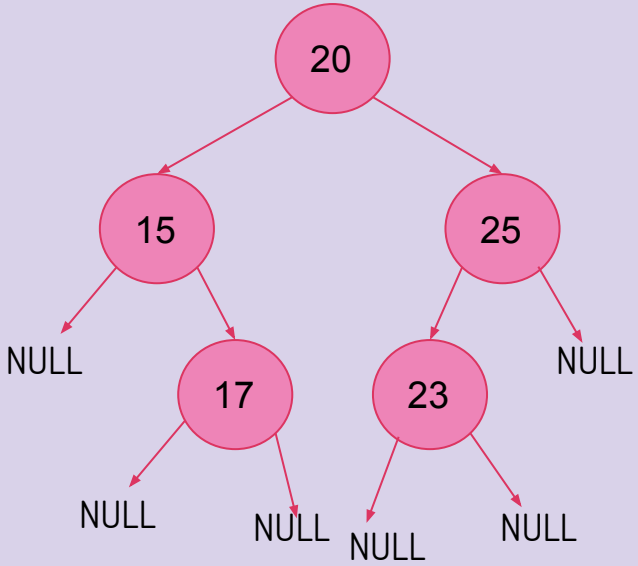


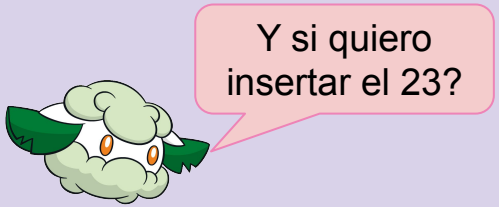
Insertar

1. Comparo la clave del elemento a insertar con la clave del nodo raíz. Si es mayor avanzo hacia el subárbol derecho, si es menor hacia el izquierdo.

2. Repetir el paso 1 hasta encontrar un elemento con clave igual o llegar al final del sub-árbol donde debo insertar el nuevo elemento.

3. Cuando se llega al final creo un nuevo nodo, asignando null a los punteros izquierdo y derecho del mismo. Luego coloco el nuevo nodo como hijo izquierdo o derecho del anterior según sea el valor de la clave.



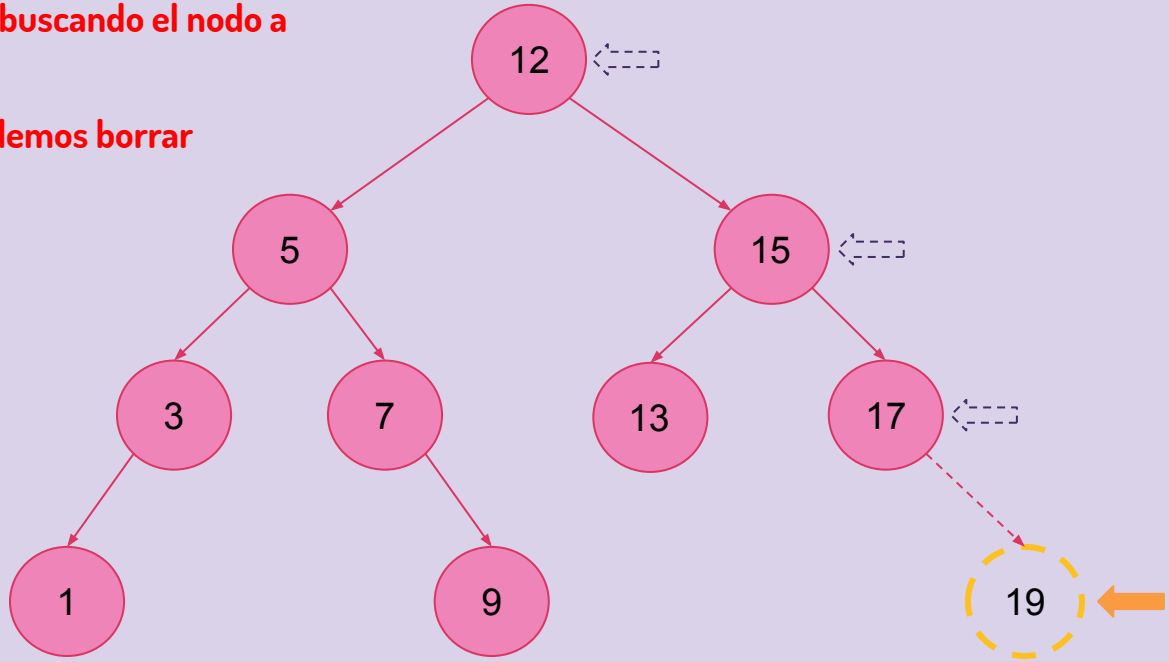


¿Complejidad?

Borrar Nodo - Nodo Hoja

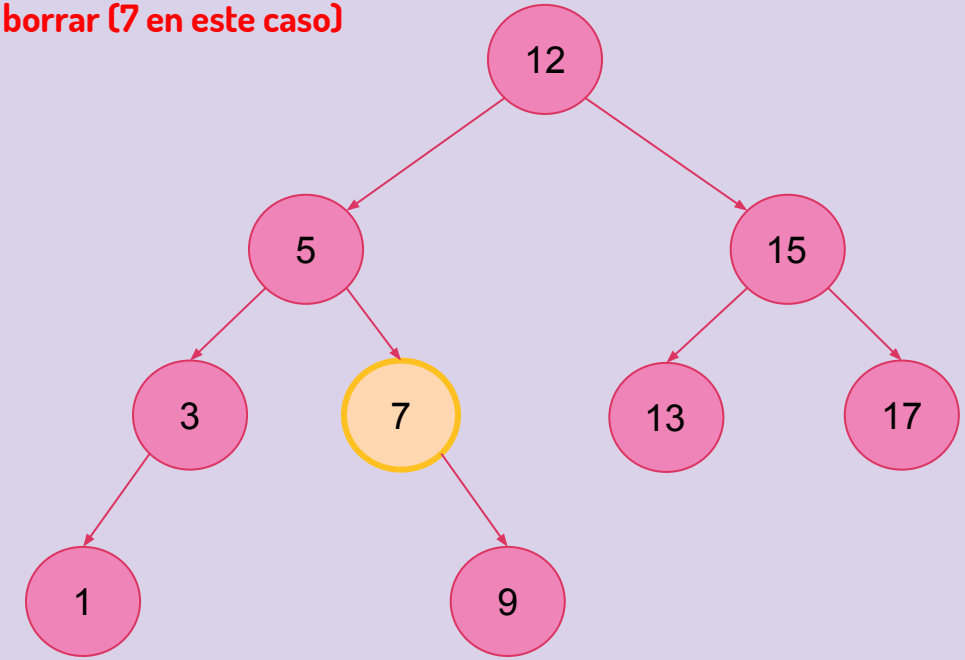
1. Bajamos por el árbol buscando el nodo a borrar

2. Al no tener hijos, podemos borrar tranquilamente



Borrar Nodo - Nodo con un hijo

1. Buscamos el nodo a borrar (7 en este caso)

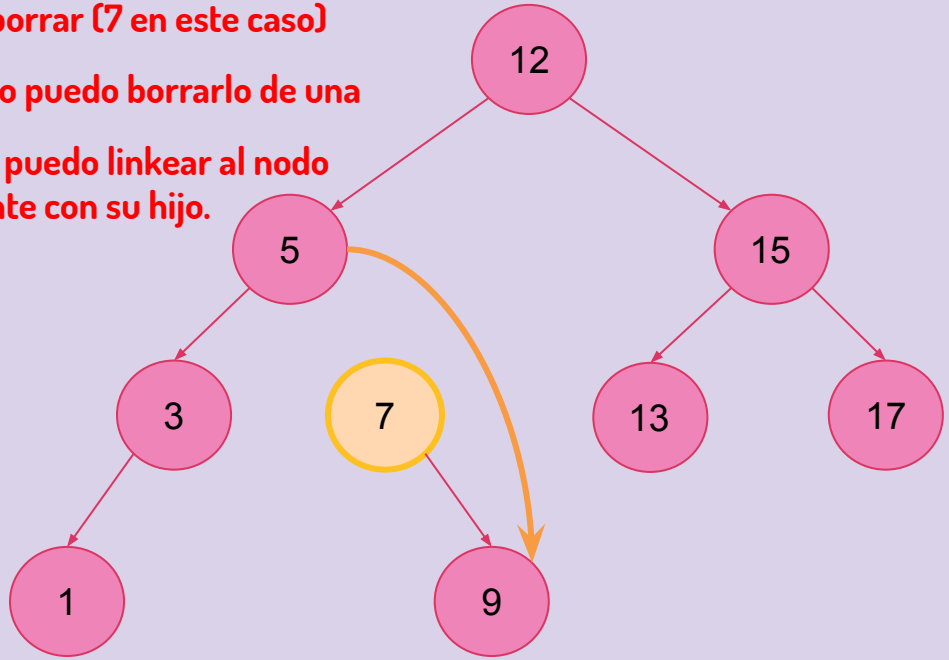


Borrar Nodo - Nodo con un hijo

1. Buscamos el nodo a borrar (7 en este caso)

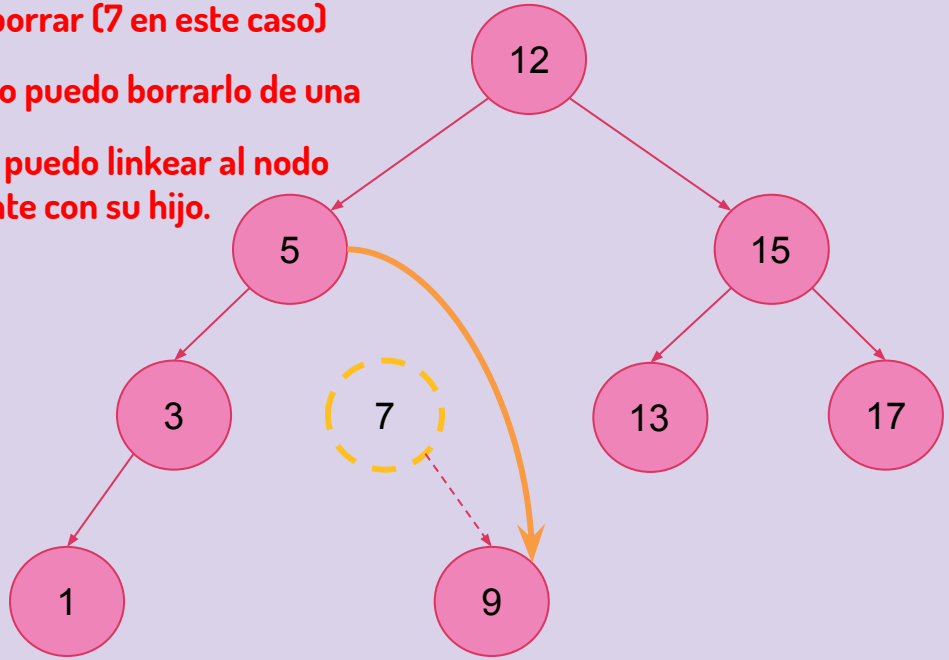
2. Como tiene un hijo no puedo borrarlo de una

3. Al tener un solo hijo, puedo linkear al nodo padre del 7 directamente con su hijo.



Borrar Nodo - Nodo con un hijo

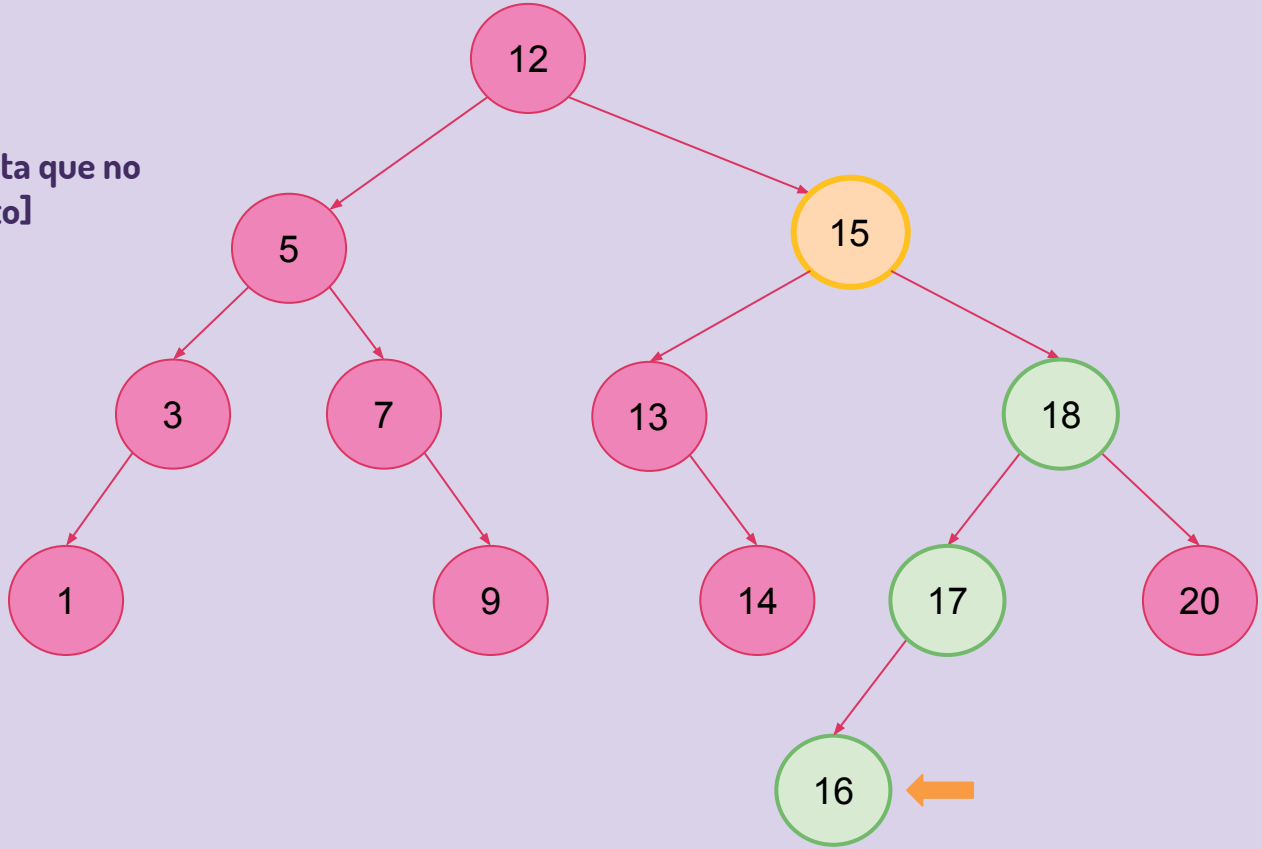
1. Buscamos el nodo a borrar (7 en este caso)
2. Como tiene un hijo no puedo borrarlo de una
3. Al tener un solo hijo, puedo linkear al nodo padre del 7 directamente con su hijo.
4. Borramos el nodo



Borrar Nodo - Nodo con dos hijos

1. Busco el sucesor inmediato:
    a. Tomo el hijo derecho inmediato
    b. Bajo por los hijos izquierdos (hasta que no haya más) [-> el 16 es el sucesor inmediato]

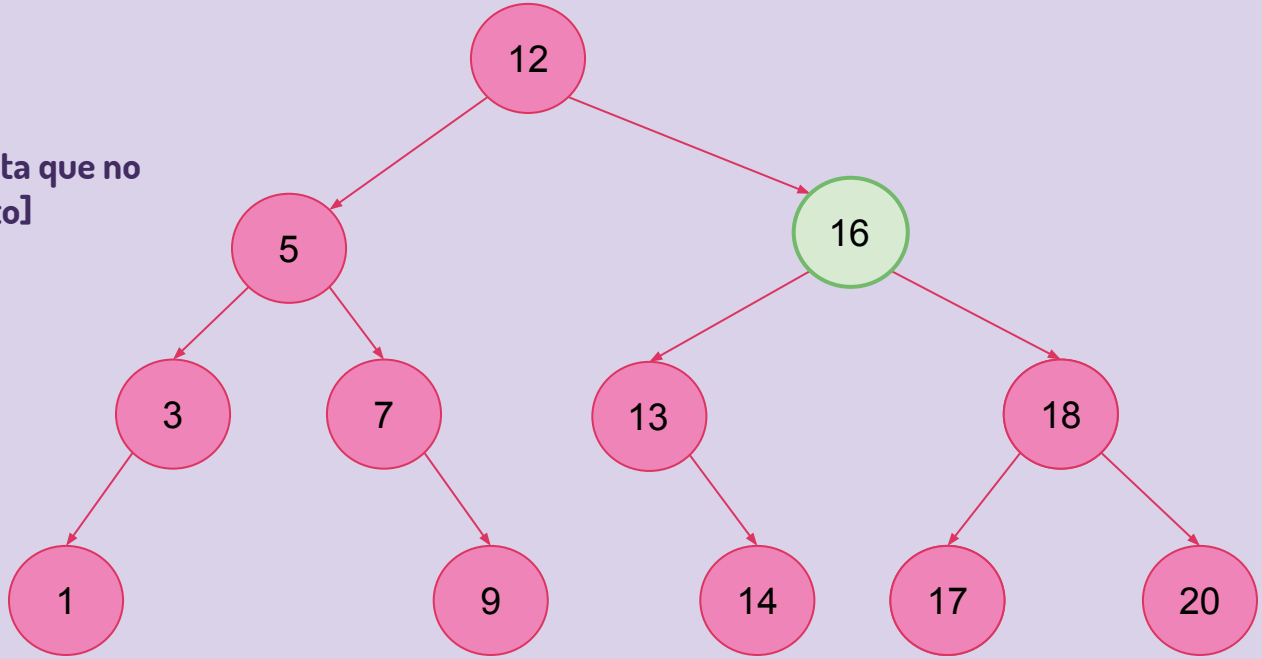
2. El nodo encontrado es el que va a reemplazar al nodo a borrar



Borrar Nodo - Nodo con dos hijos

1. Busco el sucesor inmediato:
    a. Tomo el hijo derecho inmediato
    b. Bajo por los hijos izquierdos (hasta que no haya más) [-> el 16 es el sucesor inmediato]

2. El nodo encontrado es el que va a reemplazar al nodo a borrar

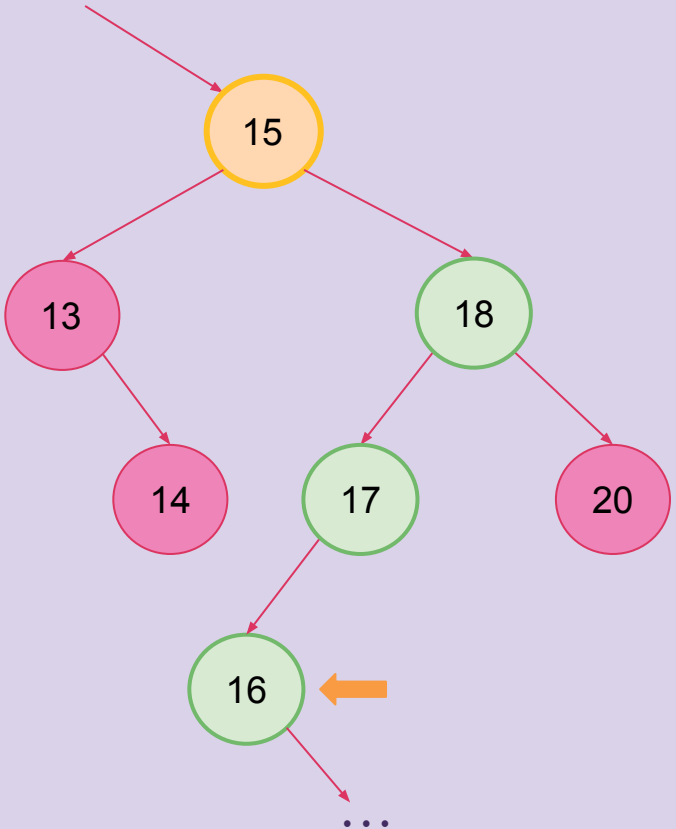


Borrar Nodo - Atención!

Qué hacemos el nodo encontrado tiene hijo derecho?

Sabemos lo siguiente:

- El 16 solo tiene UN hijo (pues si tuviera hijo izquierdo, habría que seguir bajando para encontrar el sucesor inmediato)



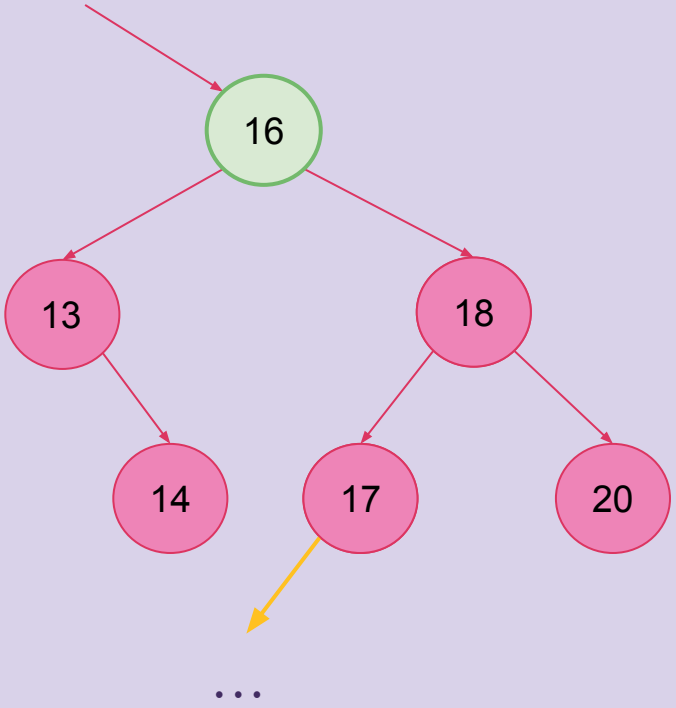
Borrar Nodo - Atención!

Qué hacemos?

En ese caso, sabemos lo siguiente:

- El 16 solo tiene UN hijo (pues si tuviera hijo izquierdo, habría que seguir bajando para encontrar el sucesor inmediato)

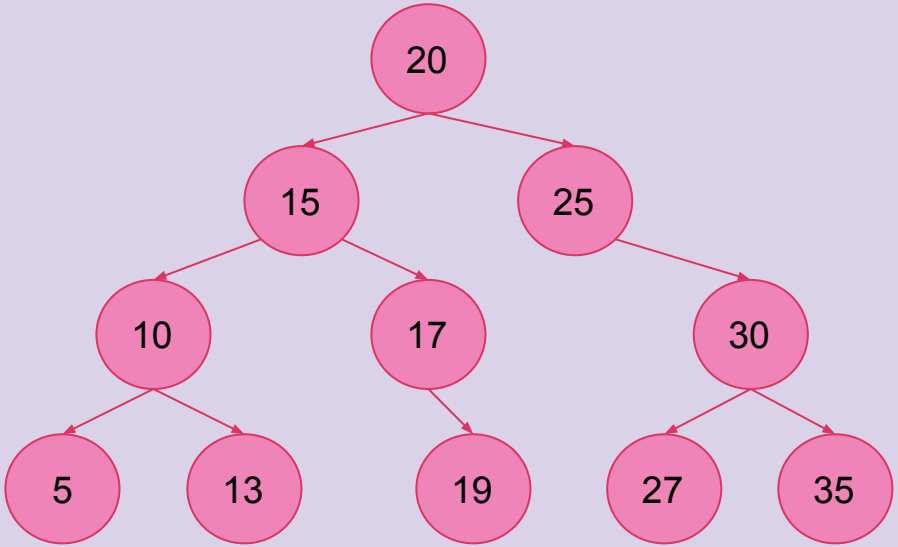
Entonces, el hijo del 16 debería linkearse con el padre (el 17) antes de realizar el reemplazo.



¿Complejidad?

Recorridos aplicados a ABBs

Que pasa si aplicamos esos recorridos a árboles ordenados?



- **Preorden:** Sirve para hacer una copia del árbol.

  [20,15,10,5,13,17,19,25,30,27,35]

- **Inorden:** Nos da un recorrido ORDENADO del árbol!

  [5,10,13,15,17,19,20,25,27,30,35]

- **Postorden:** Nos da el orden de borrado del árbol.

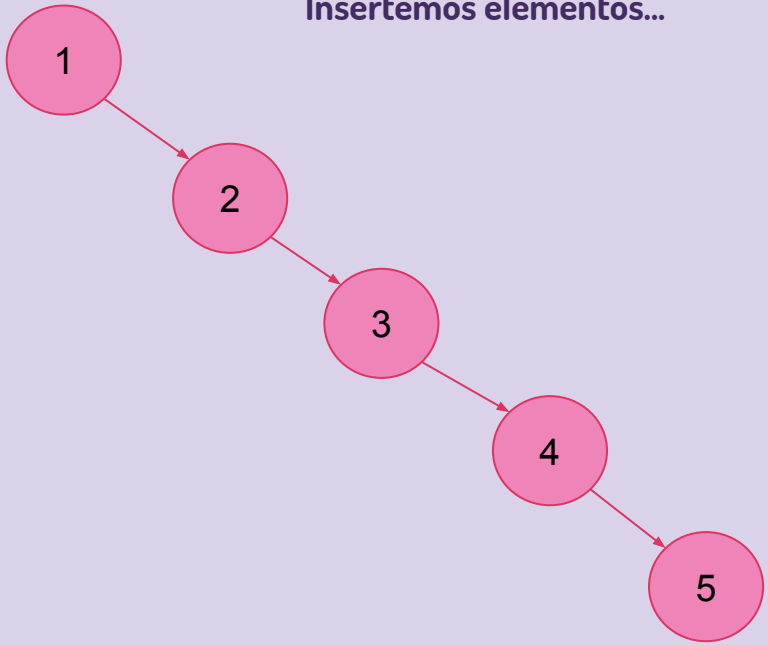
  [5,13,10,19,17,15,27,35,30,25,20]

Arboles Balanceados

Porque nos interesa balancear un árbol?

- A pesar de todo lo visto, el orden en el que se insertan los elementos en un ABB es algo de gran importancia.
- Dos ABB con los mismos elementos pero insertados en distinto orden no necesariamente son iguales
- Debido a esto los árboles pueden degenerar en listas
- Las complejidades algorítmicas de las operaciones de buscar, insertar y borrar en un ABB, en el peor de los casos, es O(N).

Insertemos elementos...



Aplicaciones

Aplicaciones

- Son utilizados para conformar índices, tanto de uno como de múltiples niveles.
- Se utilizan para mejorar numerosos algoritmos de búsqueda
- Son útiles para mantener ordenados streams de información entrante
- Estructuras de datos como TreeMap y TreeSet se implementan internamente con ABBs balanceados
- Se usa en la implementación de colas de prioridad doble (por prioridad maxima o prioridad minima)

Preguntas?

Gracias!