El conteo de votos

- Se quiere hacer una votación para supremo líder del planeta
- Todos los habitantes del planeta son posibles candidatos
- Cada persona tiene asignado un número, llamado ID
- El voto consiste en anotar el ID de tu candidato en la papeleta
- La persona cuyo ID aparezca en más papeletas gana

¿Cuál es el costo de contar los votos?

El conteo de votos

No sabemos cuantos candidatos aparecerán en los votos

¿Cómo podemos llevar cuenta solo de los ID que han salido?

Hay que considerar la eficiencia de contar un nuevo voto

El diccionario

Es un tipo de estructura que ofrece las siguientes operaciones:

- Asociar un **valor** a una **clave**, o actualizarlo
- Obtener el **valor** asociado a una **clave**

Y para ciertos casos de uso

- Eliminar una **clave** y su **valor** asociado de la estructura

El diccionario

Queremos un diccionario para contar los votos

Necesitamos que use sólo la memoria necesaria

El Árbol Binario de Búsqueda

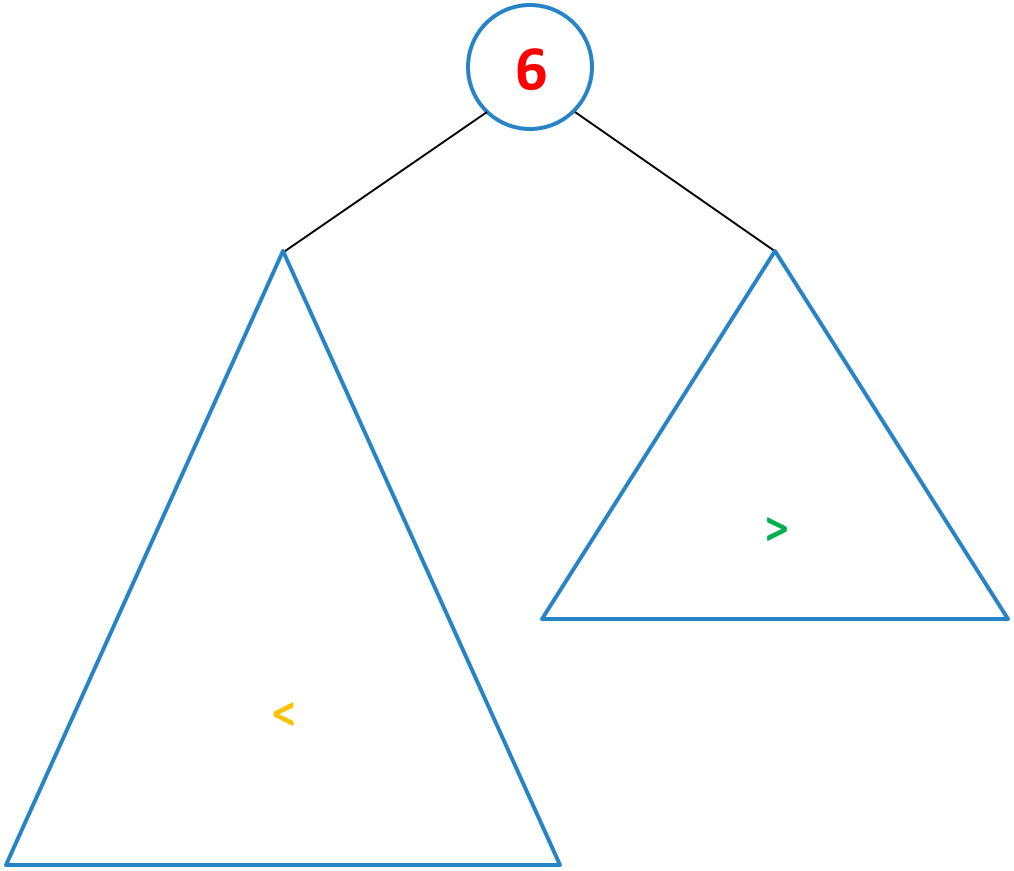
Es un **árbol binario**, con un elemento arbitrario como raíz

Los demás datos están divididos en los mayores y los menores a la raíz

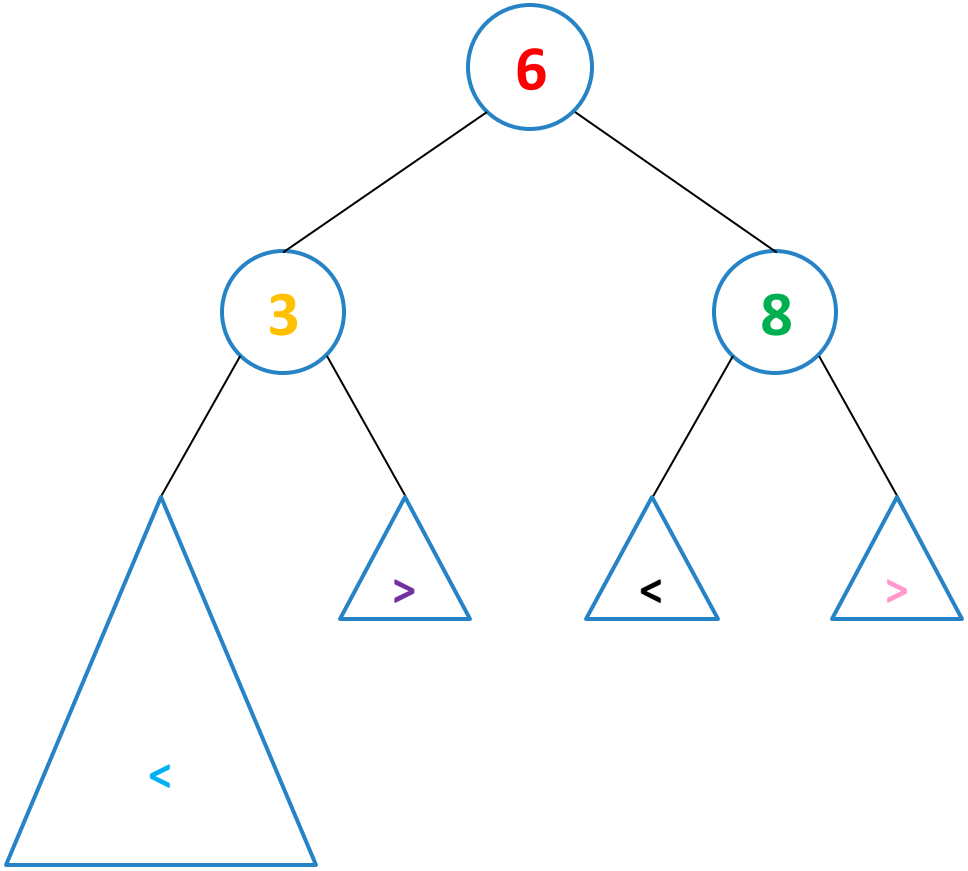
Cada uno de estos grupos está a su vez organizado como **ABB**

Los menores cuelgan a la izquierda de la raíz, y los mayores a la derecha

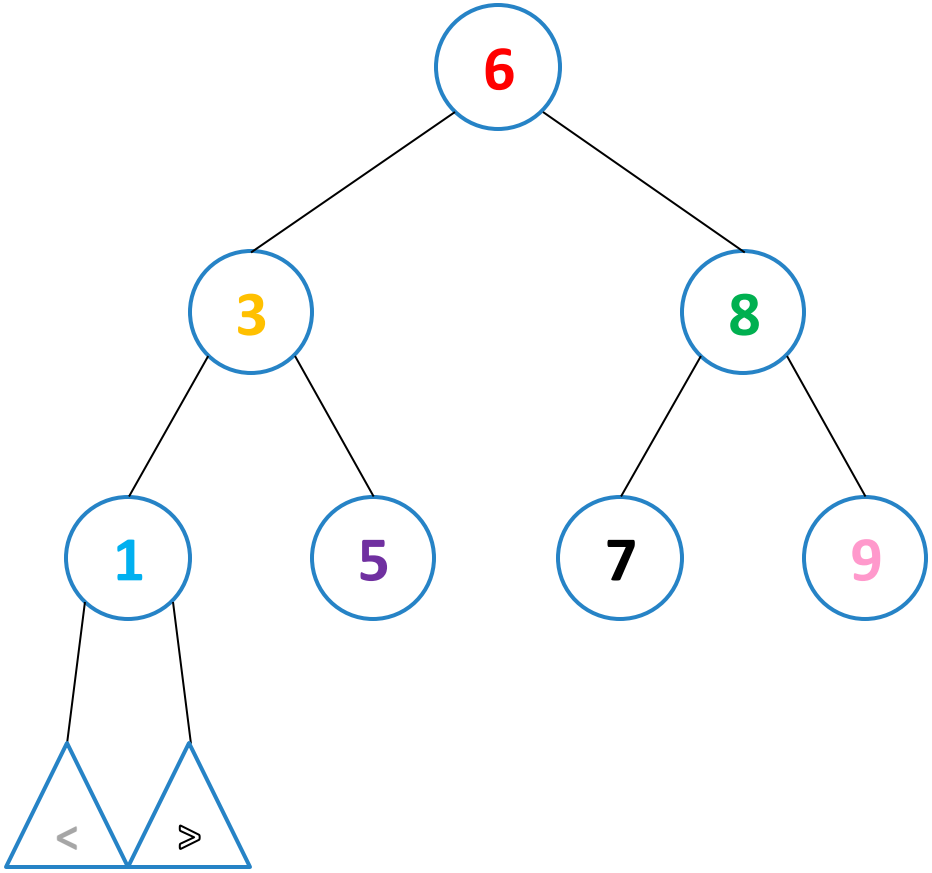
El Árbol Binario de Búsqueda



El Árbol Binario de Búsqueda



El Árbol Binario de Búsqueda



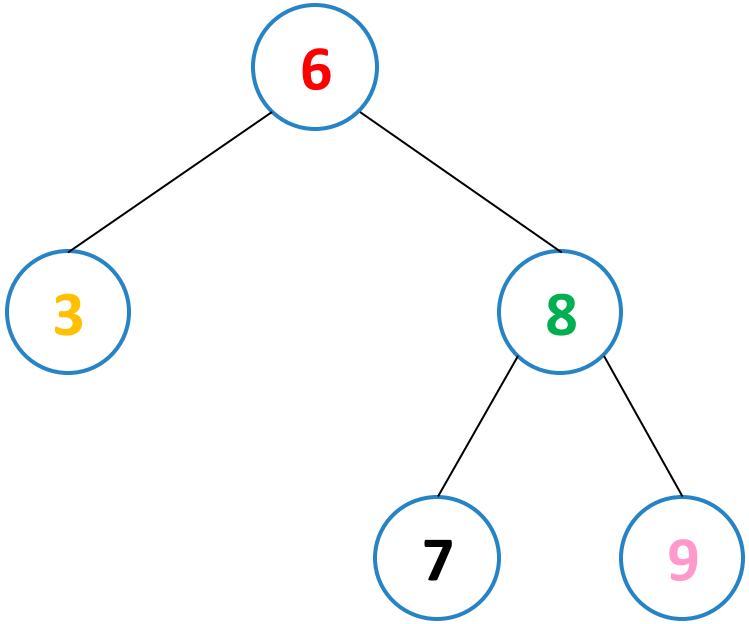
Operaciones del ABB

¿Cómo se busca un elemento en el árbol?

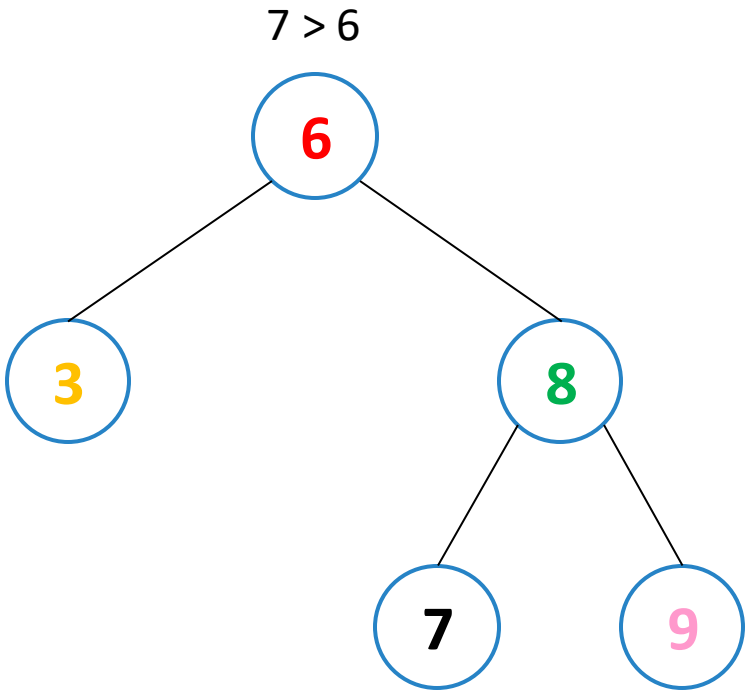
¿Qué pasa si queremos insertar un nuevo dato?

Tratemos de aprovechar que la estructura es recursiva

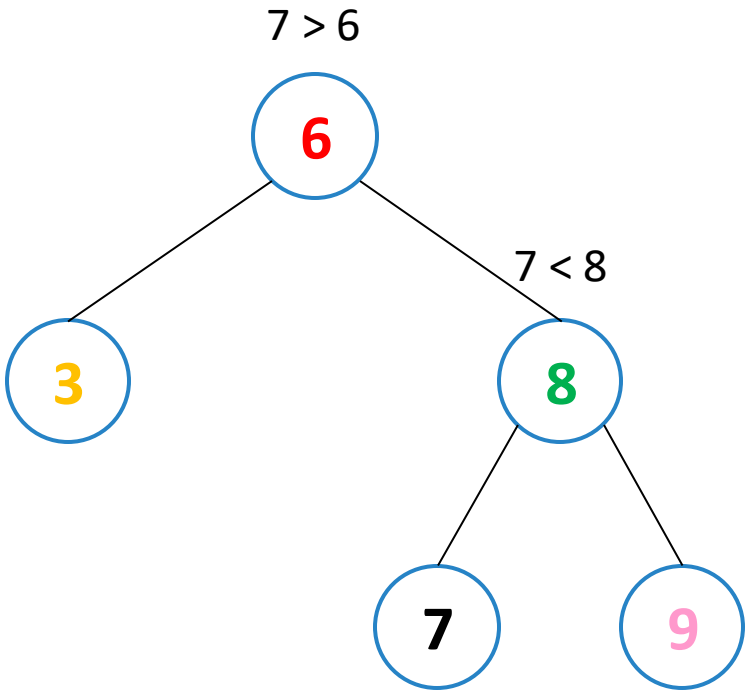
Busquemos el 7



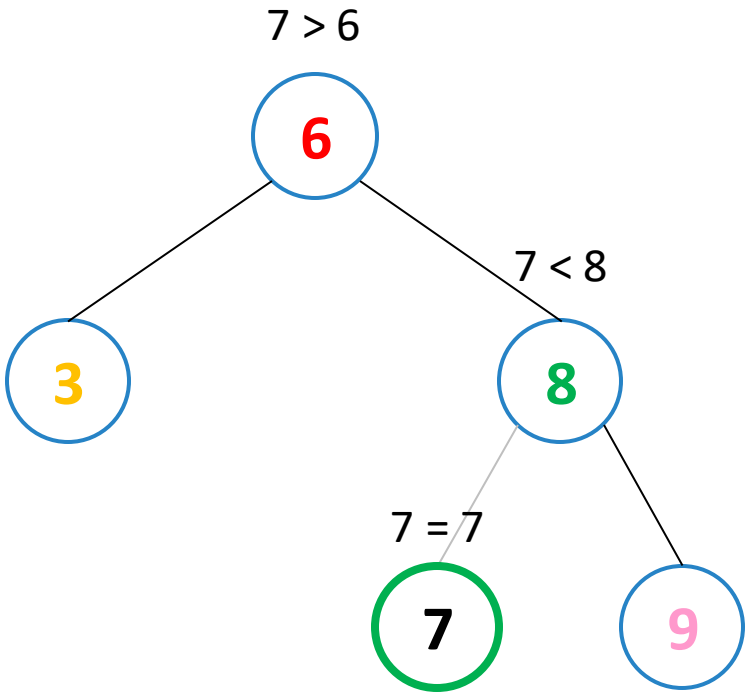
Busquemos el 7



Busquemos el 7

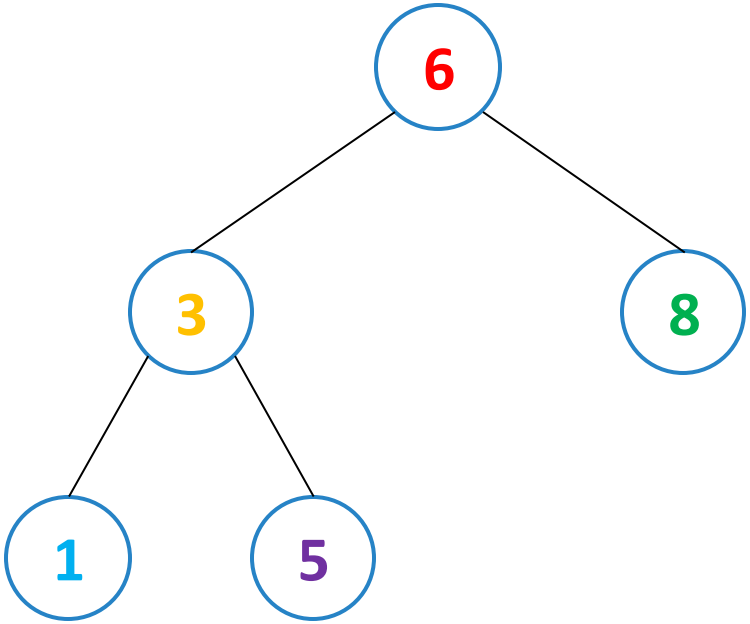


Busquemos el 7

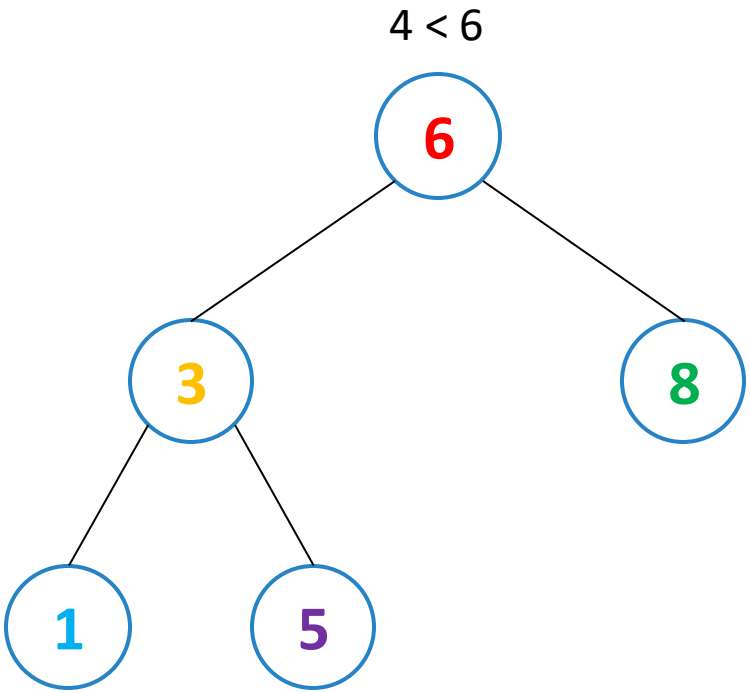


```
search(A, k):
    if A = ∅  o  A.k = k:
        return A
    else if k < A.k:
        return search(A.left, k)
    else:
        return search(A.right, k)
```

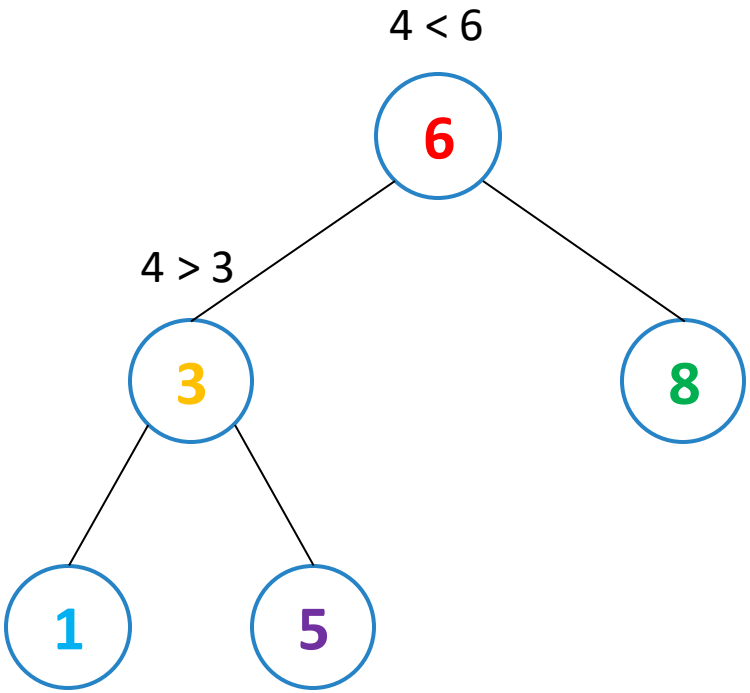
Insertemos el 4



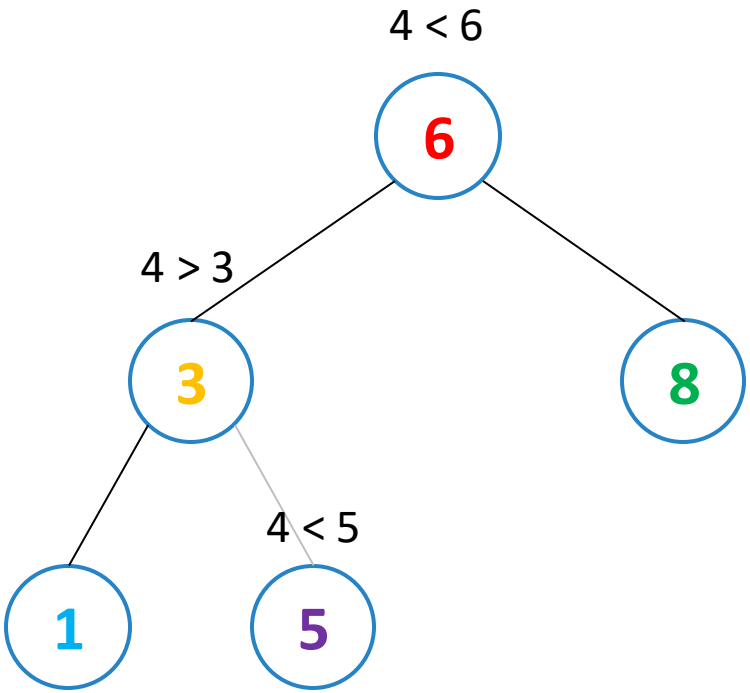
Insertemos el 4



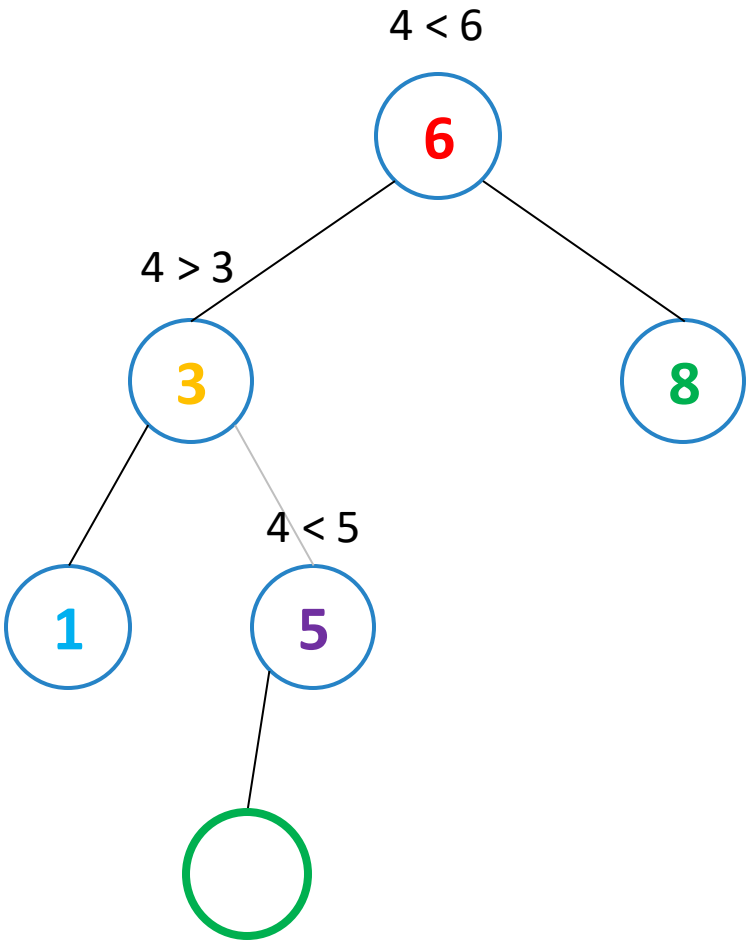
Insertemos el 4



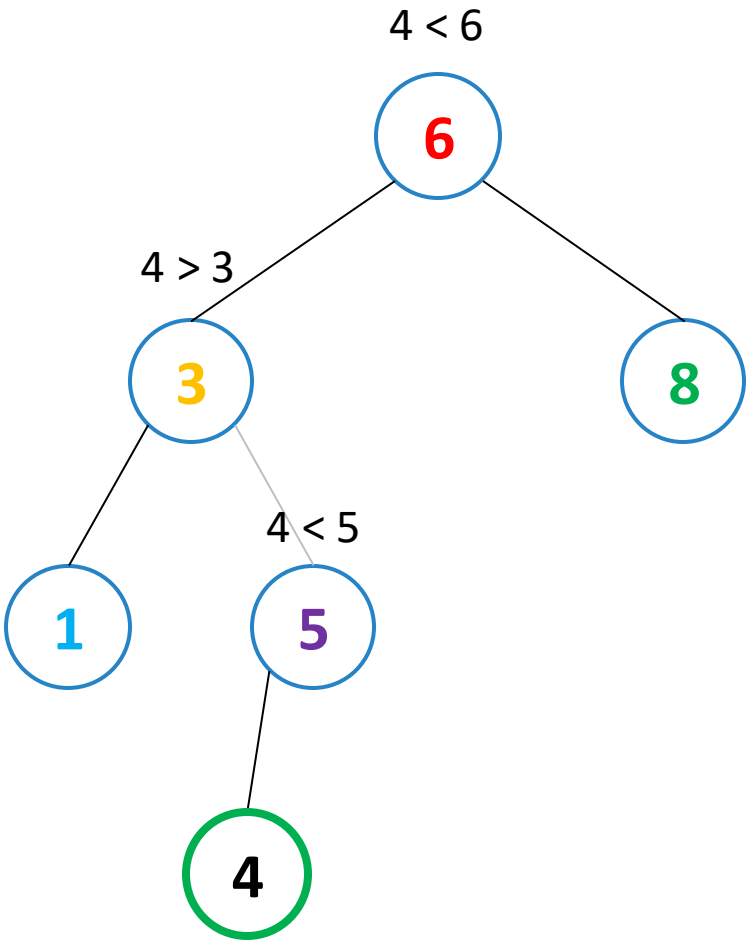
Insertemos el 4



Insertemos el 4



Insertemos el 4



$\boldsymbol{insert}(\boldsymbol{A}, \boldsymbol{k}, \boldsymbol{v})$:

$\boldsymbol{B} \leftarrow \boldsymbol{search}(\boldsymbol{A}, \boldsymbol{k})$

$\boldsymbol{B}.\boldsymbol{k} \leftarrow \boldsymbol{k}, \qquad \boldsymbol{B}.\boldsymbol{v} \leftarrow \boldsymbol{v}$

Eliminación

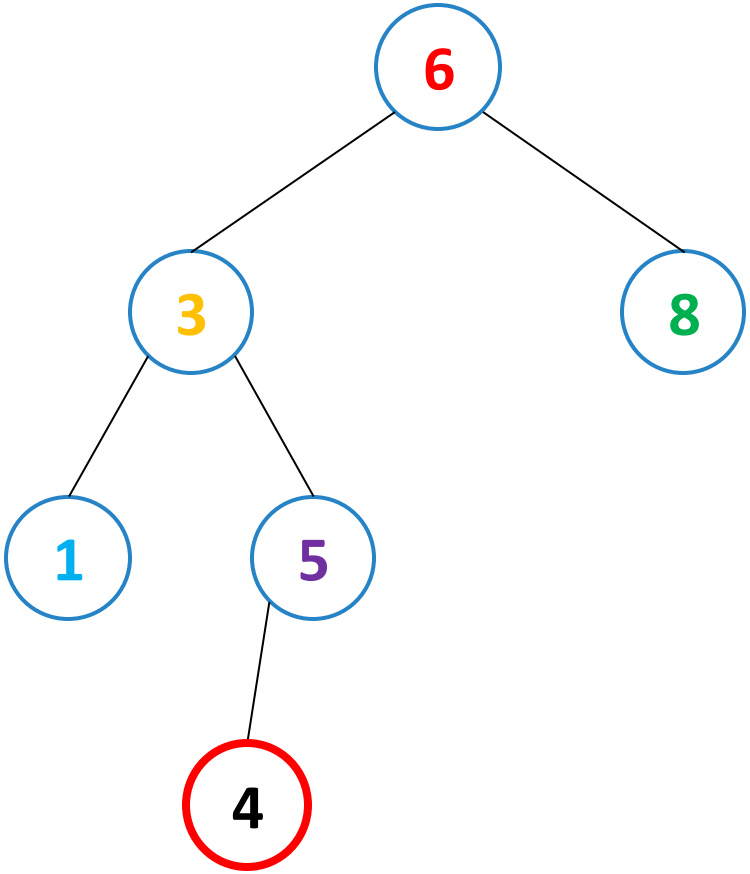
Se ha ingresado un dato erróneo al árbol

Si el dato quedó en una hoja, o tiene un solo hijo, eliminarlo es trivial

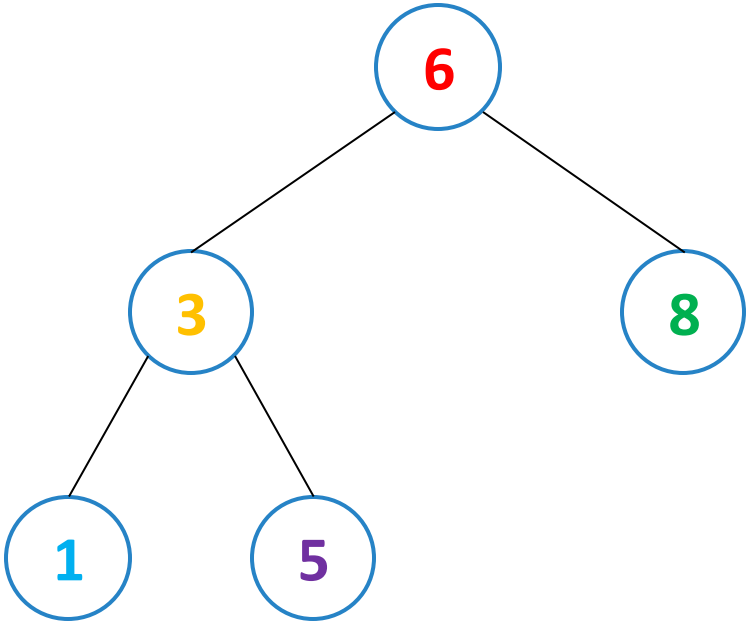
Sino, ¿Cómo podemos eliminarlo sin romper la estructura?

¿Podremos reemplazarlo por otro nodo del árbol? ¿Cuál?

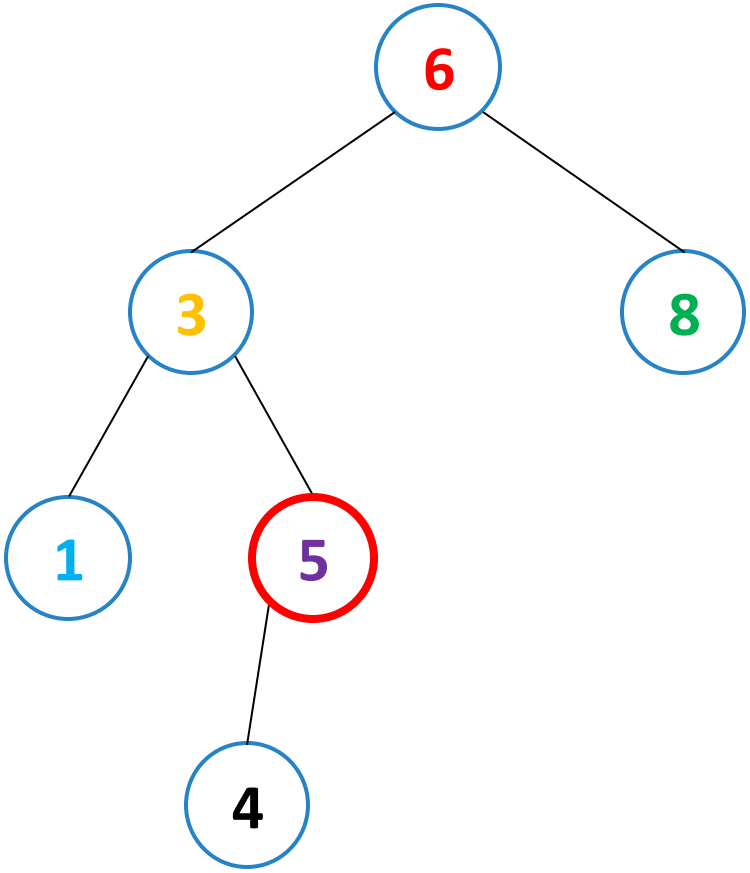
Eliminemos el 4



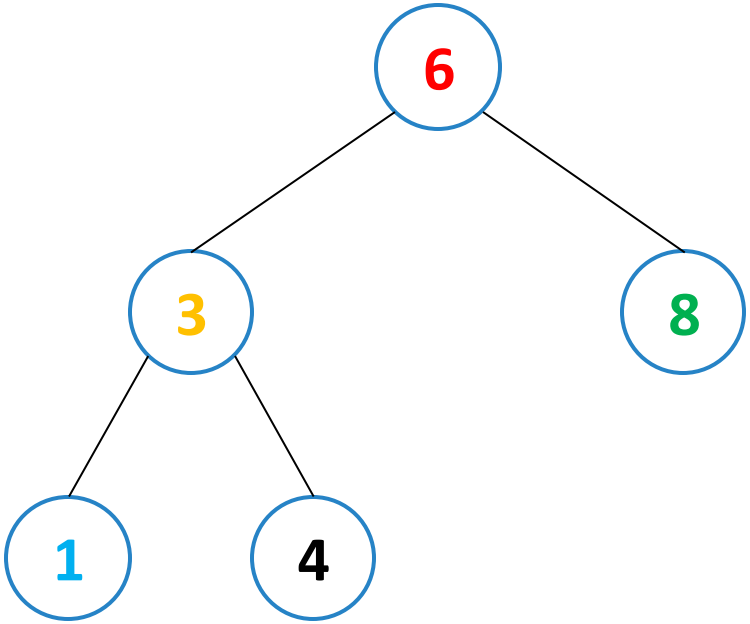
Eliminemos el 4



Eliminemos el 5



Eliminemos el 5



Antecesor y Sucesor

Si los nodos estuvieran ordenados en una lista según su k

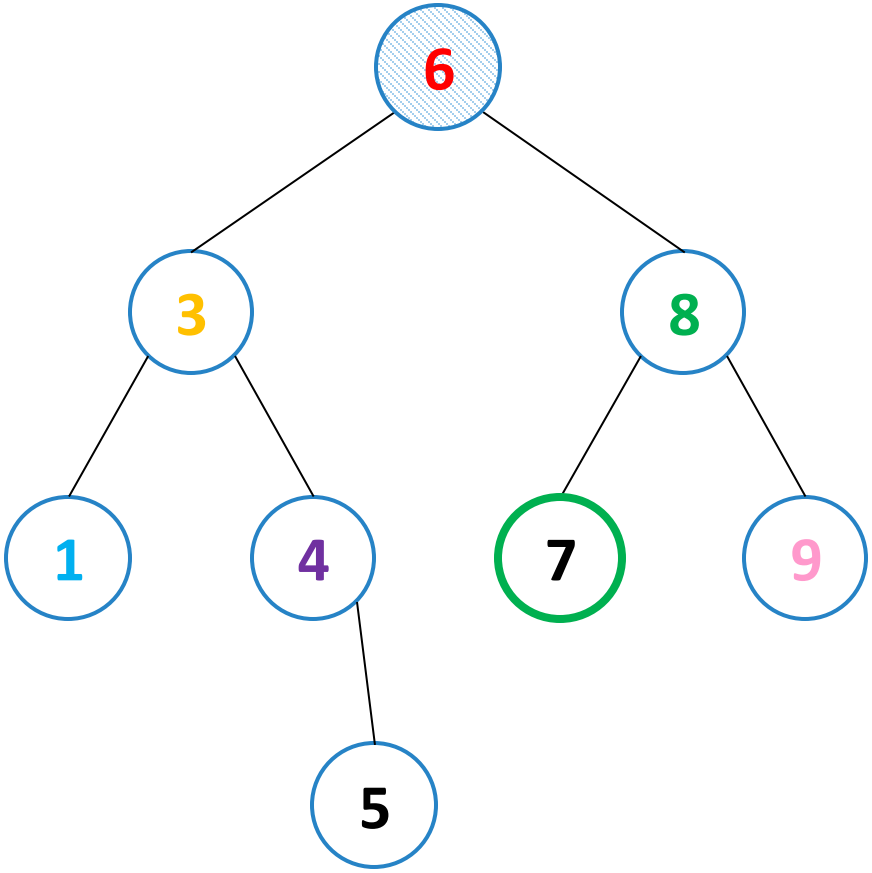
- El **sucesor** de un nodo es el siguiente de la lista
- El **antecesor** de un nodo es el anterior en la lista

¿Cómo podemos encontrar estos elementos dentro del árbol?

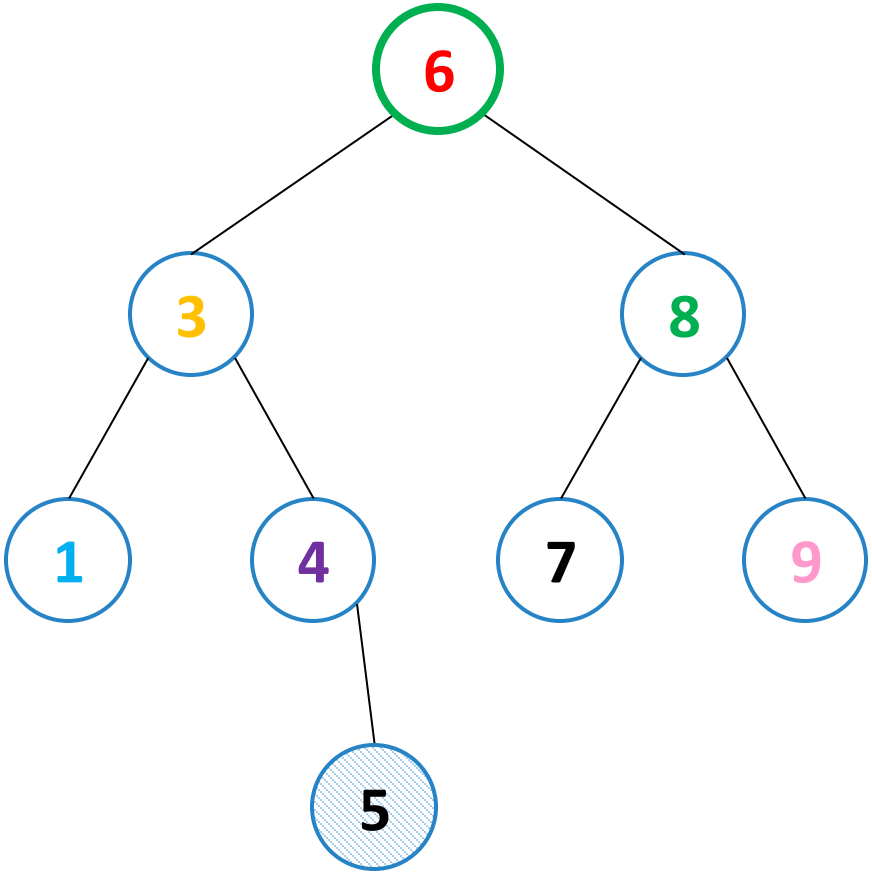
```
min(A):
    if A.left = ∅:
        return A
    else:
        return min(A.left)
```

```
succesor(A):
    if A.right ≠ ∅:
        return min(A.right)
    B ← A.parent
    while A = B.right:
        A ← B
        B ← A.parent
    return B
```

Busquemos el sucesor del 6



Busquemos el sucesor del 5



$delete(A, k)$:

$D \leftarrow search(A, k)$

if D *es hoja*, $D \leftarrow \emptyset$

else if D *tiene un solo hijo* H, $D \leftarrow H$

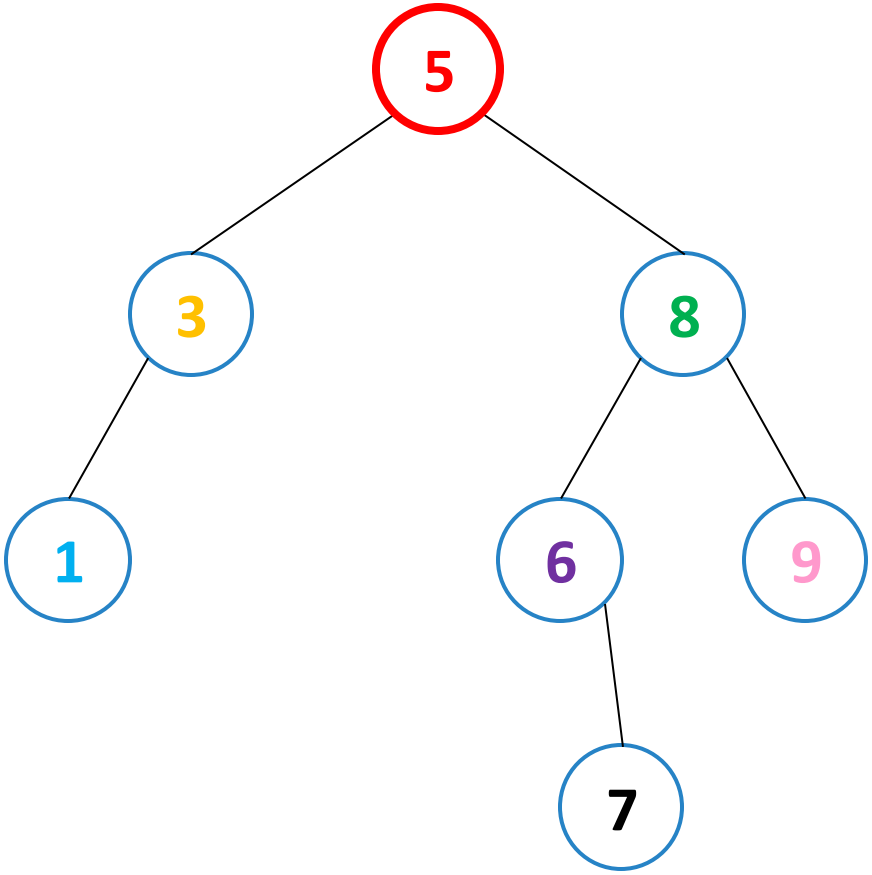
else:

$R \leftarrow succesor(D)$

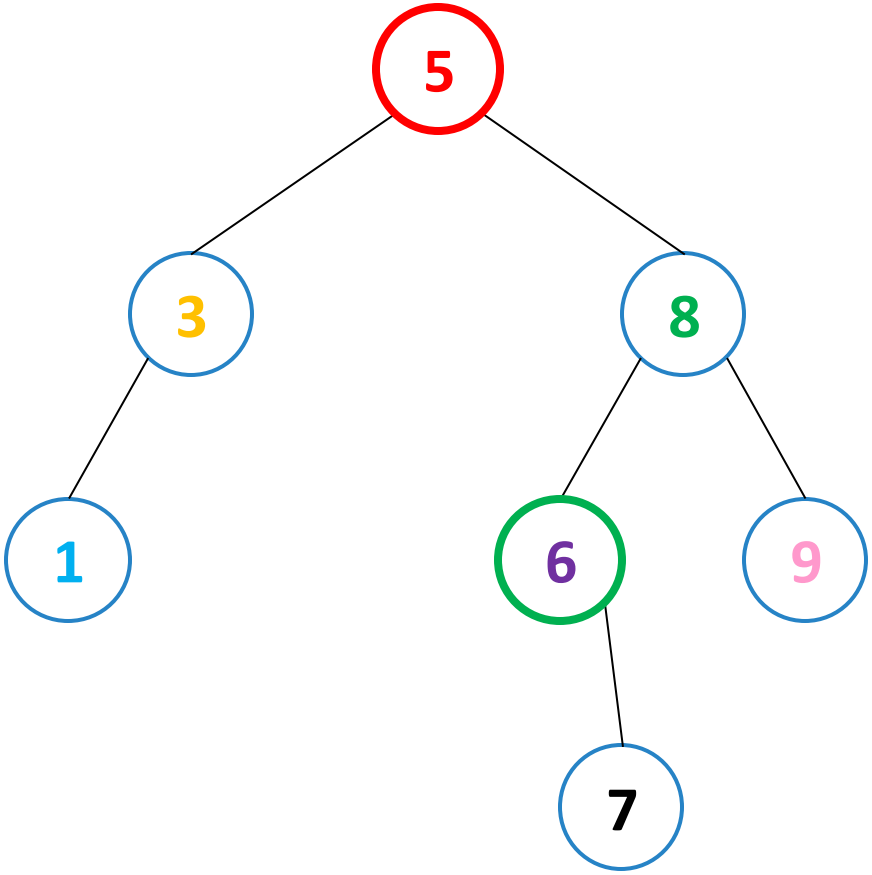
$D.k \leftarrow R.k$, $D.v \leftarrow R.v$

$delete(R, R.k)$

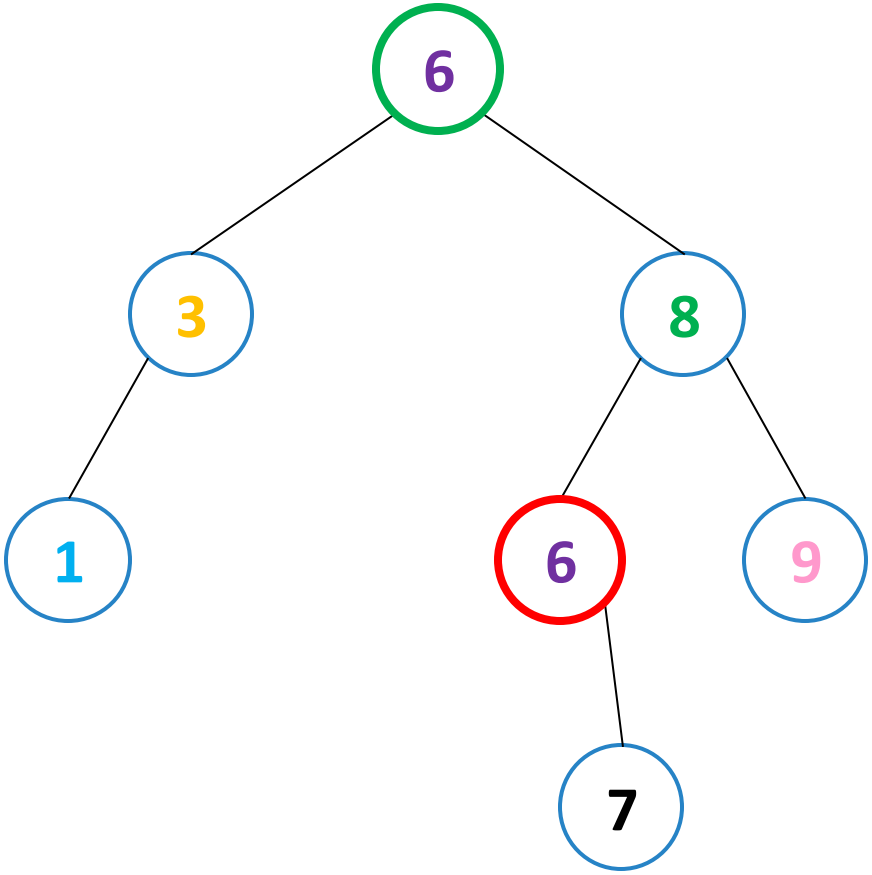
Eliminemos el 5



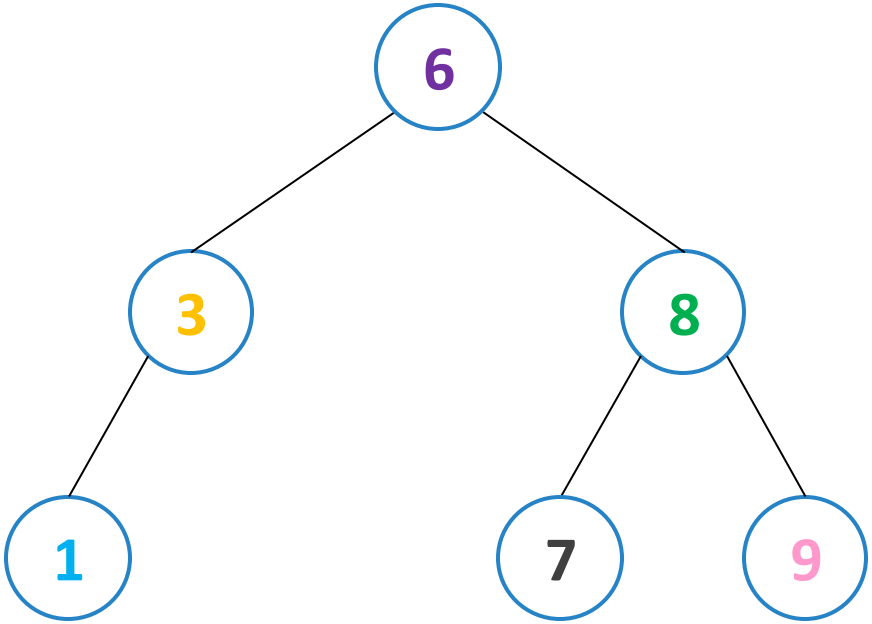
Eliminemos el 5



Eliminemos el 5



Eliminemos el 5



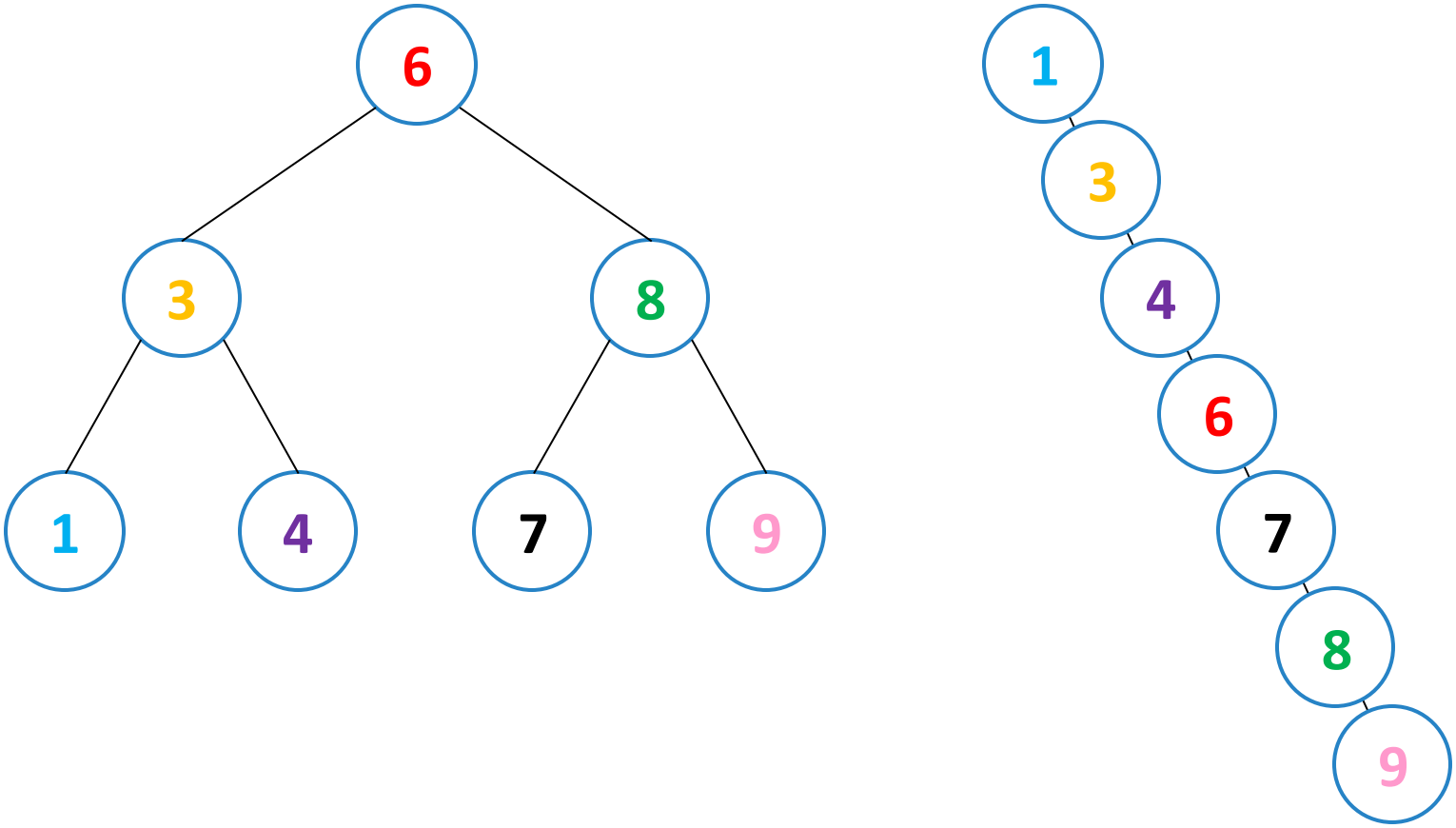
Raíces y raíces

¿Hay raíces mejores que otras?

¿Qué pasa con el árbol si no queda un buen dato como raíz?

¿Cómo varía la complejidad de las operaciones?

Mismos datos, distinto árbol



¡Todo depende del orden de inserción!