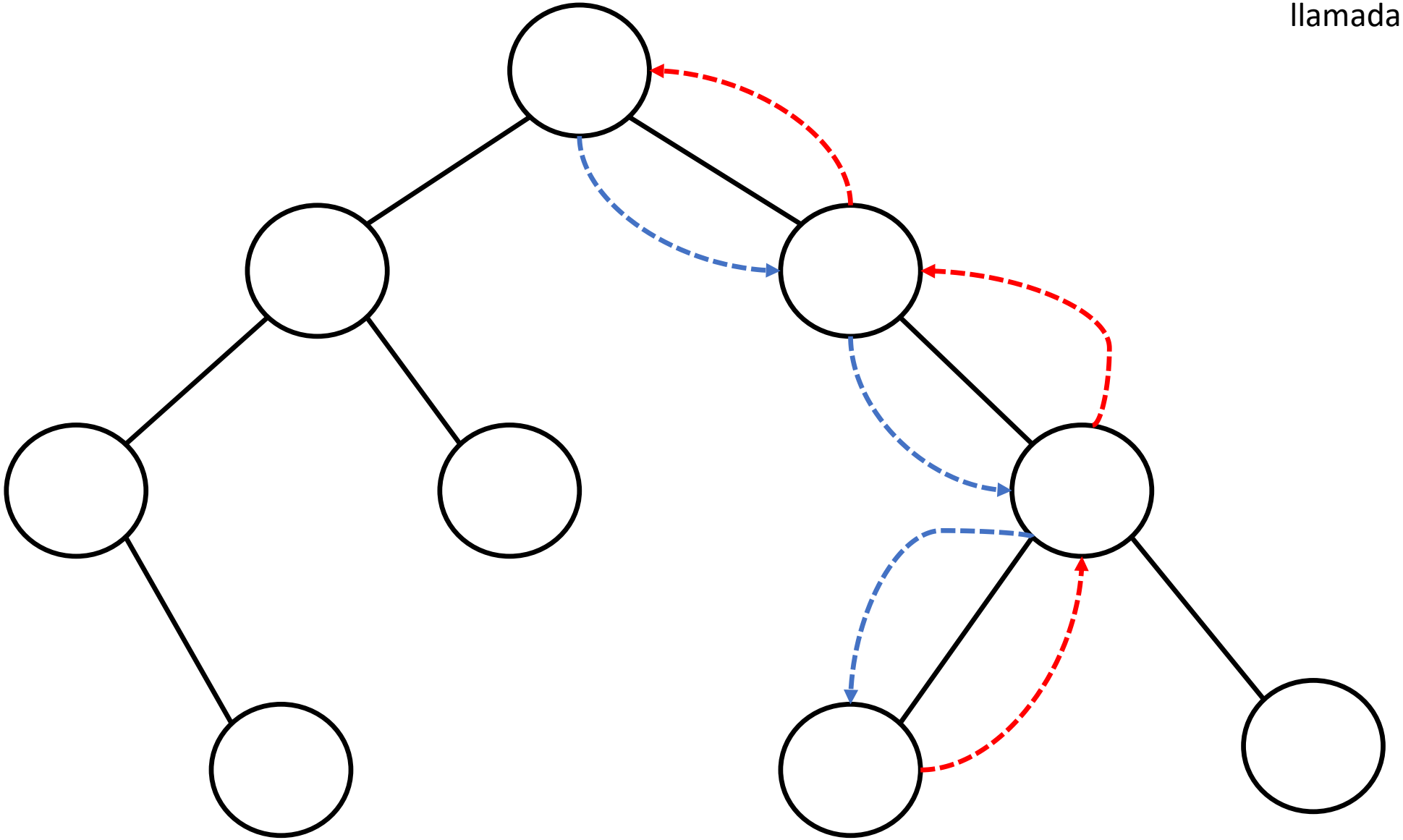
Algoritmos de árboles binarios de búsqueda basados en su propiedad



En TNodoArbolBinarioBusqueda elMetodo (unaClave): de algún tipo

```
COM
    // eventualmente, inicializar variables locales, por ejemplo resultado = de algún tipo
Si unaClave == etiqueta entonces
   // unaClave está en el árbol, eventualmente hacer algo
Sino
  Si unaClave < etiqueta entonces   // hay que buscar en el subárbol izquierdo, si lo tiene
      Si hijoIzq != nulo entonces      // tiene subárbol izquierdo, llamar recursivamente
                    resultado = hijoIzq.elMetodo(unaClave)
      sino    // no tiene subárbol izquierdo
               // unaClave no está en el árbol, eventualmente hacer algo
      fin si
   Sino        // unaClave es mayor que etiqueta, hay que buscar en el subárbol derecho, si lo tiene
      Si hijoDer != nulo entonces      // tiene subárbol izquierdo, llamar recursivamente
                    resultado = hijoDer.elMetodo(unaClave)
      sino    // no tiene subárbol derecho
              // unaClave no está en el árbol, eventualmente hacer algo
      fin si
   fin si
Fin si
    // eventualmente preparar el resultado a devolver
devolver resultado
FIN
```

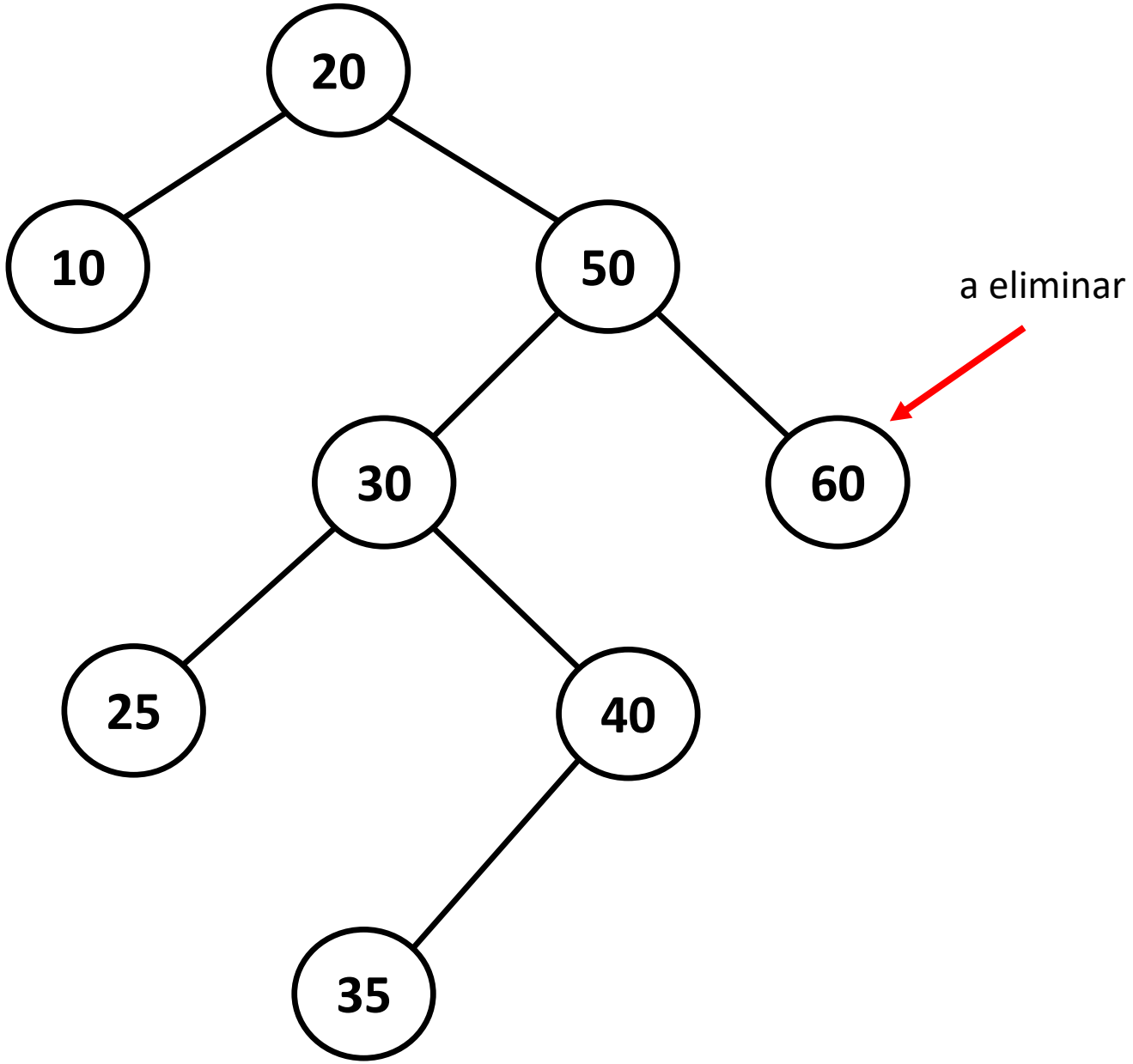
Una u otra llamada, nunca las dos. En cada paso descarta una parte del espacio de búsqueda

```
En TNodoArbolBinarioBusqueda nivelDeLaClave (unaClave):  de tipo entero
COM
de tipo entero resultado
Si unaClave == etiqueta entonces
  devolver 0 // o devolver 1, depende de la convención usada
Sino
  Si unaClave < etiqueta entonces
      Si hijoIzq != nulo entonces
              resultado = hijoIzq. nivelDeLaClave(unaClave)
      sino
              devolver -1
      fin si
   Sino
      Si hijoDer != nulo entonces
              resultado = hijoDer. nivelDeLaClave(unaClave)
      sino
              devolver -1
      fin si
   fin si
Fin si
Si resultado < 0 entonces resultado = resultado -1
Sino resultado = resultado + 1 fin si
devolver resultado
FIN
```

Algoritmos de árboles binarios de búsqueda basados en su propiedad: búsqueda, búsqueda e inserción, búsqueda y eliminación

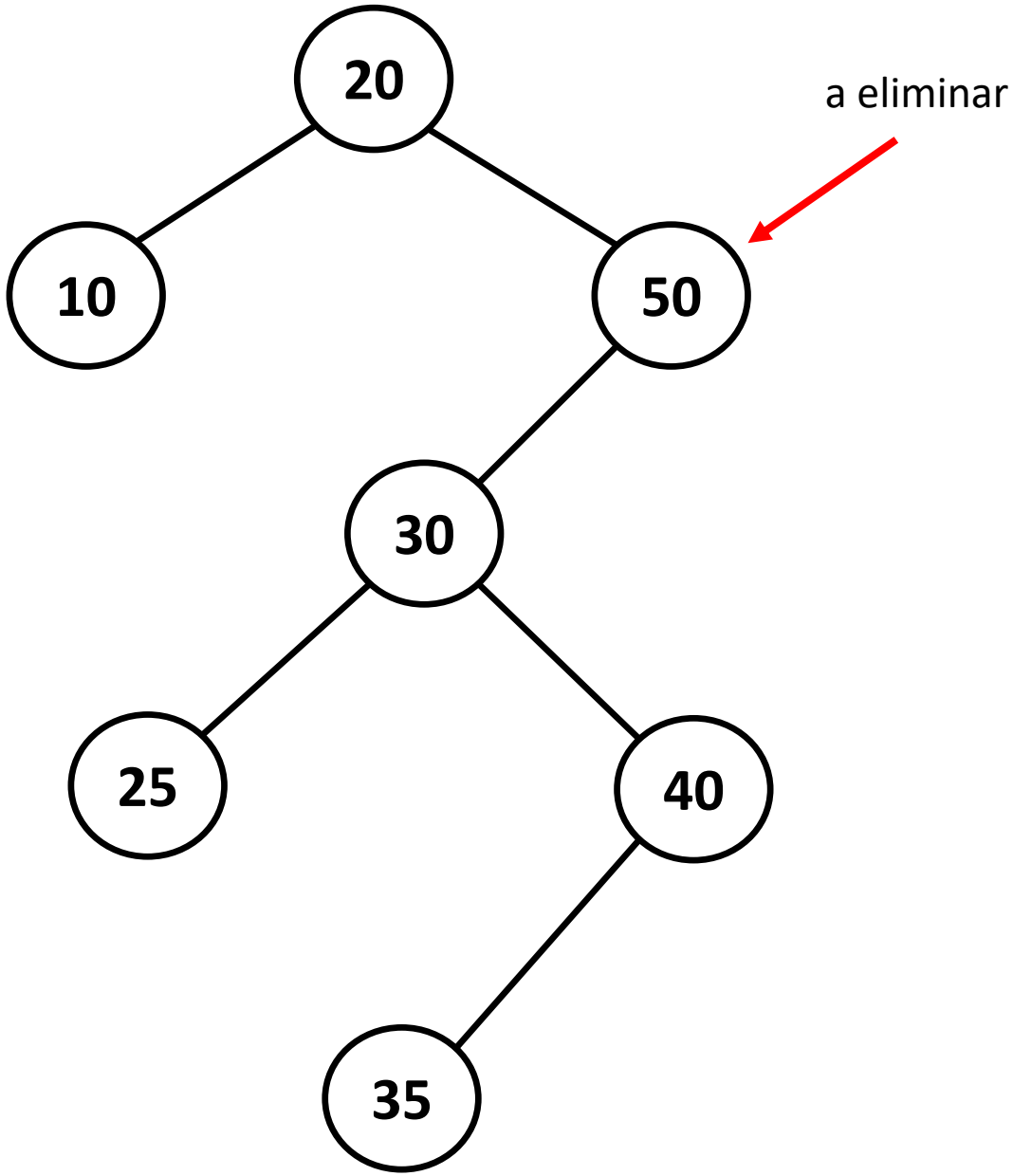
Eliminación en árboles binarios de búsqueda

Hoja, se elimina



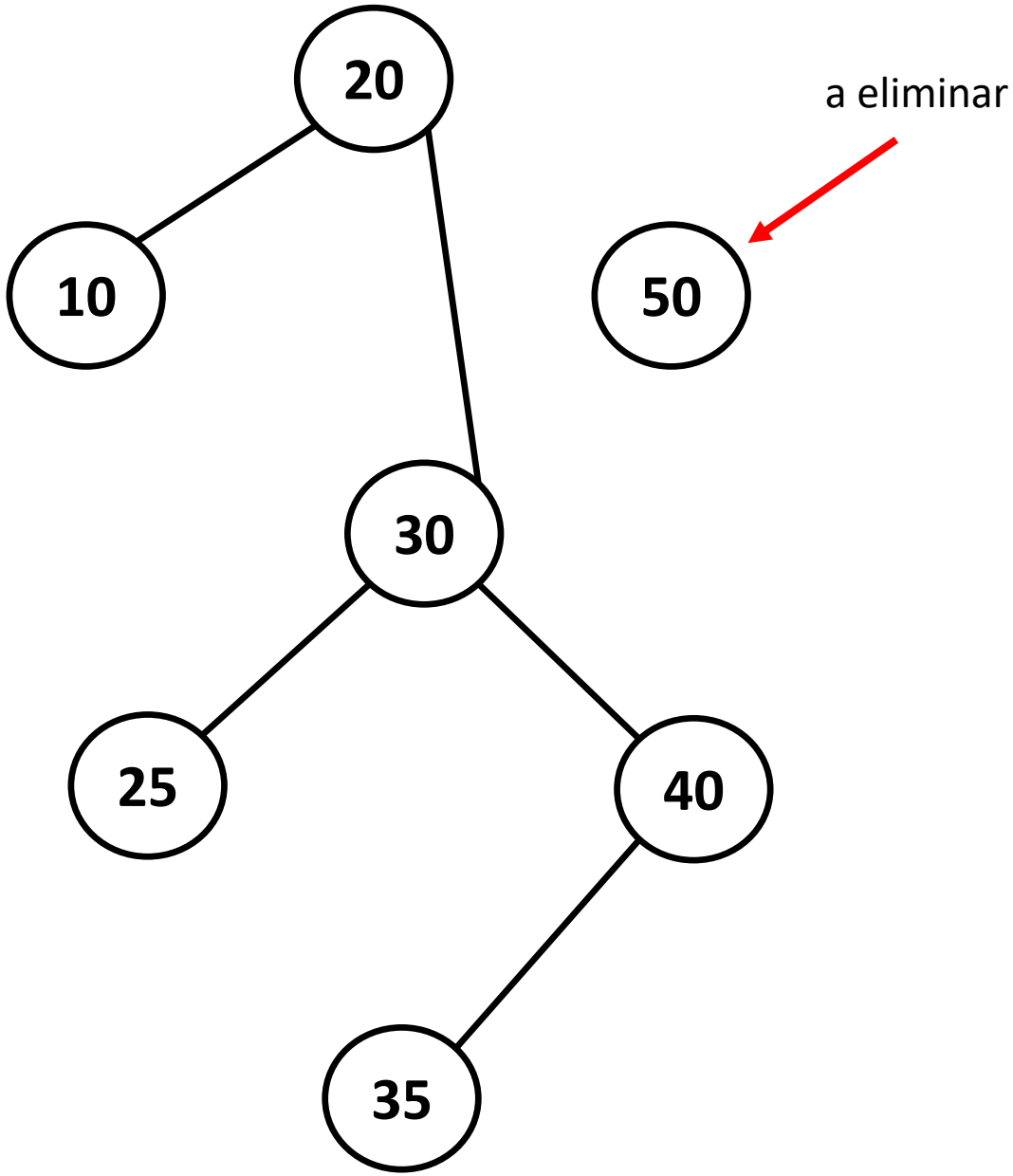
Eliminación en árboles binarios de búsqueda

Le falta un hijo, se le pasa al padre el hijo no nulo.



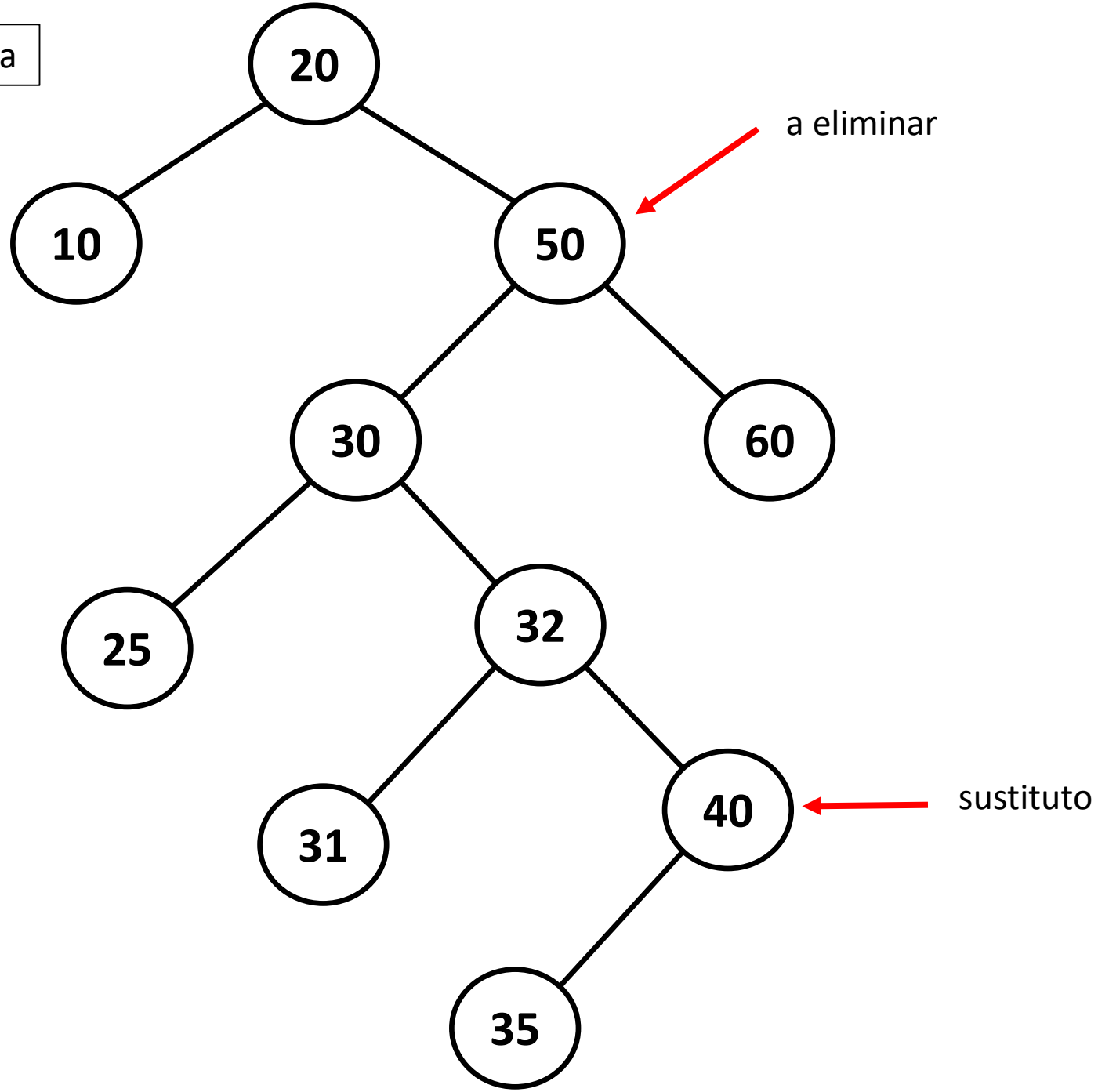
Eliminación en árboles binarios de búsqueda

Le falta un hijo, se le pasa al padre el hijo no nulo.



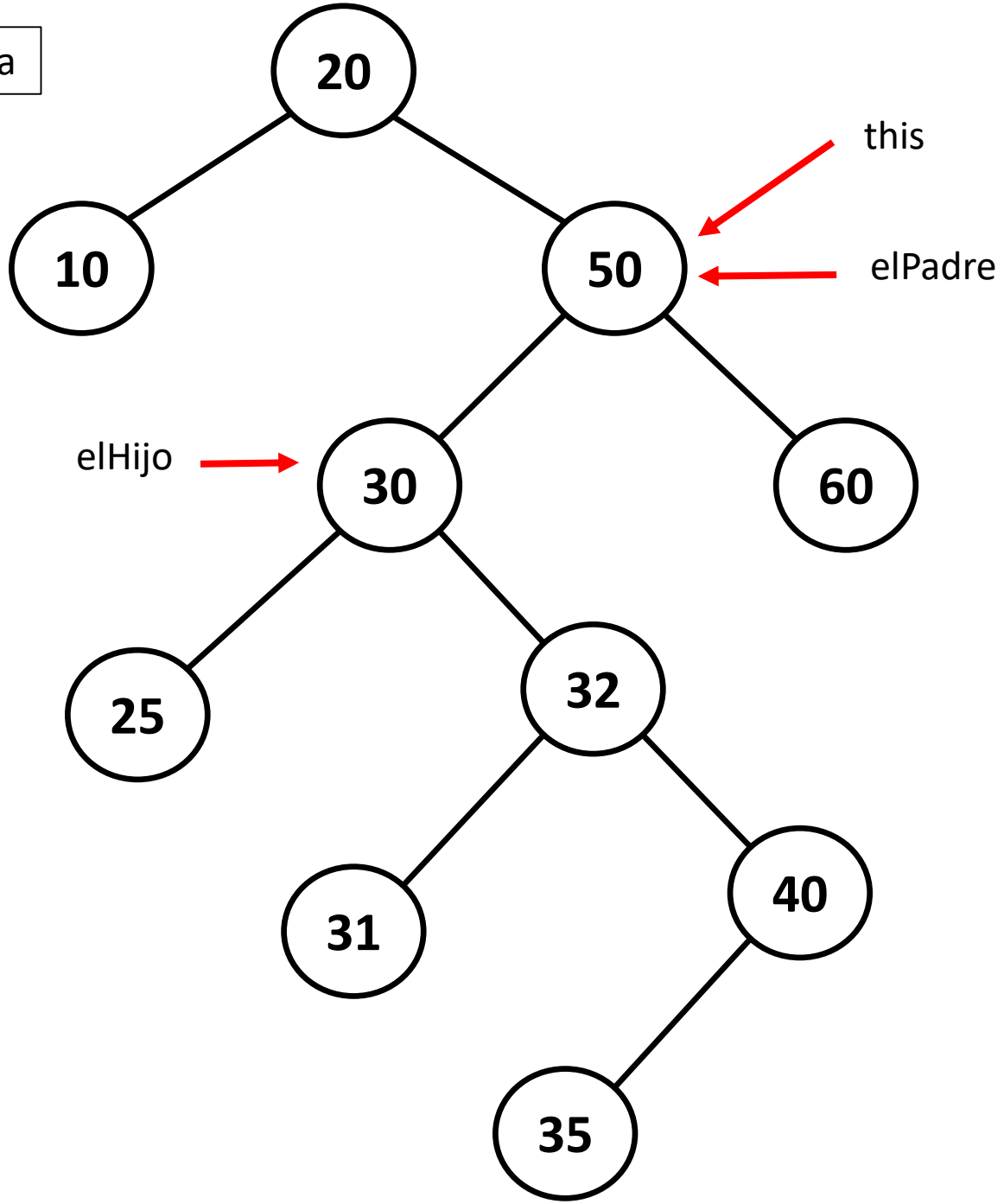
Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



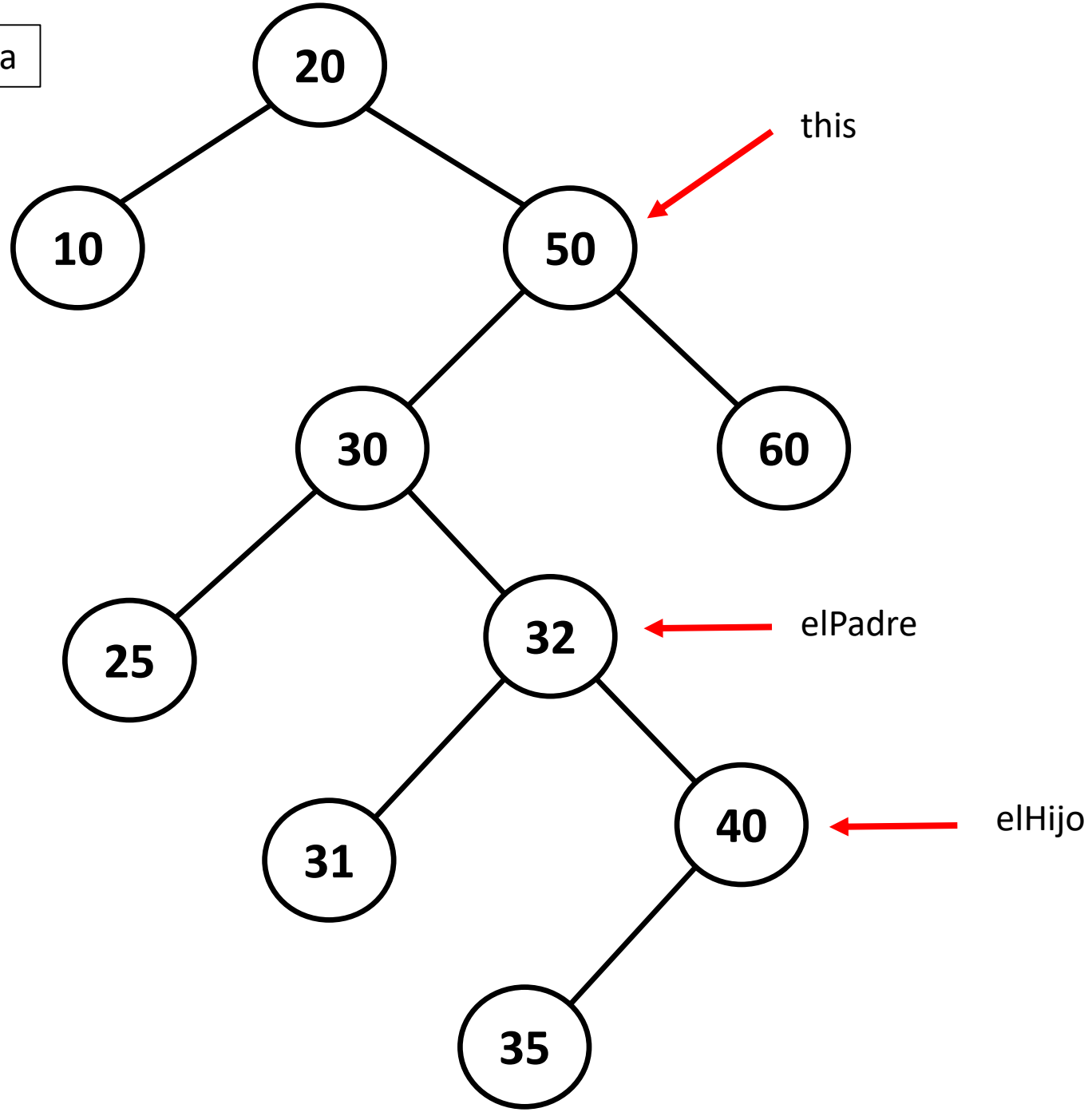
Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



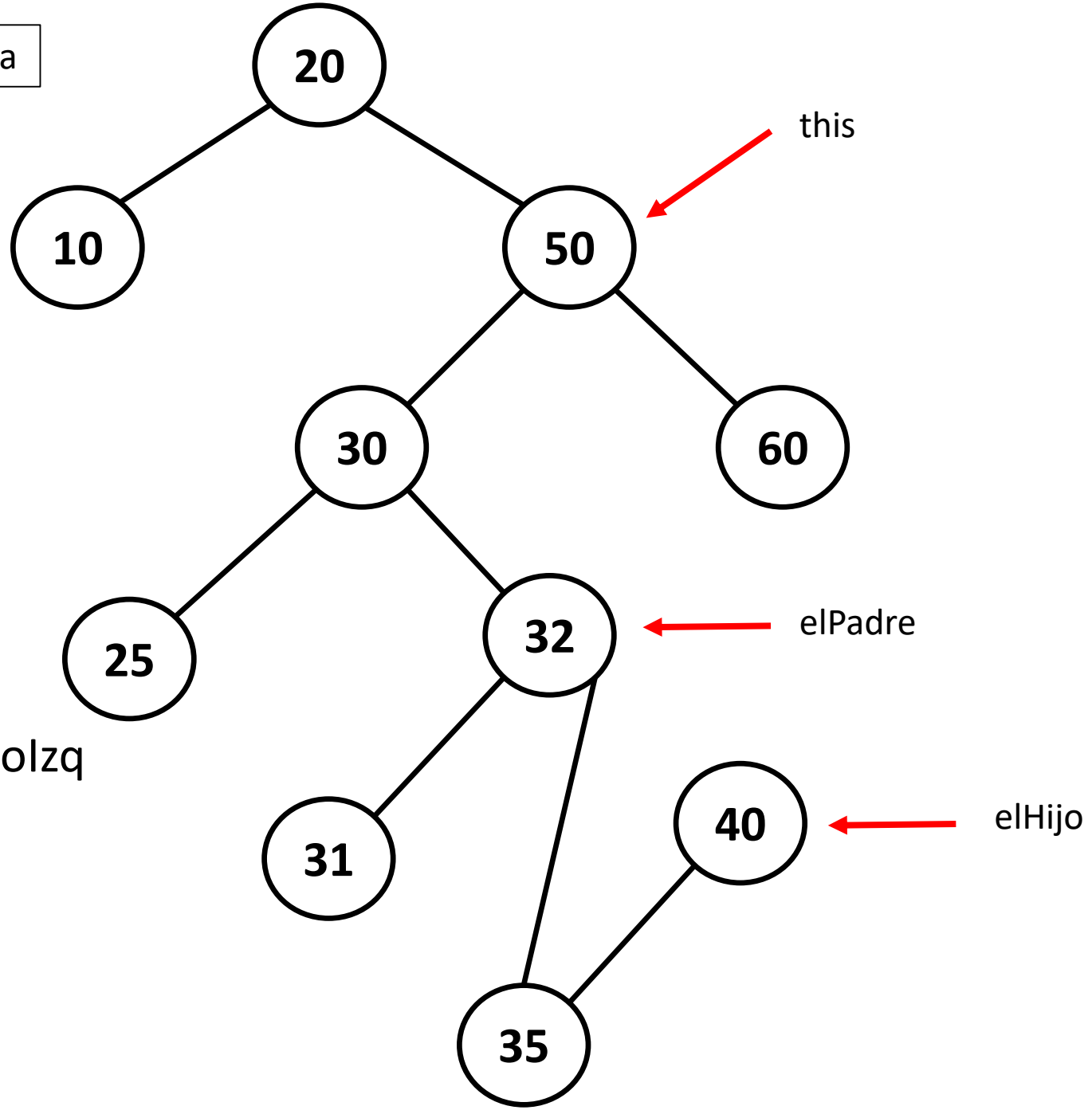
Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



Eliminación en árboles binarios de búsqueda

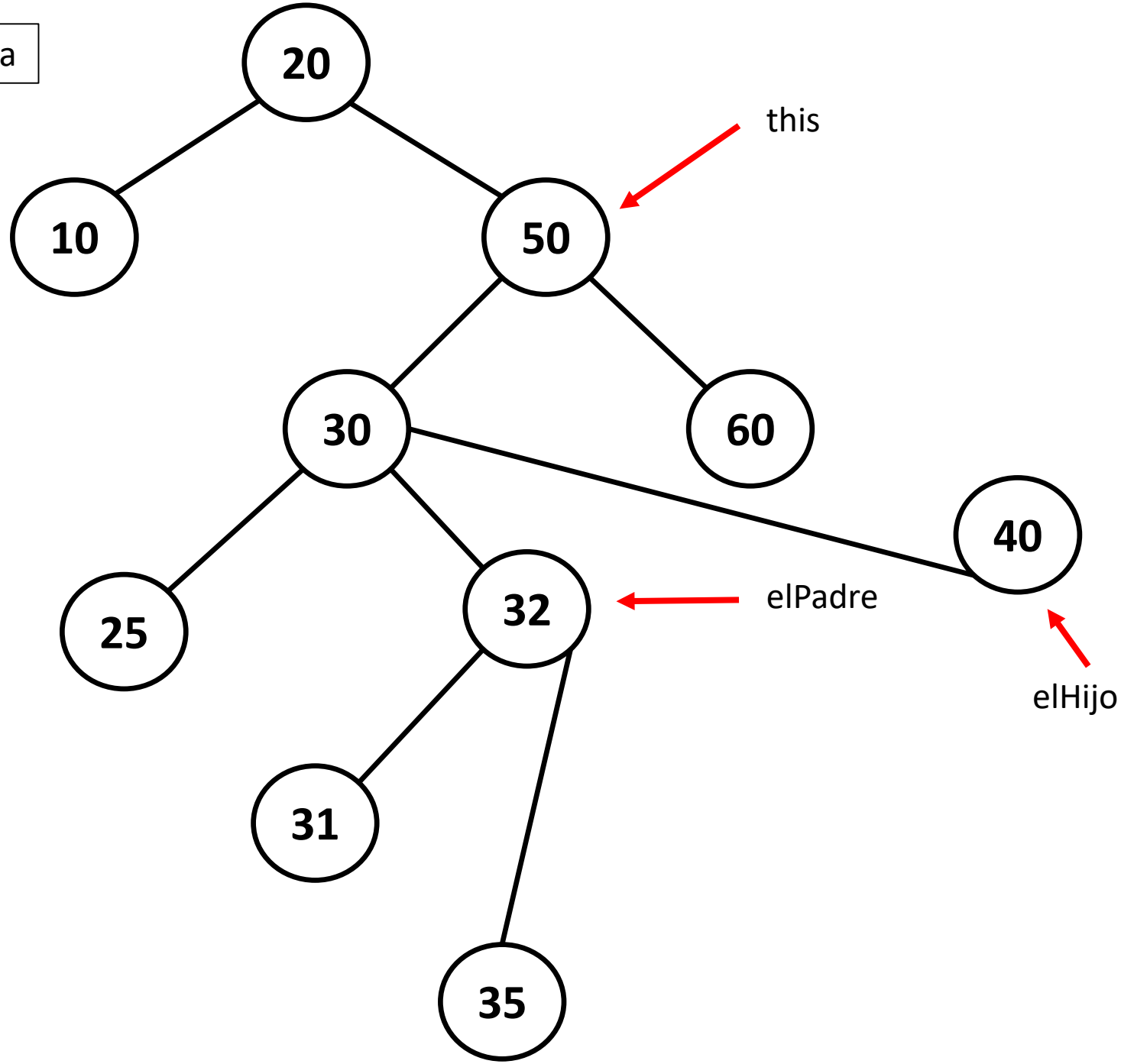
Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



elPadre.hijoDer ← this.elHijo.hijoIzq

Eliminación en árboles binarios de búsqueda

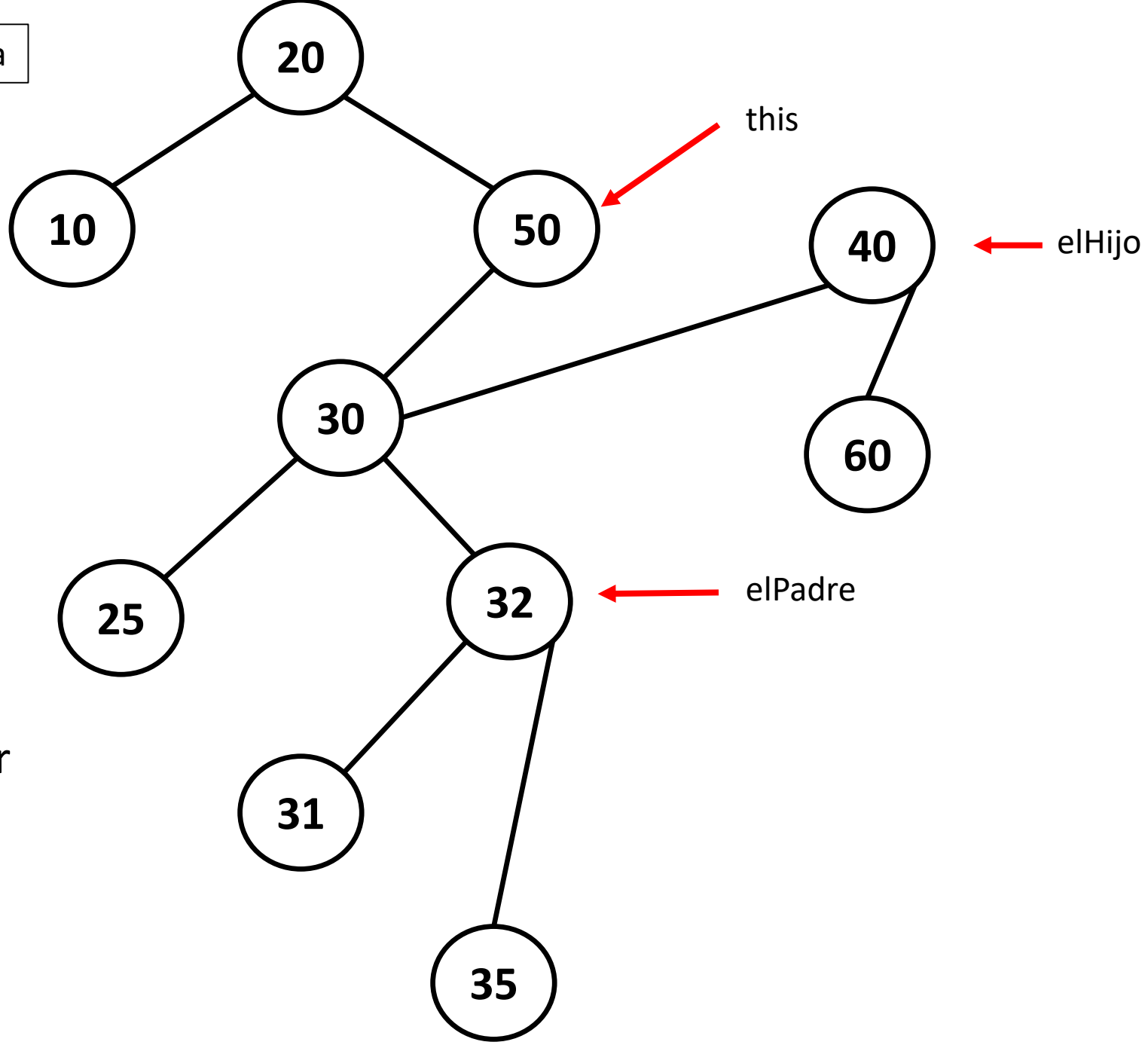
Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



elHijo.hijoIzq ← this.hijoIzq

Eliminación en árboles binarios de búsqueda

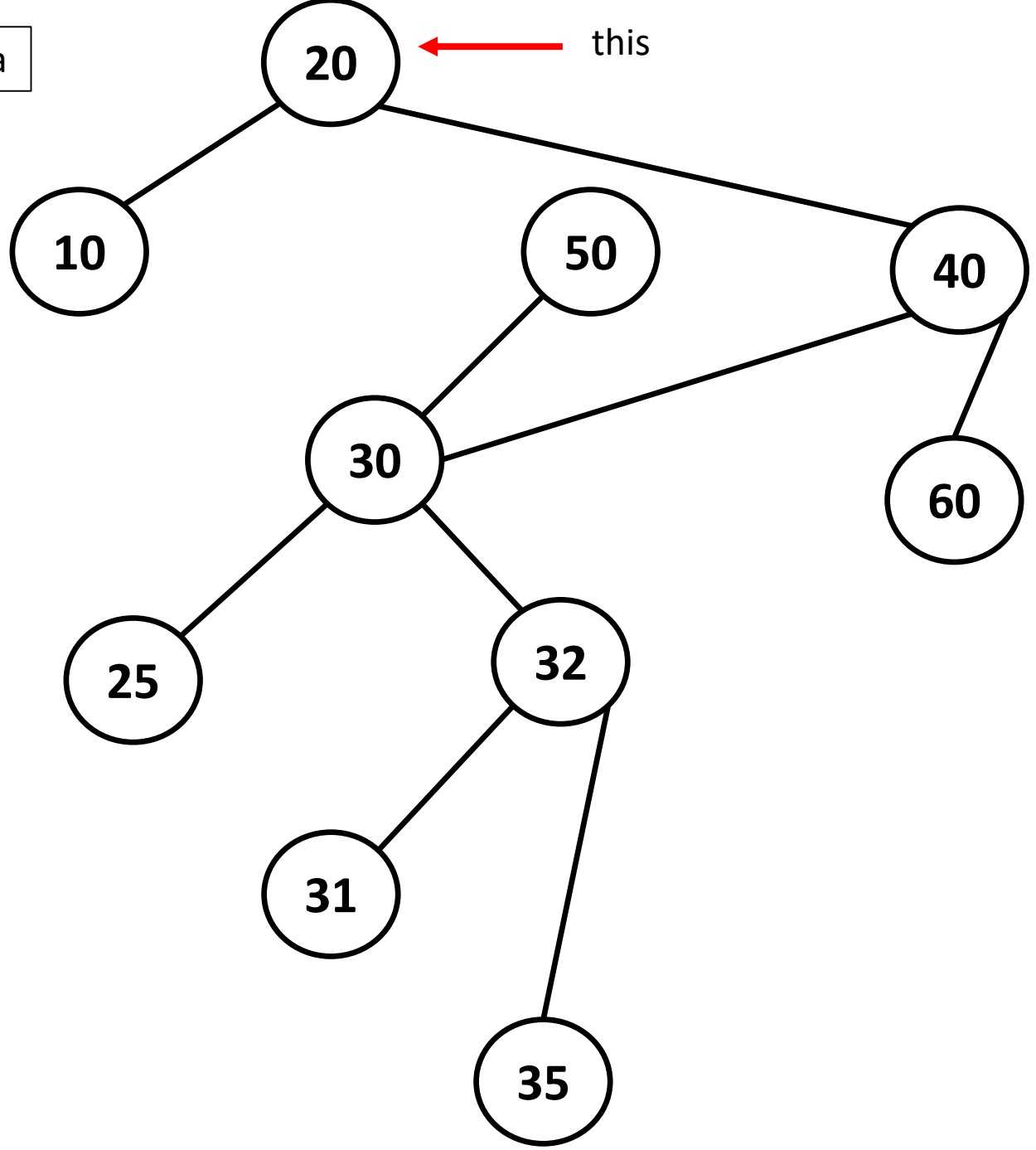
Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



elHijo.hijoDer ← this.hijoDer

Eliminación en árboles binarios de búsqueda

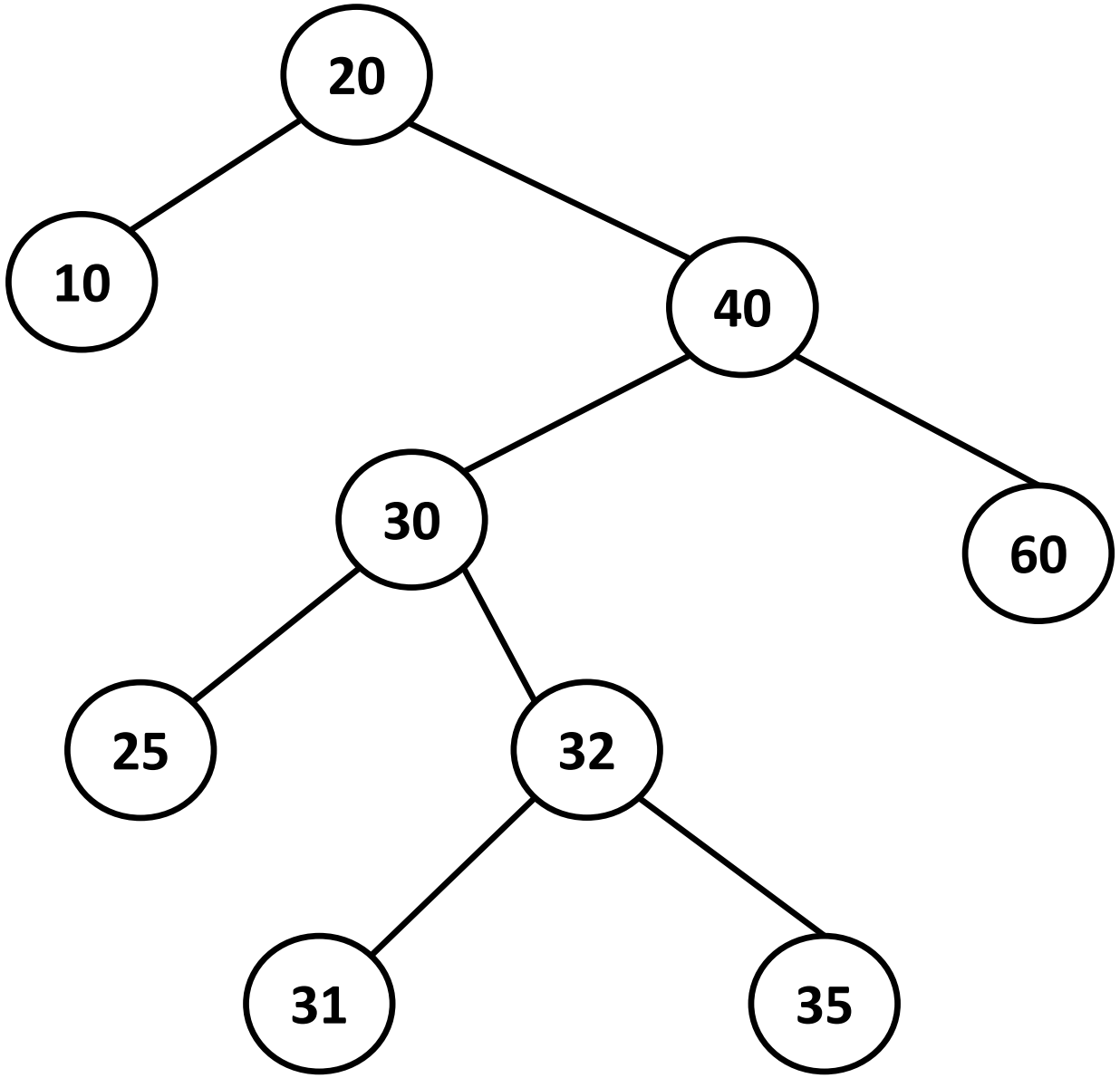
Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



retornar elHijo

Eliminación en árboles binarios de búsqueda

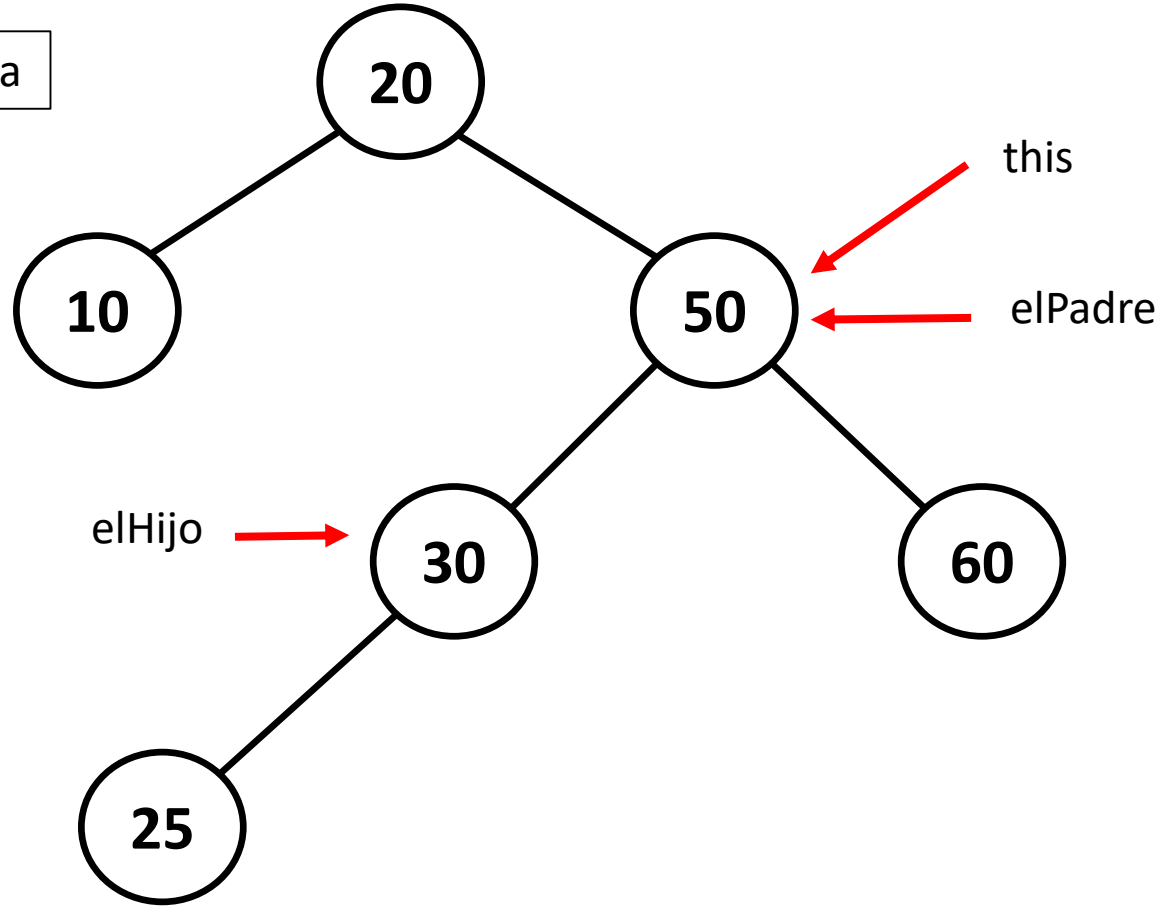
Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo



Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo

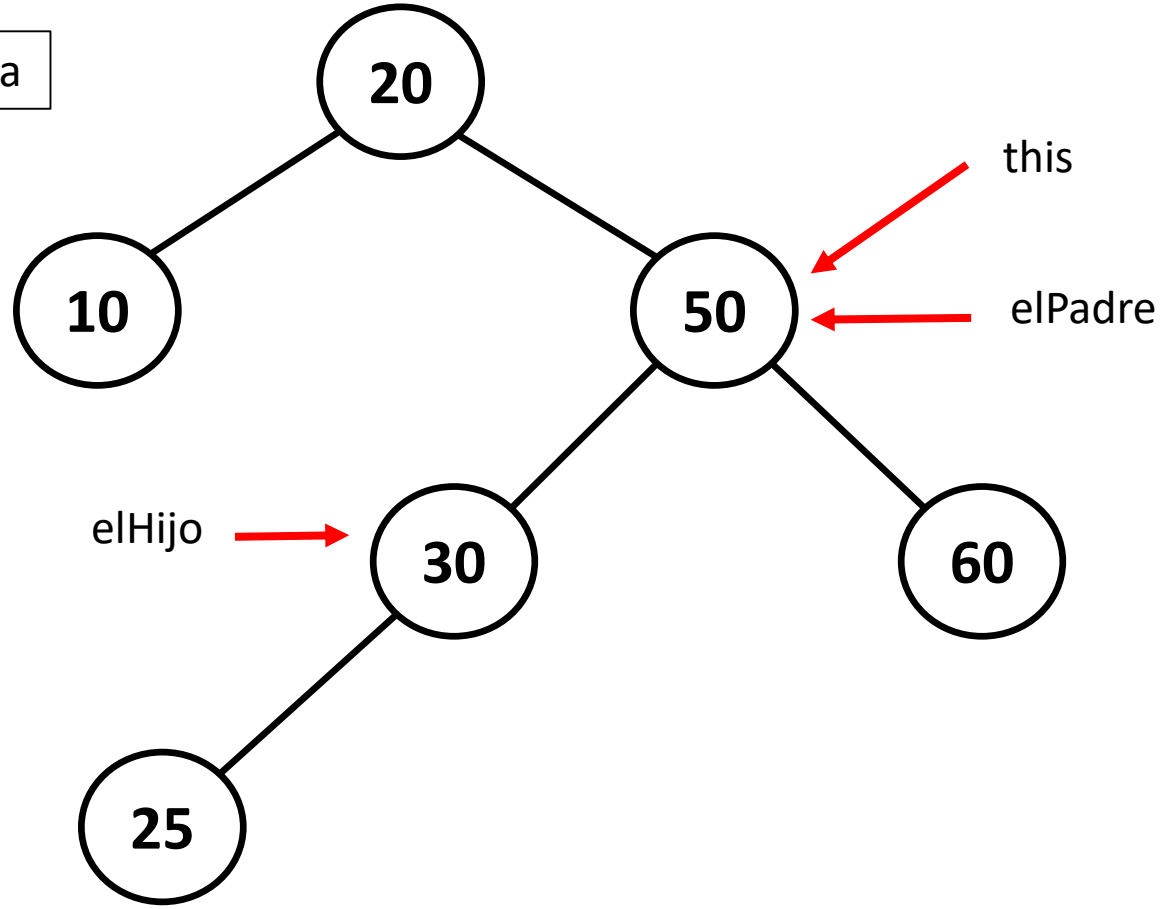
Caso especial, es su propio hijo



Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo

Caso especial, es su propio hijo

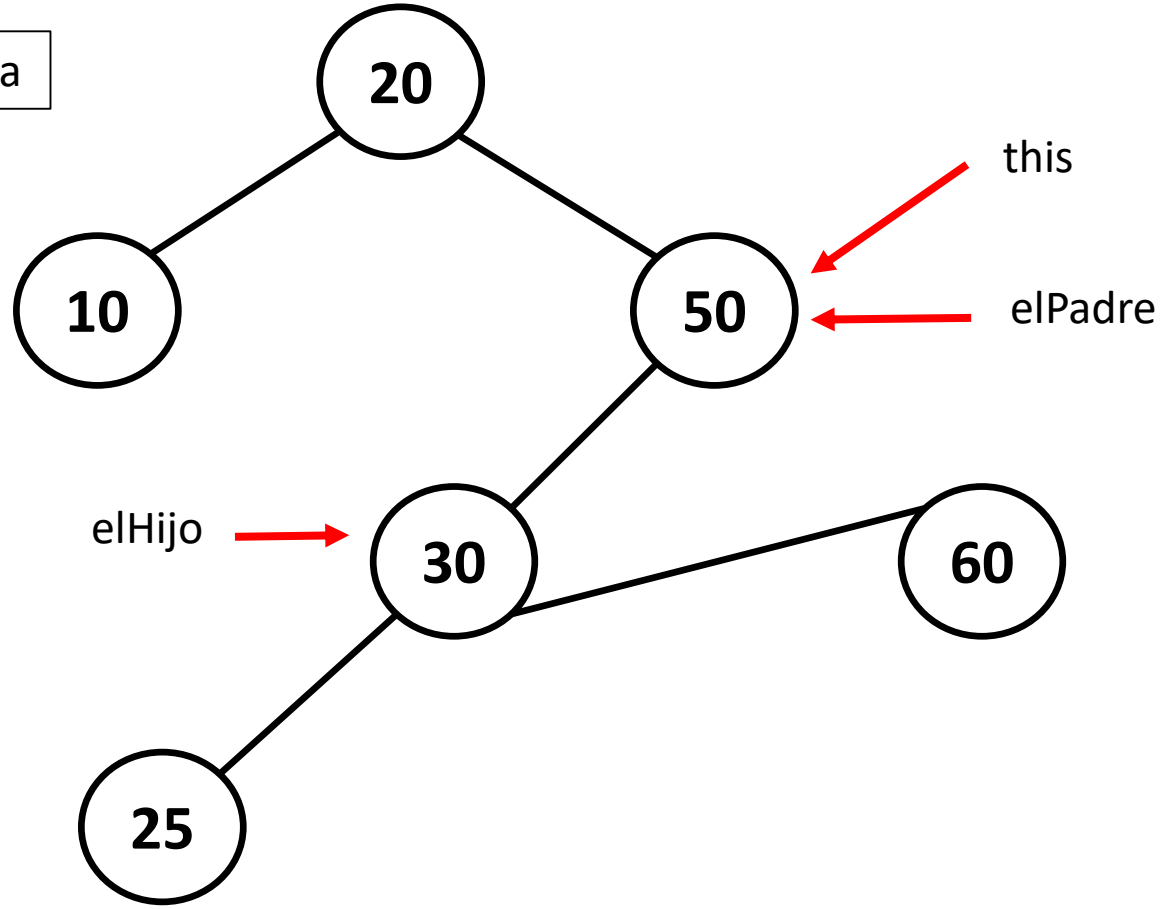


Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo

Caso especial, es su propio hijo

elHijo.hijoDer ← hijoDer

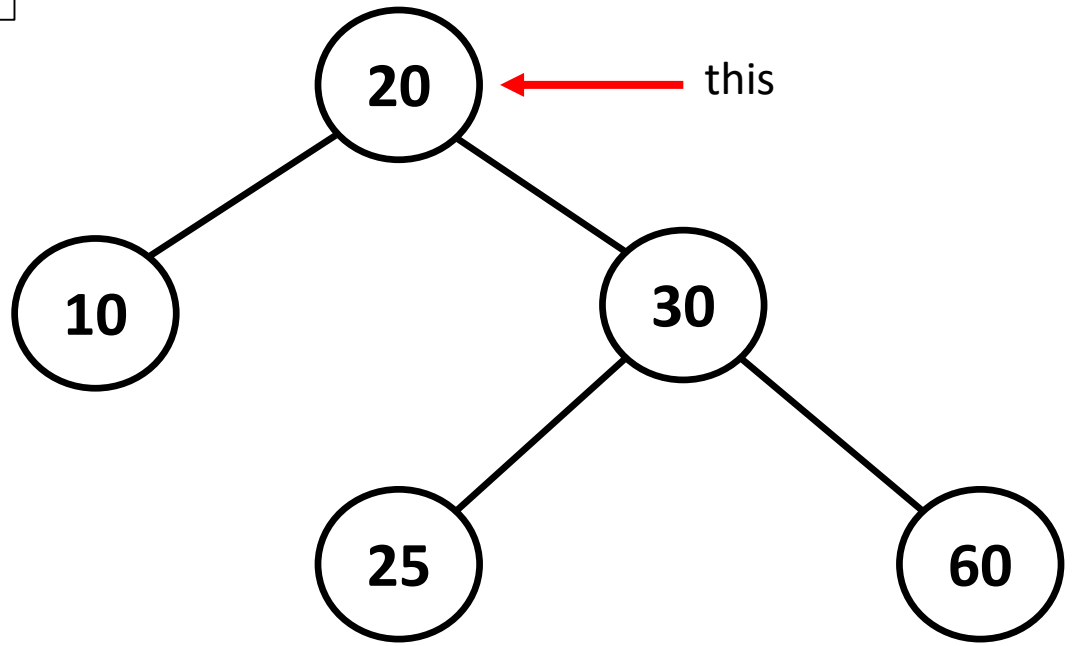


Eliminación en árboles binarios de búsqueda

Nodo completo, el nodo se sustituye por el mayor del subárbol izquierdo

Caso especial, es su propio hijo

return elHijo



```
de TNodoArbolBinarioBusqueda elMetodo (unaClave):  de algún tipo
COM
Si unaClave < etiqueta entonces   // hay que buscar en el subárbol izquierdo, si lo tiene
        Si hijoIzq != nulo entonces      // tiene subárbol izquierdo, llamar recursivamente
                        resultado = hijoIzq.elMetodo(unaClave)
                        retornar lo que corresponda

        sino    // no tiene subárbol izquierdo
                 // unaClave no está en el árbol, retornar lo que corresponda
        fin si
Fin si
Si unaClave > etiqueta entonces // unaClave es mayor que etiqueta, hay que buscar en el subárbol derecho, si lo tiene
        Si hijoDer != nulo entonces      // tiene subárbol izquierdo, llamar recursivamente
                        resultado = hijoDer.elMetodo(unaClave)
                        retornar lo que corresponda
        sino    // no tiene subárbol derecho
                // unaClave no está en el árbol, retornar lo que corresponda
        fin si
Fin si
 // unaClave está en el árbol, eventualmente hacer algo
retornar lo que corresponda
FIN
```

En ElementoArbolBinario.eliminar (UnaEtiqueta) : de Tipo ElementoAB

```
COM
  Si UnaEtiqueta < etiqueta entonces                    // si esta, está en el subárbol izquierdo
    Si hijoIzq <> nulo entonces
          hijoIzq ← hijoIzq.eliminar(UnaEtiqueta)       //actualiza el hijo, con el mismo u otro valor
    Finsi
    retornar (this)                                     //al padre le devuelve el mismo hijo
  Finsi
  Si UnaEtiqueta > etiqueta entonces                    // si esta, está en el subárbol derecho
     Si hijoDer <> nulo entonces
          hijoDer ← hijoDer.eliminar(UnaEtiqueta)       //actualiza el hijo, con el mismo u otro valor
     Finsi
     retornar (this)                                    // al padre le devuelve el mismo hijo
  Finsi
  // Cuando encuentra el nodo a eliminar llama, por claridad, al método que hace el trabajo
  retornar quitaElNodo                                  // al padre le devuelve el nuevo hijo
Fin
```

En ElementoArbolBinario.quitaElNodo: de Tipo ElementoAB;

Comienzo

```
(1)  Si hijoIzq = nulo entonces                  // le falta el hijo izquierdo o es hoja
         retornar hijoDer                        // puede retornar un nulo, si es hoja
(2) Si hijoDer = nulo entonces                   // le falta el hijo derecho
         retornar hijoIzq
(3)   // es un nodo completo
        elHijo ← hijoIzq                         // va al subárbol izquierdo
        elPadre ← this
        mientras elHijo.hijoDer <> nulo hacer
           elPadre ← elHijo
           elHijo ← elHijo.hijoDer
        fin mientras                             // elHijo es el mas a la derecha del subárbol izquierdo
       Si elPadre <> this entonces
         elPadre.hijoDer ← elHijo.hijoIzq
         elHijo.hijoIzq ← hijoIzq
       Finsi
       elHijo.hijoDer ← hijoDer
       retornar elHijo                           // elHijo quedara en lugar de this
Fin
```