

Traza — BSTree con Delete 8 (Caso E)

Construcción con 4 Inserts (10, 8, 11, 7) seguida de Delete 8. Ejercita las dos reglas recursivas de Insert con anidamiento y el **Caso E** del Delete ($j = i$, izquierdo no vacío, derecho vacío $\rightarrow$ retornar el izquierdo).

Término a evaluar:

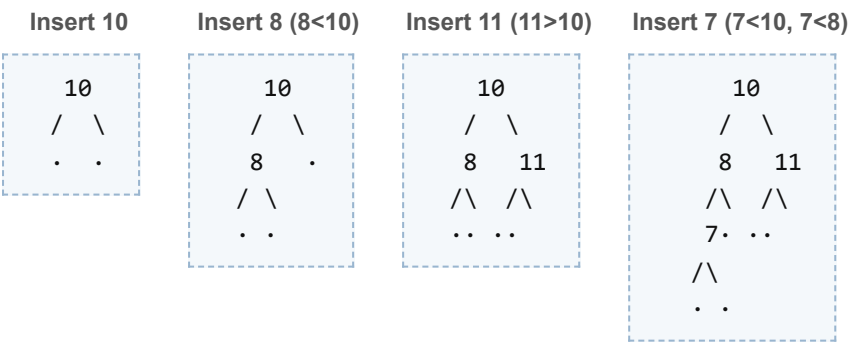
```
BSTree_Delete(  
  BSTree_Insert(  
    BSTree_Insert(  
      BSTree_Insert(  
        BSTree_Insert(BTree_Empty() as BSTree, 10),  
          8),  
        11),  
      7),  
    8)
```

Spec: BSTree-hide.nereus. Convención: $\cdot$ = BTree_Empty() as BSTree.

0. Convenciones

- Los pasos de construcción se numeran **A1–A8**; los del Delete **D1–D3**.
- El redex se resalta en **amarillo**.
- Las ligaduras (bindings) se listan antes del término reescrito.

Parte A — Construcción del BST



A1 — caso base de BSTree_Insert (i = 10)

Sobre BTree_Empty() as BSTree → singleton 10.

```
BSTree_Delete(  
  BSTree_Insert(  
    BSTree_Insert(  
      BSTree_Insert(  
        BTree_Create(BTree_Empty() as BSTree, BTree_Empty() as BSTree, 10) as BSTree,  
        8),  
      11),  
    7),  
  8)
```

A2 — Insert 8 con $i < j$ ($8 < 10$)

Axioma:

```
(i < j) implies  
BSTree_Insert(BTree_Create(b1, b2, j) as BSTree, i) =  
  BTree_Create(BSTree_Insert(b1 as BSTree, i), b2, j) as BSTree;
```

Bindings: b1 = b2 = ., j = 10, i = 8.

```
BSTree_Delete(  
  BSTree_Insert(  
    BSTree_Insert(  
      BTree_Create(  
        BSTree_Insert(BTree_Empty() as BSTree, 8),  
        BTree_Empty() as BSTree,  
        10) as BSTree,  
      11),  
    7),  
  8)
```

A3 — caso base interior (i = 8)

```
BSTree_Delete(  
  BSTree_Insert(  
    BSTree_Insert(  
      BTree_Create(  
        BTree_Create(., ., 8) as BSTree,  
        BTree_Empty() as BSTree,  
        10) as BSTree,  
      11),  
    7),  
  8)
```

A4 — Insert 11 con $i > j$ ($11 > 10$)

Axioma:

```
(i > j) implies
BSTree_Insert(BTree_Create(b1, b2, j) as BSTree, i) =
  BTree_Create(b1, BSTree_Insert(b2 as BSTree, i), j) as BSTree;
```

Bindings: $b1 = \text{BTree_Create}(\cdot, \cdot, 8)$ **as** BSTree, $b2 = \cdot$, $j = 10$, $i = 11$.

```
BSTree_Delete(
  BSTree_Insert(
    BTree_Create(
      BTree_Create( $\cdot$ ,  $\cdot$ , 8) as BSTree,
      BSTree_Insert(BTree_Empty() as BSTree, 11),
      10) as BSTree,
    7),
  8)
```

A5 — caso base interior ($i = 11$)

```
BSTree_Delete(
  BSTree_Insert(
    BTree_Create(
      BTree_Create( $\cdot$ ,  $\cdot$ , 8) as BSTree,
      BTree_Create( $\cdot$ ,  $\cdot$ , 11) as BSTree,
      10) as BSTree,
    7),
  8)
```

A6 — Insert 7 con $i < j$ ($7 < 10$), nivel raíz

Bindings: $b1 = \text{BTree_Create}(\cdot, \cdot, 8)$ **as** BSTree, $b2 = \text{BTree_Create}(\cdot, \cdot, 11)$ **as** BSTree, $j = 10$, $i = 7$.

```
BSTree_Delete(
  BTree_Create(
    BSTree_Insert(BTree_Create( $\cdot$ ,  $\cdot$ , 8) as BSTree, 7),
    BTree_Create( $\cdot$ ,  $\cdot$ , 11) as BSTree,
    10) as BSTree,
  8)
```

A7 — Insert 7 con $i < j$ ($7 < 8$), recursión en el subárbol izquierdo

Sobre `BSTree_Insert(BTree_Create(·, ·, 8) as BSTree, 7)`:

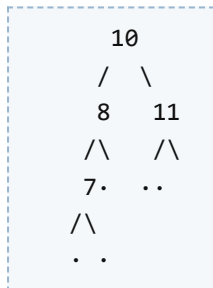
Bindings: `b1' = b2' = ·, j' = 8, i = 7`. Condición $7 < 8$ ✓

```
BSTree_Delete(  
  BTree_Create(  
    BTree_Create(  
      BSTree_Insert(BTree_Empty() as BSTree, 7),  
      BTree_Empty() as BSTree,  
      8) as BSTree,  
    BTree_Create(·, ·, 11) as BSTree,  
    10) as BSTree,  
  8)
```

A8 — caso base innermost ($i = 7$)

```
BSTree_Delete(  
  BTree_Create(  
    BTree_Create(  
      BTree_Create(·, ·, 7) as BSTree,  
      BTree_Empty() as BSTree,  
      8) as BSTree,  
    BTree_Create(·, ·, 11) as BSTree,  
    10) as BSTree,  
  8)
```

BST construido — 4 nodos



Hasta acá terminamos la fase de construcción. Llamemos `T` a este BST. Ahora arranca el `BSTree_Delete(T, 8)`.

Parte D — Borrar 8 (Caso A → Caso E)

D1 — BSTree_Delete Caso A ($j < i$: $8 < 10$), recurrir en el subárbol izquierdo

Axioma:

```
(j < i) implies
BSTree_Delete(BTree_Create(b1, b2, i) as BSTree, j) =
  BTree_Create(BSTree_Delete(b1 as BSTree, j), b2, i) as BSTree;
```

Bindings:

```
b1 = BTree_Create(BTree_Create(.,.,7) as BSTree, ., 8) as BSTree
b2 = BTree_Create(., ., 11) as BSTree
i = 10, j = 8 → j < i ✓
```

```
BTree_Create(
  BSTree_Delete(BTree_Create(BTree_Create(.,.,7) as BSTree, ., 8) as BSTree, 8),
  BTree_Create(., ., 11) as BSTree,
  10) as BSTree
```

D2 — BSTree_Delete Caso E ($j = i$, izq. no vacío, der. vacío) ← punto interesante

Sobre $\text{BSTree_Delete}(\text{BTree_Create}(\text{BTree_Create}(\cdot, \cdot, 7) \text{ as BSTree}, \cdot, 8) \text{ as BSTree}, 8)$:

Variables del patrón (renombramiento local para no confundir con el outer):

```
b1' = BTree_Create(., ., 7) as BSTree → not BTree_IsEmpty(b1') ✓
b2' = BTree_Empty() as BSTree → BTree_IsEmpty(b2') ✓
i' = 8, j = 8 → j = i' ✓
```

Verificación de los 4 casos con $j = i$:

Caso	Condición de hijos	Aplica
C	$\text{IsEmpty}(b1') \wedge \text{IsEmpty}(b2') = F \wedge T = F$	No
D	$\text{IsEmpty}(b1') \wedge \neg \text{IsEmpty}(b2') = F \wedge F = F$	No
E	$\neg \text{IsEmpty}(b1') \wedge \text{IsEmpty}(b2') = T \wedge T = T$	Sí ←
F	$\neg \text{IsEmpty}(b1') \wedge \neg \text{IsEmpty}(b2') = T \wedge F = F$	No

Axioma Caso E:

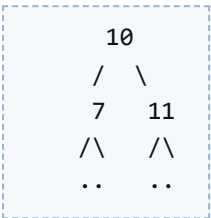
```
(j = i and not BTree_IsEmpty(b1) and BTree_IsEmpty(b2)) implies
BSTree_Delete(BTree_Create(b1, b2, i) as BSTree, j) =
  b1 as BSTree;
```

Resultado: $b1' = \text{BTree_Create}(\cdot, \cdot, 7) \text{ as BSTree}$. El subárbol izquierdo simplemente sube a ocupar el lugar del nodo borrado.

D3 — Sustituir el resultado del Delete interno y obtener la forma normal

```
BTree_Create(  
  BTree_Create(·, ·, 7) as BSTree,  
  BTree_Create(·, ·, 11) as BSTree,  
  10) as BSTree
```

Vista de árbol:



Lectura semántica: el 8 era un nodo con un solo hijo (7 a la izquierda, sin descendiente derecho). El Caso E lo elimina *sin necesidad de buscar reemplazo*: el subárbol izquierdo (cuyo único nodo era 7) sube y ocupa el lugar del 8. La invariante BST se preserva: $7 < 10 \checkmark$, $11 > 10 \checkmark$.

Verificación y resumen

Aspecto	Antes del Delete	Después del Delete
Elementos	{7, 8, 10, 11}	{7, 10, 11}
Estructura	raíz 10 con izq=BTree_Create(BTree_Create(·,·,7), ·, 8), der=BTree_Create(·,·,11)	raíz 10 con izq=BTree_Create(·,·,7), der=BTree_Create(·,·,11)
Altura	3	2
Invariante BST	$7 < 8 < 10 < 11 \checkmark$	$7 < 10 < 11 \checkmark$

Por qué este caso es distinto a las trazas anteriores:

- [BSTree-hide-trace.pdf](#): *Delete 10* sobre un BST de 3 nodos con dos hijos no vacíos → **Caso F** (necesitó BSTree_Min + sub-Delete).
- [BSTree-hide-traces.pdf](#): *Delete 8* sobre el mismo BST de 3 nodos donde 8 era una hoja → **Caso A** → **Caso C** (recurrir + borrar nodo hoja).
- **Este documento**: *Delete 8* sobre un BST de 4 nodos donde 8 tiene **un solo hijo** (7 a la izquierda) → **Caso A** → **Caso E** (recurrir + reemplazar por el único hijo).

Falta ejercitar solamente el **Caso D** ($j = i$, izq. vacío, der. no vacío); se obtiene con un BST asimétrico al espejo, por ejemplo *Delete 12* sobre *Insert(Insert(Insert(empty, 10), 12), 15)*.

Detalle algorítmico: el Caso E (y su simétrico D) es la razón por la que el Delete de un BST se complica solo cuando el nodo a borrar tiene **dos hijos** (Caso F): en ese caso no hay un único hijo natural para

promover y hay que buscar el sucesor in-order (el mínimo del subárbol derecho). Aquí, con un solo hijo, basta promoverlo directamente.

Traza generada para la spec `BSTree-hide.nereus`. Pasos: 8 para la construcción (incluye un Insert con doble nivel de recursión, A6–A8), 3 para el Delete. Total 11 reducciones.