Problemas resueltos

Problemas resueltos

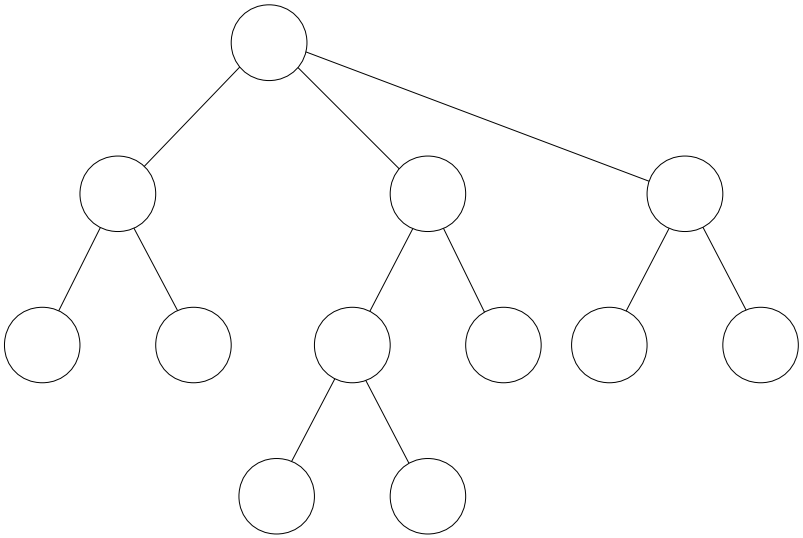
Da el código para el operator++ de un iterador que recorre un árbol en inorden.

Cada nodo del árbol puede tener varios hijos. Si tiene uno, será el hijo a la izquierda, si tiene varios más serán, hijos a la derecha.

Recordamos inorden: Para recorrer un nodo en inorden se recorren en inorden los hijos de la izquierda, luego la raíz y luego en inorden los hijos de la derecha.

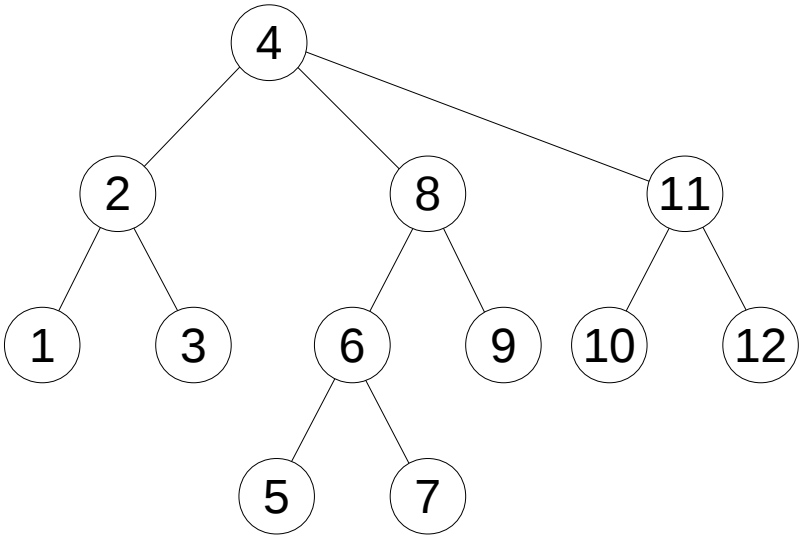
Problemas resueltos

En primer lugar vamos a extraer las diferentes reglas que necesitamos considerar. Para ello dibujamos un árbol (grande para considerar varios casos).



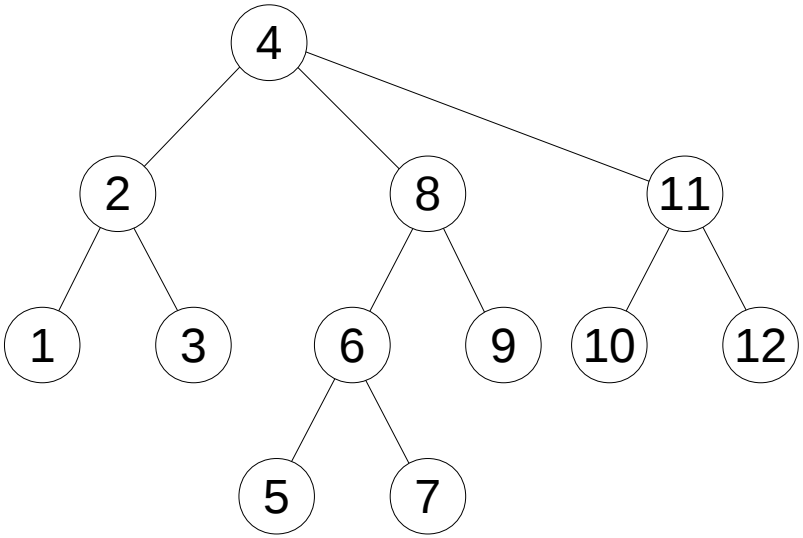
Problemas resueltos

Numeramos los nodos según recorrido inorden.



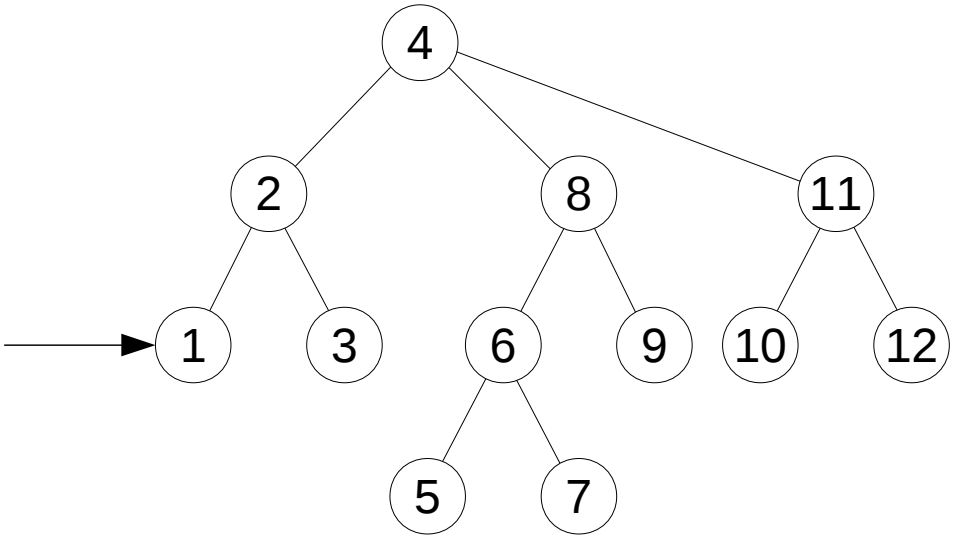
Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.



Problemas resueltos

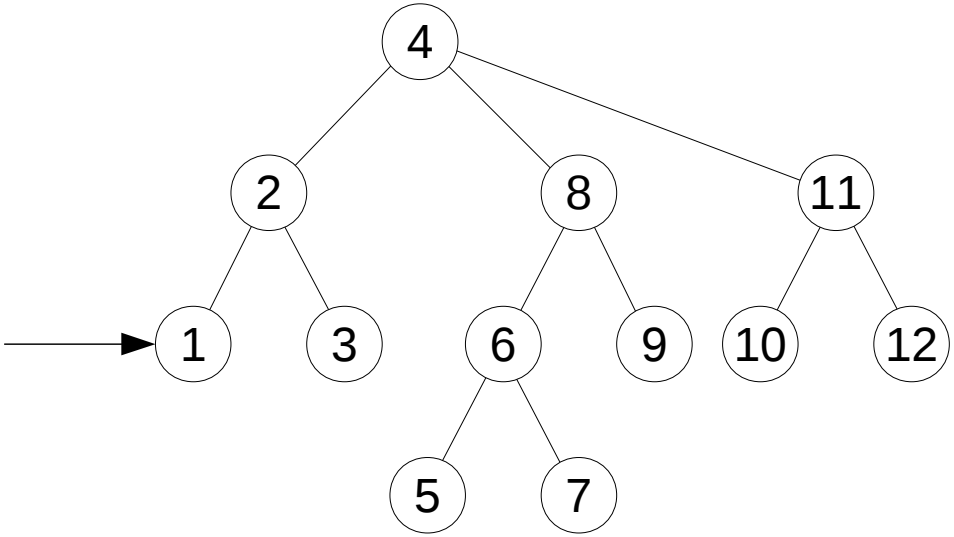
Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

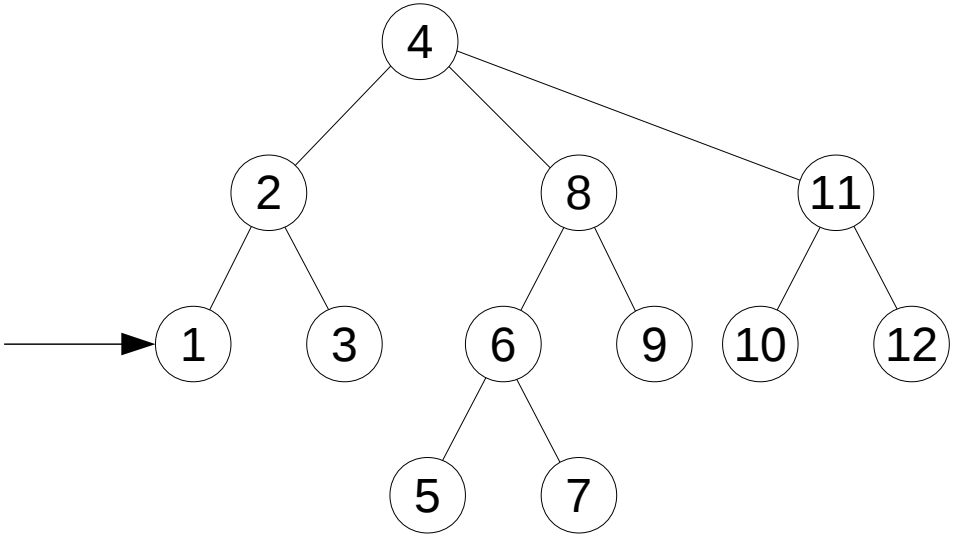
`Iniciamos en el hijo más a la izquierda.`



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

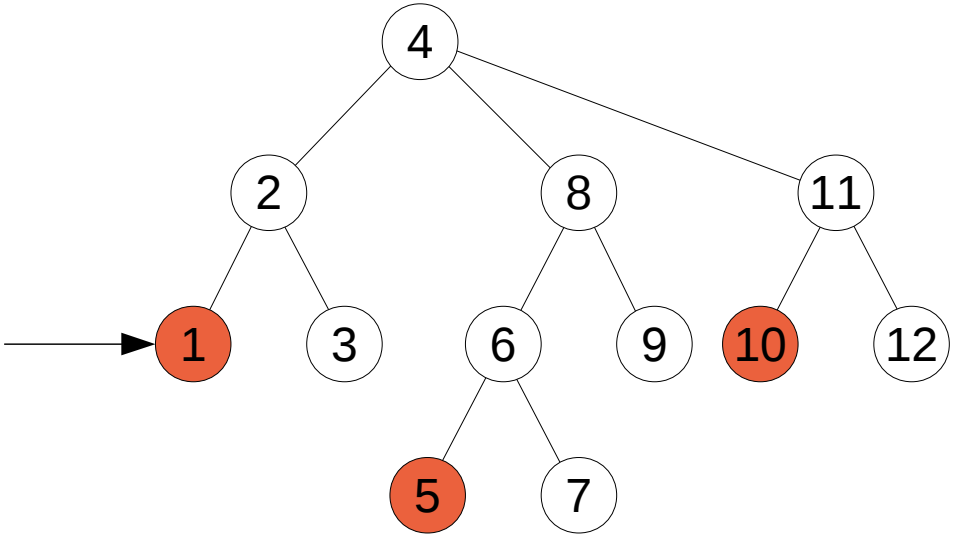
`Iniciamos en el hijo más a la izquierda.`



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

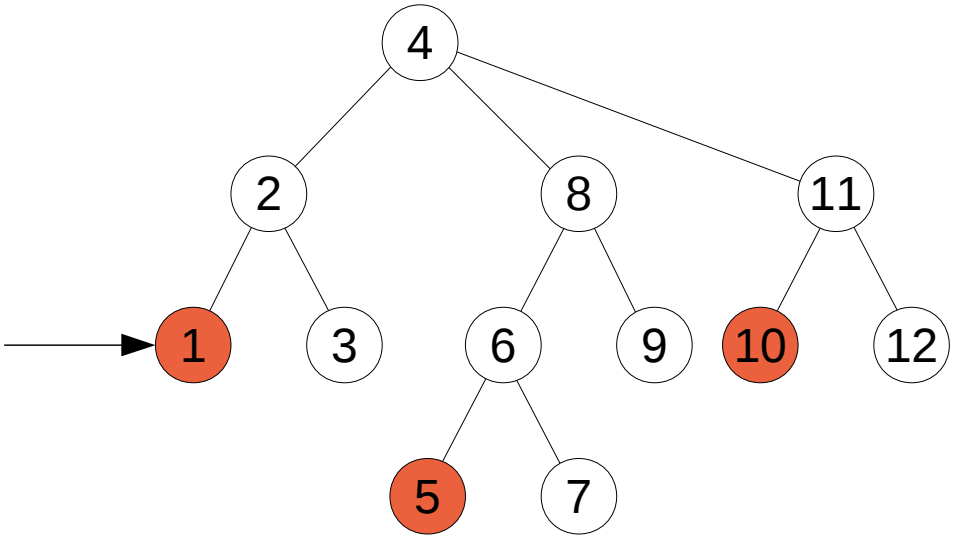
`Iniciamos en el hijo más a la izquierda.`



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

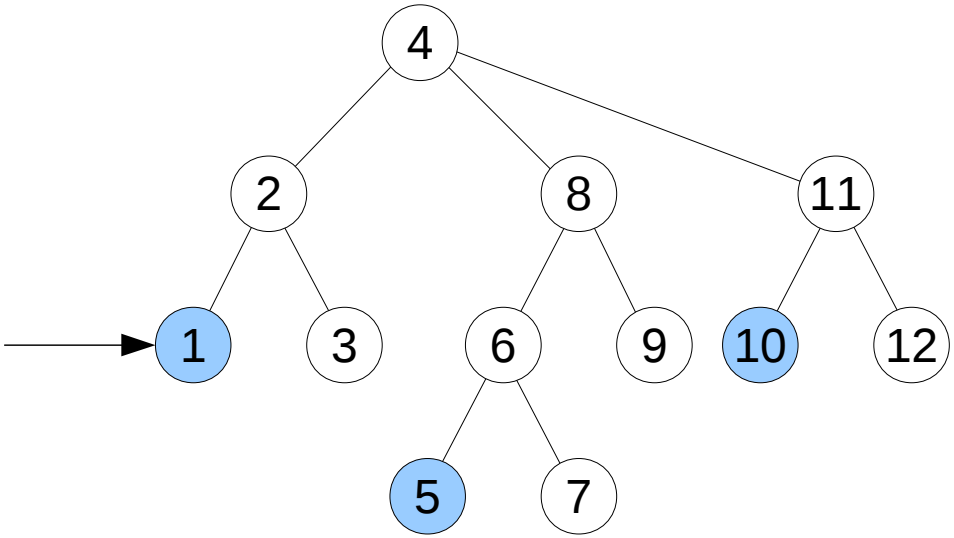
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
}
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

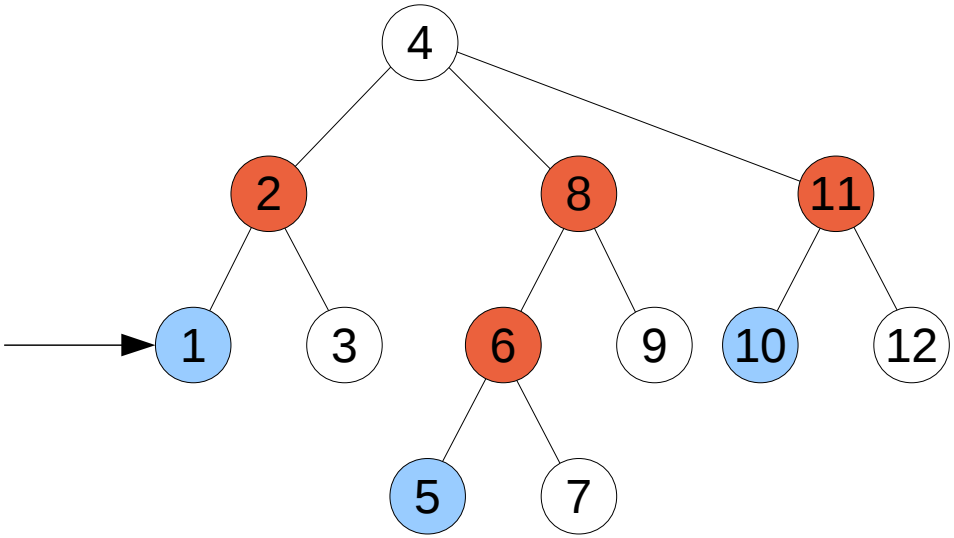
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
}
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

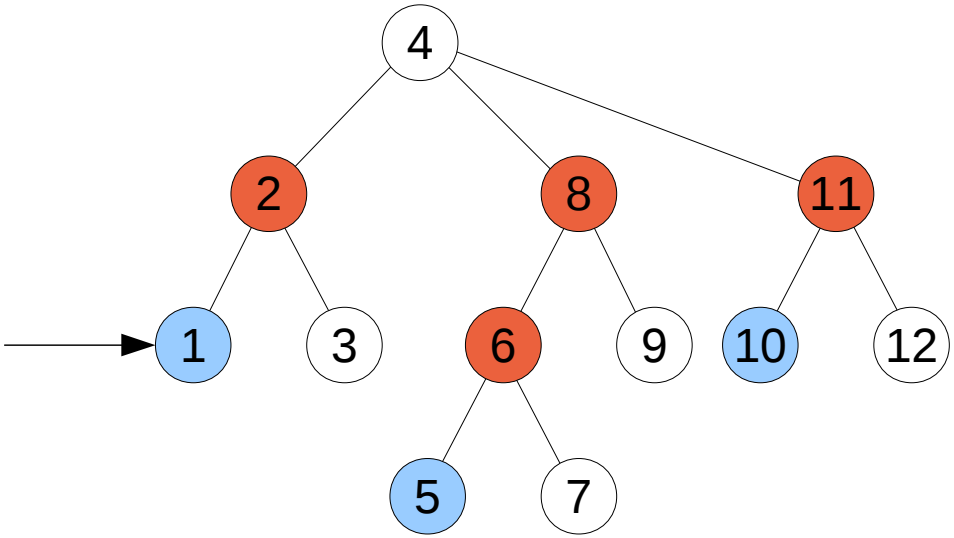
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
}
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

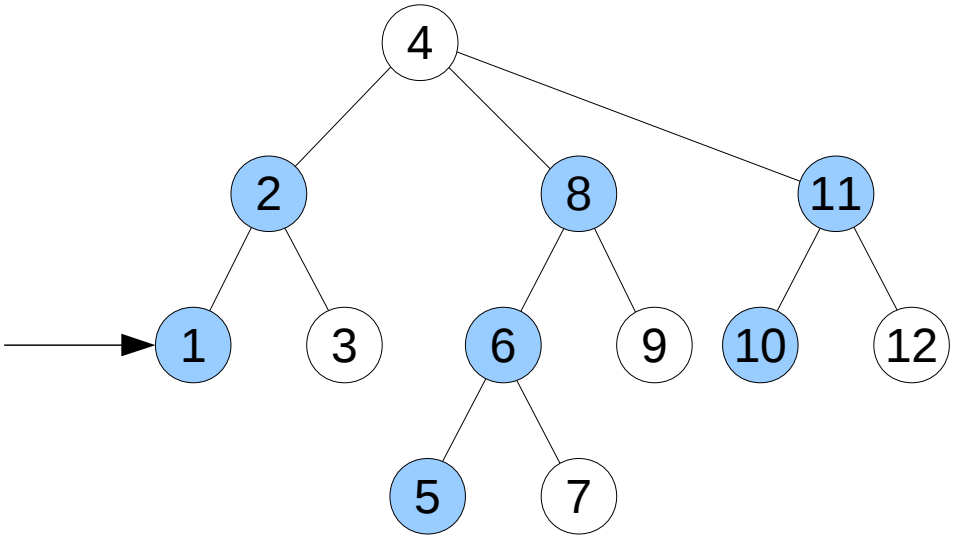
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

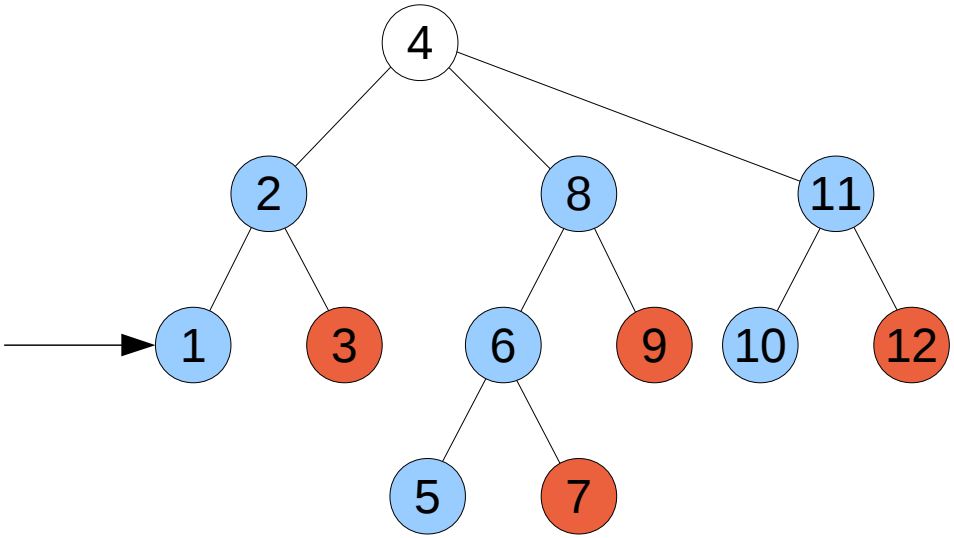
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

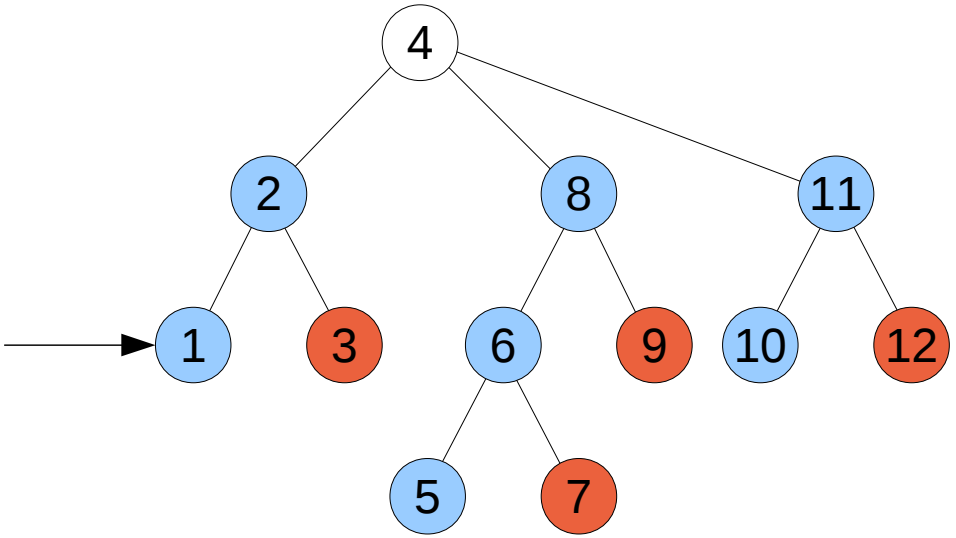
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

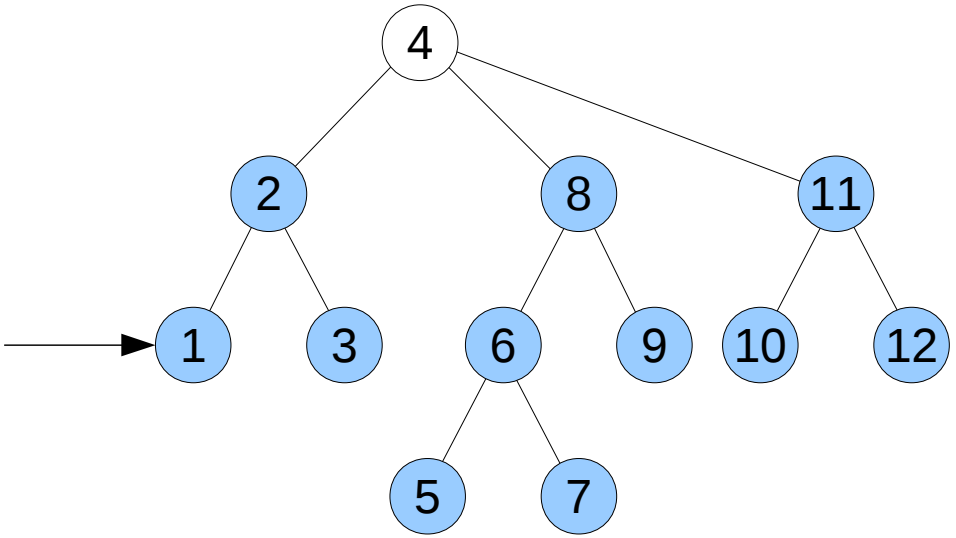
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha:
      Aux = padre
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

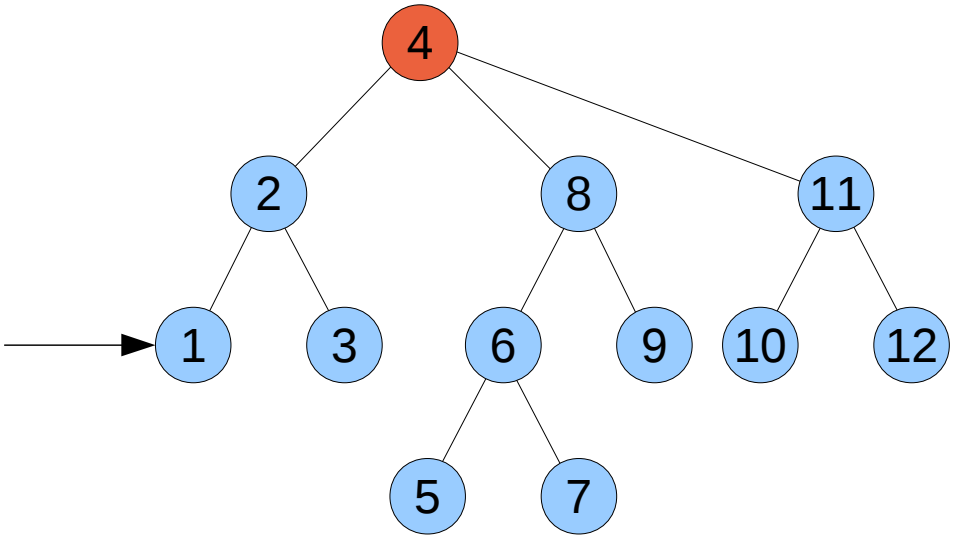
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha:
      Aux = padre
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

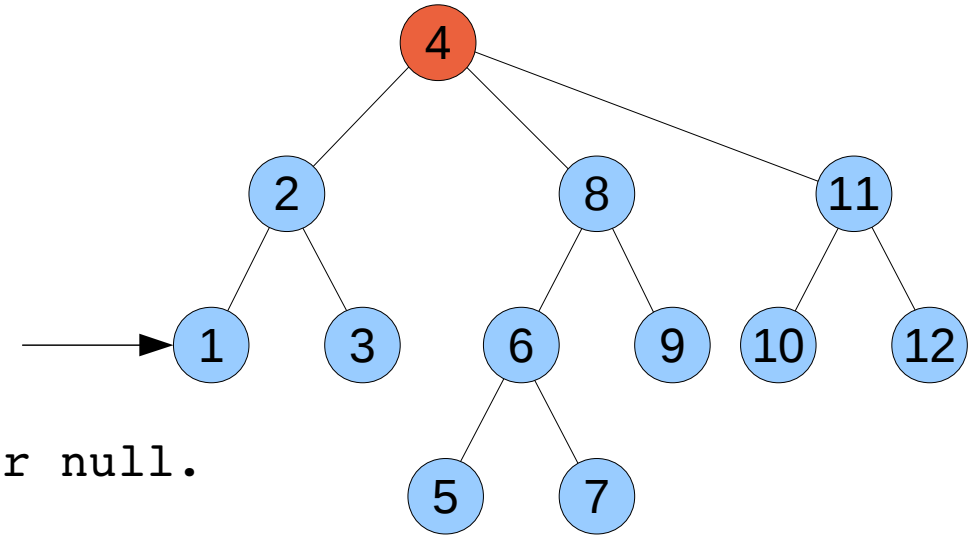
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha:
      Aux = padre
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

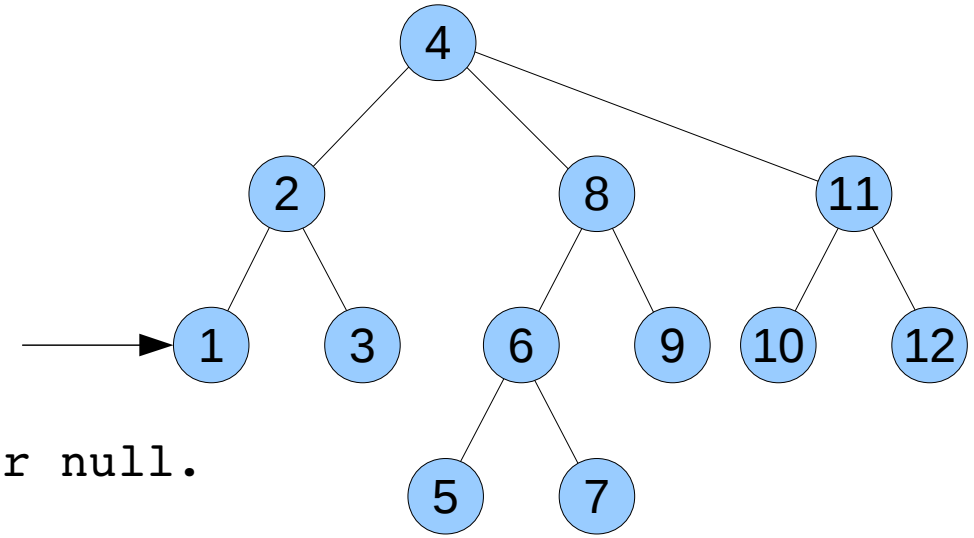
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Consideramos los distintos casos... Mientras lo consideramos queda marcado en rojo, si ya está considerado, en azul.

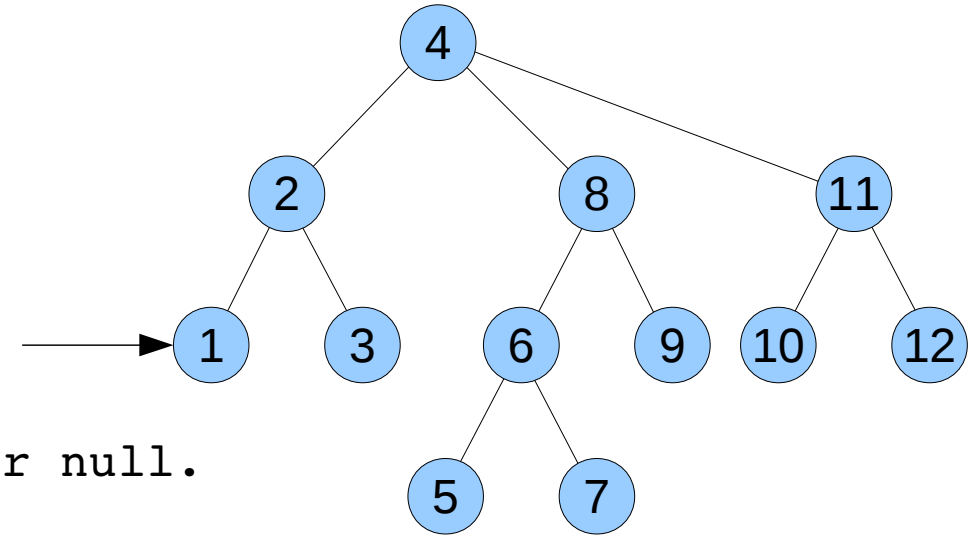
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

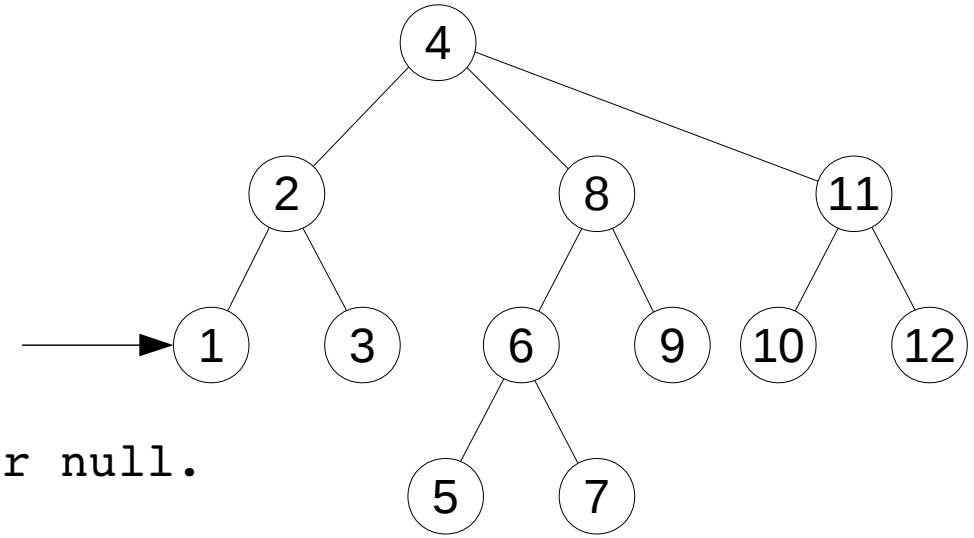
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

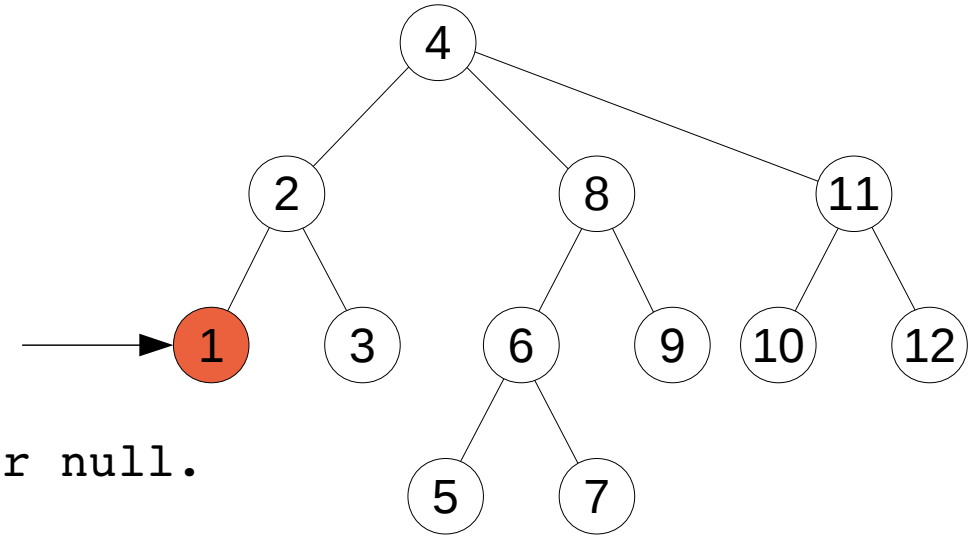
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

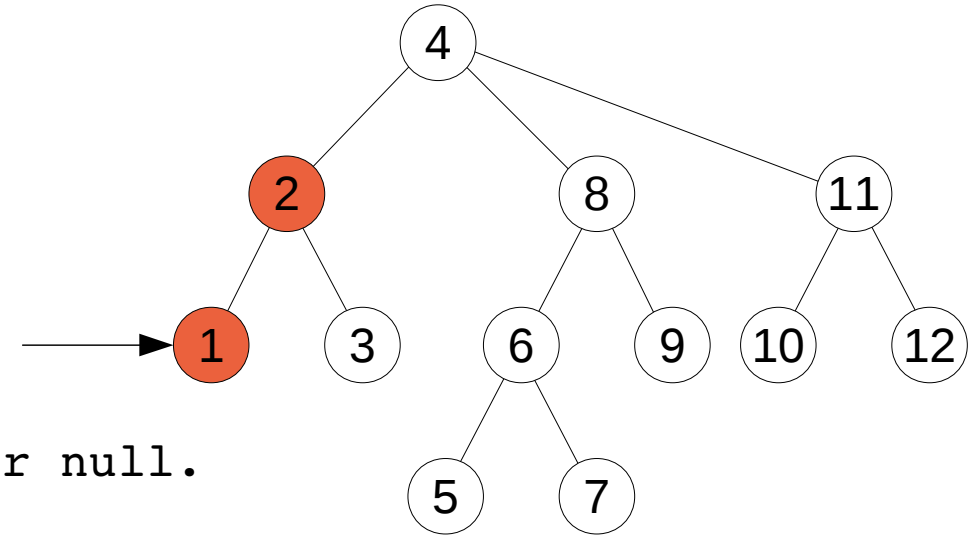
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

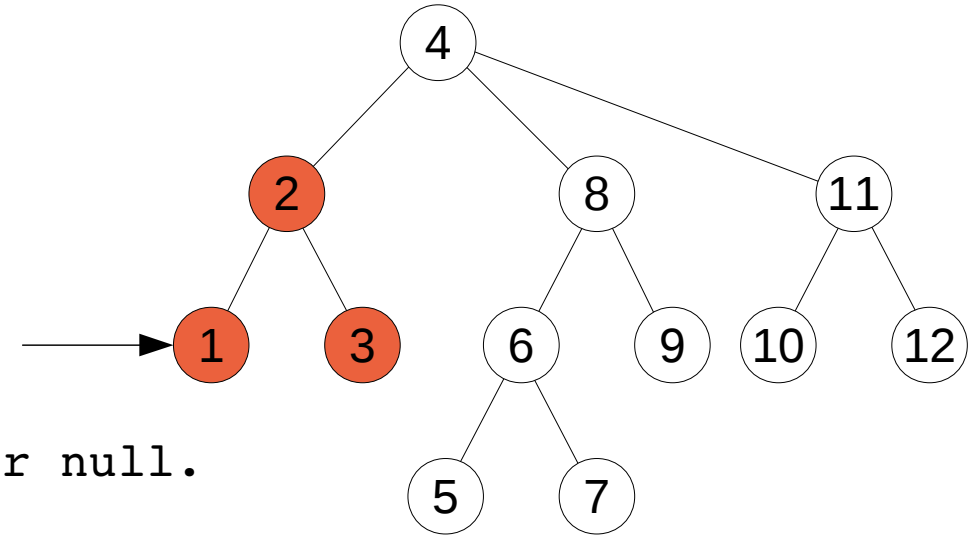
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

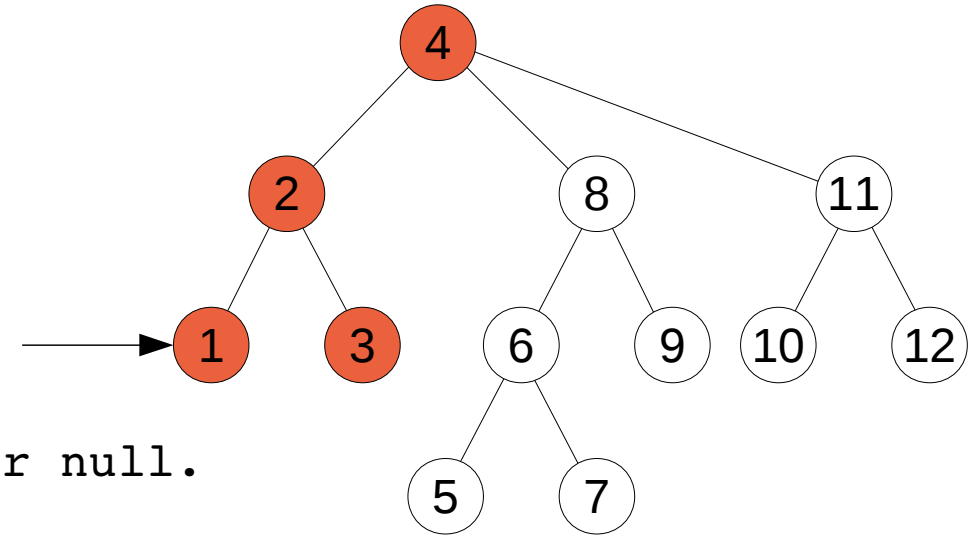
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

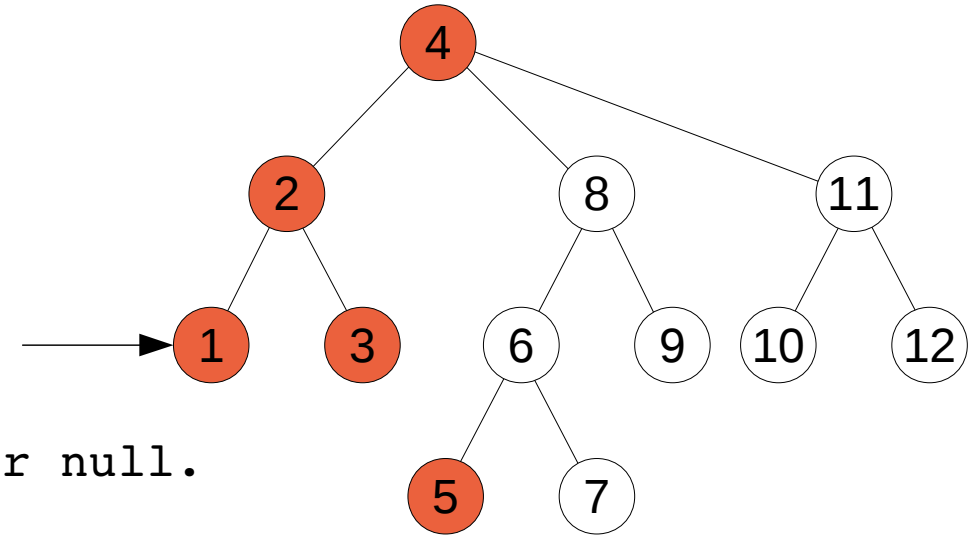
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

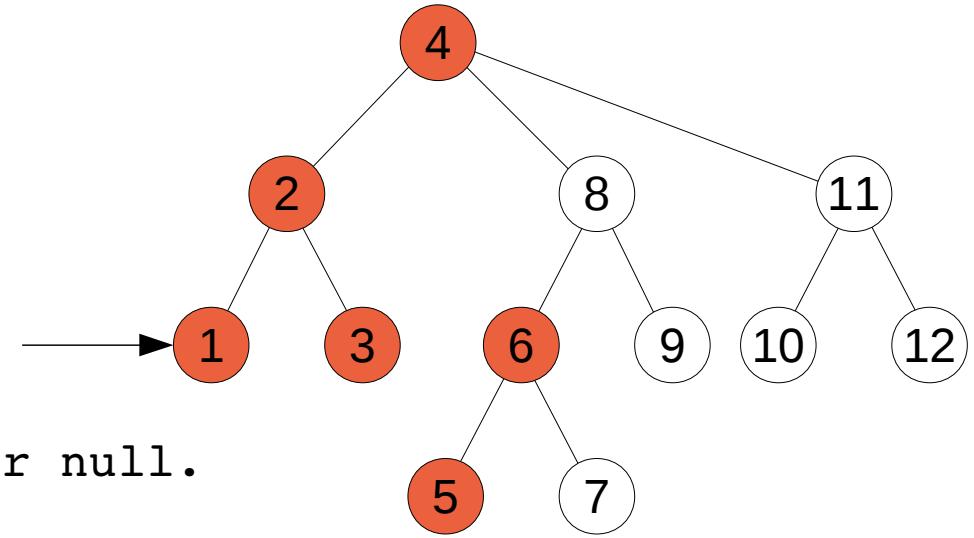
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

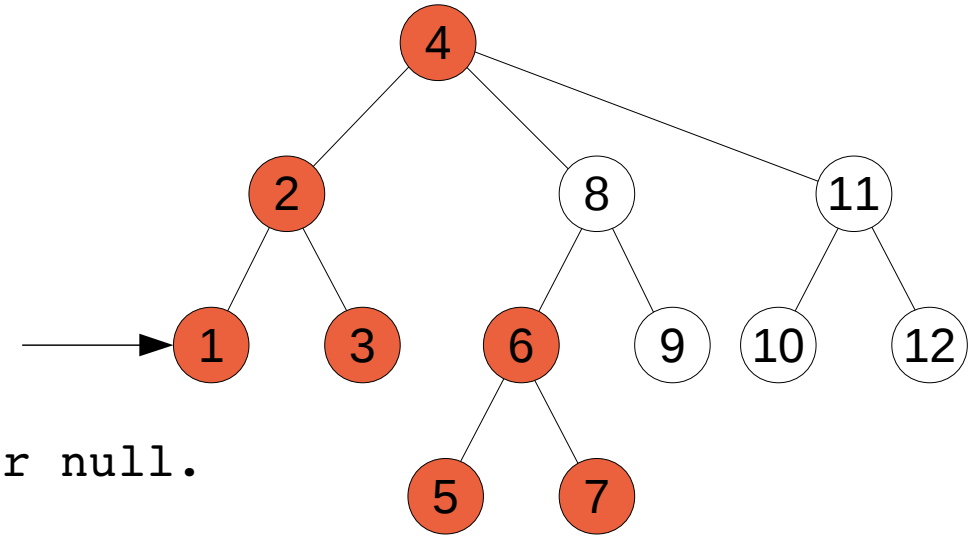
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

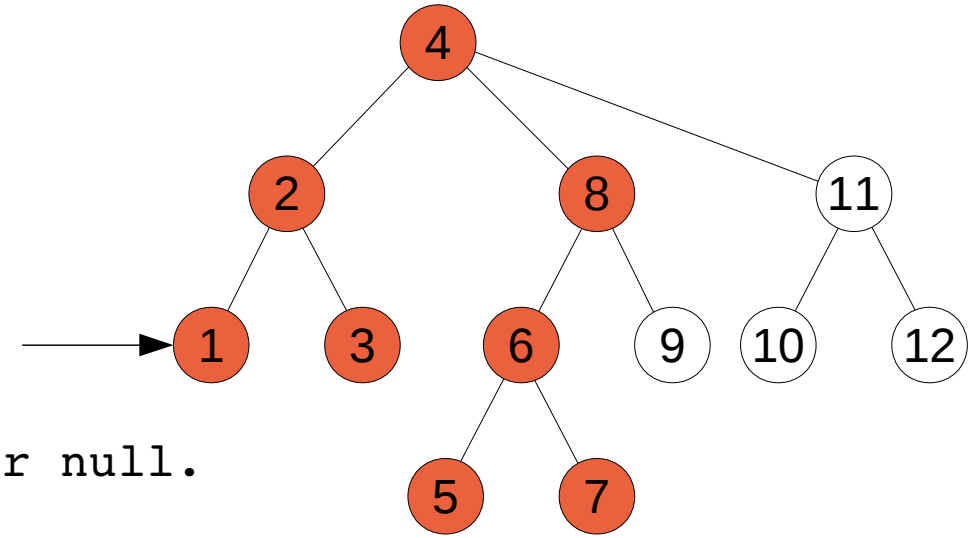
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

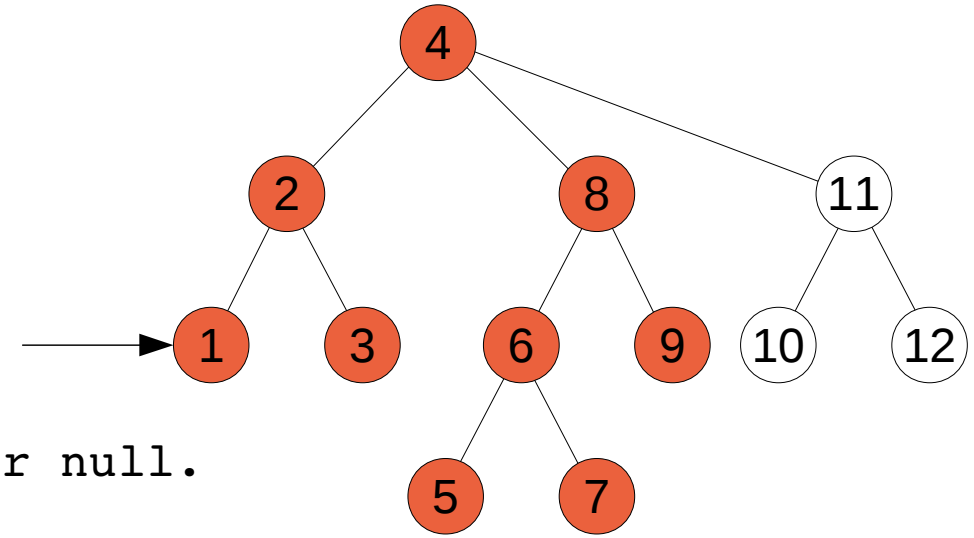
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

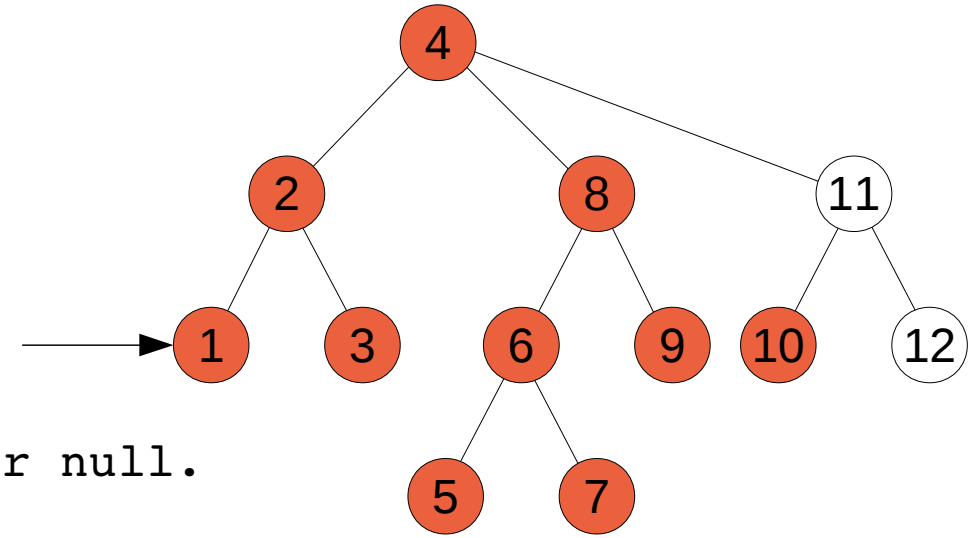
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

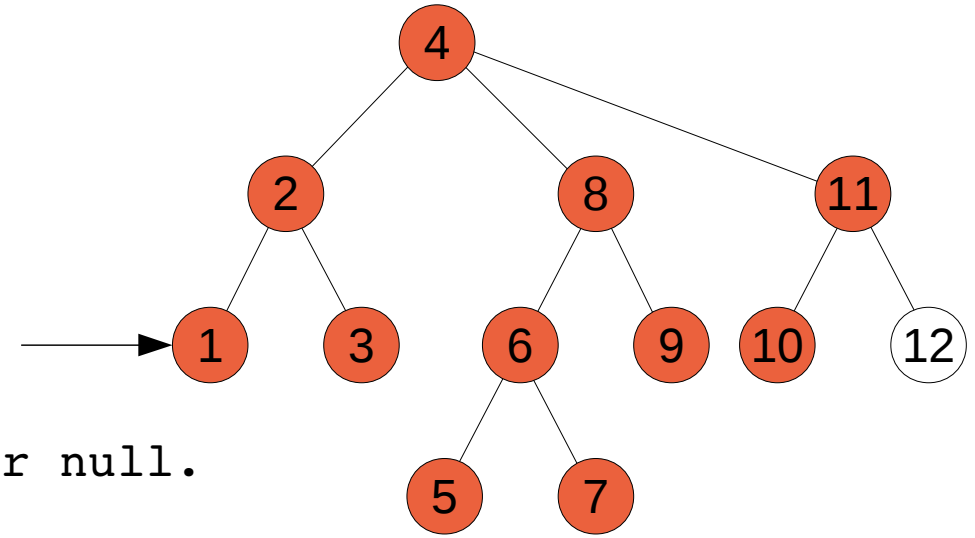
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

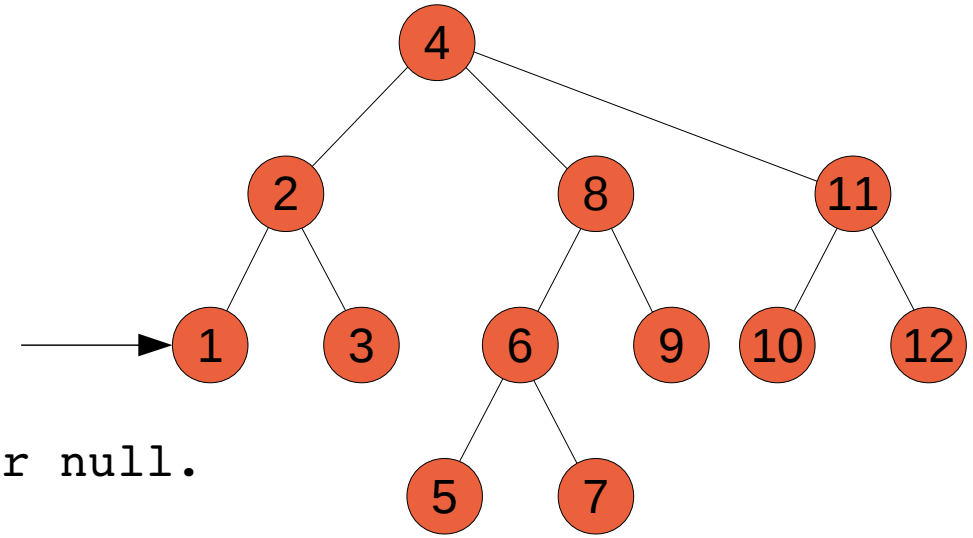
```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```



Problemas resueltos

Comprobamos!

```
Iniciamos en el hijo más a la izquierda.
Si no tengo hijos: {
  Si soy el más a la izquierda:
    Devuelvo el padre.
  Si soy el más a la derecha: {
    Aux = actual
    Hasta que Aux no sea
      hijo más a la derecha: {
      Aux = padre
      Si aux.padre == null, devolver null.
    }
    Si aux es hijo izquierda:
      Devolver el padre.
    Si aux es hijo derecha (no último):
      Devolver hijo más a la izq de hermano derecha
  }
}
Si tengo hijos:
  Devuelvo el hijo más a la
    izquierda de mi primer hijo
    derecho
```