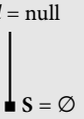
Precondition: *head* is a tree representing some set **S**. We are inserting *value*.

Tree(*head*, **S**)

Is *head* null?

head = null

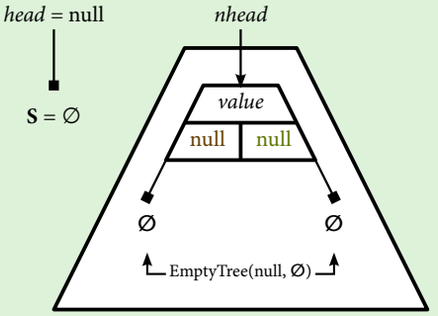
The tree at *head* is empty, and represents the set ∅. We can return a one-node tree.



EmptyTree(*head*, **S**)

Construct one-node tree containing *value*, pointed at by new variable *nhead*.

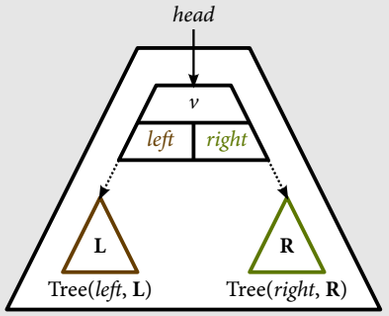
We require ∅ ∪ {*value*} = {*value*}; new tree is ∅ ∪ {*value*} ∪ ∅ = {*value*}. Return *nhead*.



NonEmptyTree(*head*, ∅ ∪ {*value*} ∪ ∅)

head ≠ null

head points to a non-empty tree with some *v* at the root and two, possibly empty, subtrees.

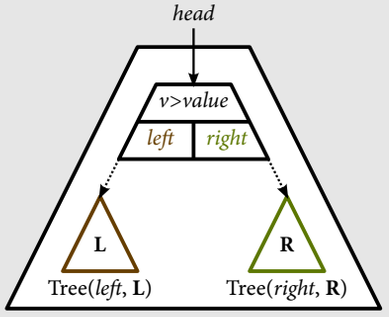


NonEmptyTree(*head*, **S** = **L** ∪ {*v*} ∪ **R**)

compare(*value*, *v*)

value < *v*

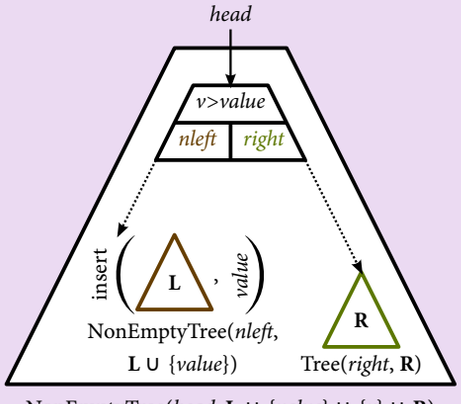
We cannot insert into the *right* tree, because that would break ∀*r*∈**R**. *v*<*r*.



NonEmptyTree(*head*, **S** = **L** ∪ {*v*} ∪ **R**)

Recursively call insert(*left*, *value*), yielding *nleft*; set *left* field to *nleft*.

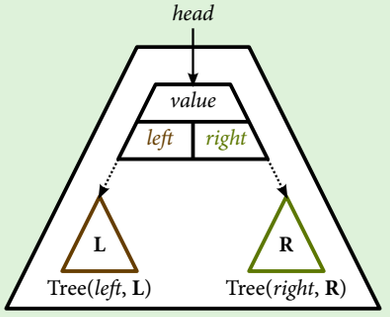
Tree represents **L** ∪ {*value*} ∪ {*v*} ∪ **R**, which = **S** ∪ {*value*}. Return *head*.



NonEmptyTree(*head*, **L** ∪ {*value*} ∪ {*v*} ∪ **R**)

value = *v*

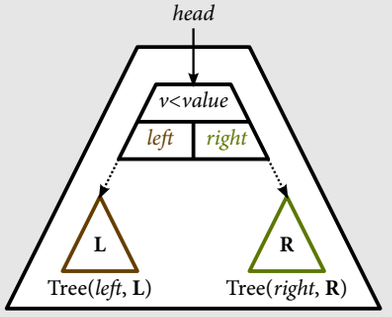
value ∈ {*v*}, so *value* ∈ **L** ∪ {v} ∪ **R**, so *value* ∈ **S**, so **S** ∪ {*value*} = **S**. Return *head*.



NonEmptyTree(*head*, **S** = **L** ∪ {*value*} ∪ **R**)

value > *v*

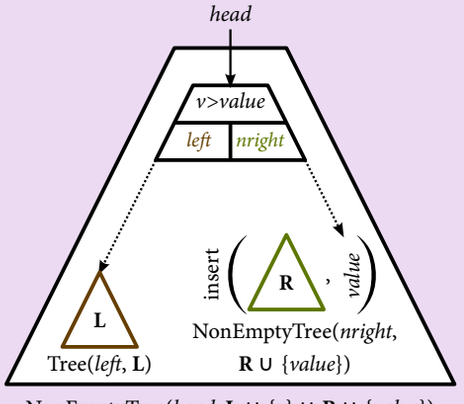
We cannot insert into the *left* tree, because that would break ∀*l*∈**L**. *v*>*l*.



NonEmptyTree(*head*, **S** = **L** ∪ {*v*} ∪ **R**)

Recursively call insert(*right*, *value*), yielding *nright*; set *right* field to *nright*.

Tree represents **L** ∪ {*v*} ∪ **R** ∪ {*value*}, which = **S** ∪ {*value*}. Return *head*.



NonEmptyTree(*head*, **L** ∪ {*v*} ∪ **R** ∪ {*value*})