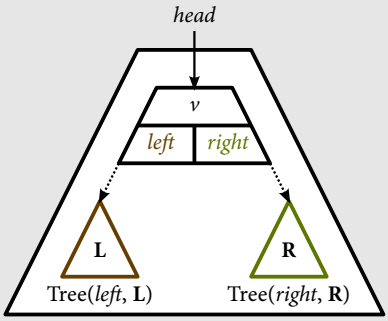
Precondition: *head* is a tree representing some set **S**. We are removing *value*.

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

Is *head* null?

head ≠ null

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



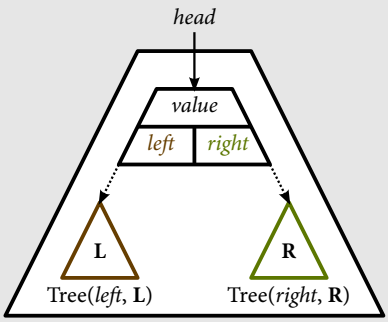
Tree(*left*, **L**) Tree(*right*, **R**)

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

compare(*value*, *v*)

value = *v*

We need to remove *value* at the root. We have a helper function for that: removeRoot.

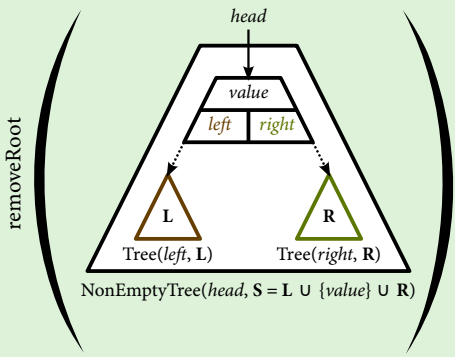


Tree(*left*, **L**) Tree(*right*, **R**)

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

***nhead* = removeRoot(*head*). Return *nhead*.**

removeRoot returns tree representing **L** ∪ **R**, which is **S** \ {*value*}. Pass up return value.



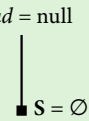
Tree(*left*, **L**) Tree(*right*, **R**)

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

Tree(*nhead*, **L** ∪ **R**)

head = null

The tree at *head* is empty, and represents the set ∅. ∅ \ {*value*} = ∅. Return *head*.

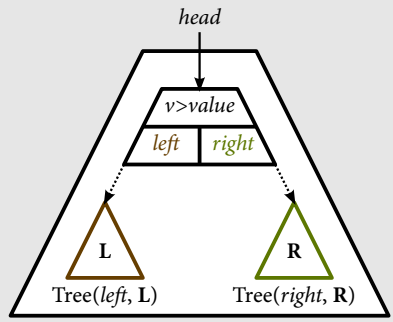


EmptyTree(*head*, **S**)

value < *v* (symmetrical)

value < *v*

{*v*} and **R** do not contain *value*, and so we have removed from them. We now remove from **L**.

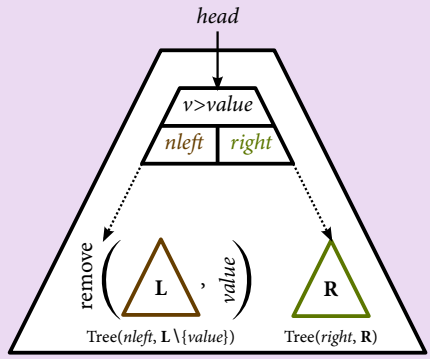


Tree(*left*, **L**) Tree(*right*, **R**)

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

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

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



Tree(*nleft*, **L** \ {*value*}) Tree(*right*, **R**)

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