



Rank Annotated Trees

Todor Peev, Vincent Trélat

École Nationale Supérieure des Mines de Nancy
Département Informatique

May 23, 2022

① Introduction

② Proofs

Functions

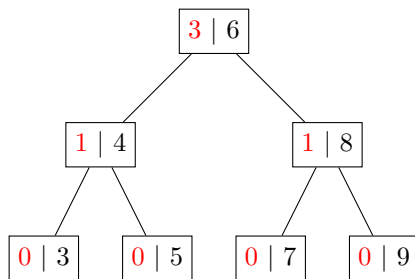
First lemmas

③ Inorder traversal and getting rank

Definition of the function

Proof of correctness

Definition



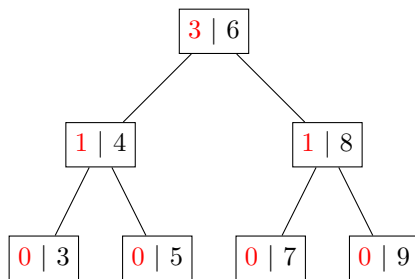
(Type definition)

```
datatype 'a rtree = Leaf | Node "'a rtree" nat 'a "'a rtree"
```

Example 1

```
<<<<>, 0, 3, <>>, 1, 4, <>>, 3, 6 :: nat, <<<<>, 0, 7, <>>, 1, 8, <<>, 0, 9, <>>>>
```

Definition



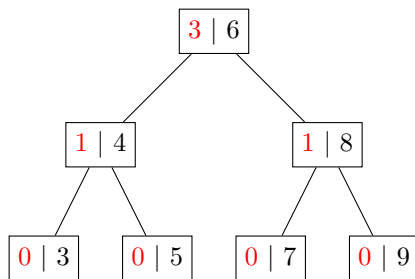
(Type definition)

```
datatype 'a rtree = Leaf | Node "'a rtree" nat 'a "'a rtree"
```

Example 1

```
<<<<>, 0, 3, <>>, 1, 4, <>>, 3, 6 :: nat, <<<<>, 0, 7, <>>, 1, 8, <<<<>, 0, 9, <>>>>
```

Definition



(Type definition)

```
datatype 'a rtree = Leaf | Node "'a rtree" nat 'a "'a rtree"
```

Example 1

```
<<<<, 0, 3, <>>, 1, 4, <>>, 3, 6 :: nat, <<<<, 0, 7, <>>, 1, 8, <<>, 0, 9, <>>>>
```

First functions

```
fun num_nodes :: "'a rtree  $\Rightarrow$  nat" where
  "num_nodes  $\langle \rangle$  = 0" |
  "num_nodes  $\langle l, -, -, r \rangle$  = 1 + num_nodes l + num_nodes r"
```

```
fun rbst :: "('a::linorder) rtree  $\Rightarrow$  bool" where
  "rbst  $\langle \rangle$  = True" |
  "rbst  $\langle l, n, x, r \rangle$  = (( $\forall a \in \text{set\_rtree } l. a < x$ )  $\wedge$ 
    ( $\forall a \in \text{set\_rtree } r. x < a$ )  $\wedge$ 
    rbst l  $\wedge$ 
    rbst r  $\wedge$ 
    n = num_nodes l)"
```

First functions

```
fun num_nodes :: "'a rtree ⇒ nat" where
  "num_nodes ⟨⟩ = 0" |
  "num_nodes ⟨ l, -, -, r ⟩ = 1 + num_nodes l + num_nodes r"
```

```
fun rbst :: "('a::linorder) rtree ⇒ bool" where
  "rbst ⟨⟩ = True" |
  "rbst ⟨ l, n, x, x ⟩ = ((∀a ∈ set_rtree l. a < x) ∧
    (∀a ∈ set_rtree r. x < a) ∧
    rbst l ∧
    rbst r ∧
    n = num_nodes l)"
```

Some useful lemmas


```
lemma set_rtree_rbst:  
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨l, n, x, r⟩ ⇒ a < x ⇒  
  a ∈ set_rtree l"
```

```
lemma rins_set: "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:  
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar: "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

```
lemma set_rtree_rbst:  
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨l, n, x, r⟩ ⇒ a < x ⇒  
  a ∈ set_rtree l"
```

```
lemma rins_set: "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:  
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar: "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

```
lemma set_rtree_rbst:  
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨l, n, x, r⟩ ⇒ a < x ⇒  
  a ∈ set_rtree l"
```

```
lemma rins_set: "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:  
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar: "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

```
lemma set_rtree_rbst:  
  "rbst ⟨l, n, x, r⟩ ⇒ a ∈ set_rtree ⟨l, n, x, r⟩ ⇒ a < x ⇒  
  a ∈ set_rtree l"
```

```
lemma rins_set: "set_rtree (rins x t) = insert x (set_rtree t)"
```

```
lemma num_nodes_rins_notin:  
  "x ∉ set_rtree t ⇒ rbst t ⇒ num_nodes (rins x t) = 1 + num_nodes t"
```

```
lemma rins_invar: "x ∉ set_rtree t ⇒ rbst t ⇒ rbst (rins x t)"
```

Inorder traversal and getting rank

- Tree traversal : inorder function (in-order DFS)
- Getting rank : rank function w.r.t. the structure of the rank annotated tree

```
fun rank:: "'a::linorder ⇒ 'a rtree ⇒ nat" where
  "rank a ⟨⟩ = 0" |
  "rank a ⟨ l, n, x, r ⟩ =
    (if a = x then n
     else if a > x then 1 + n + rank a r
     else rank a l)"
```

- Selection : select function

- Tree traversal : inorder function (in-order DFS)
- Getting rank : rank function w.r.t. the structure of the rank annotated tree

```
fun rank:: "'a::linorder ⇒ 'a rtree ⇒ nat" where
  "rank a ⟨⟩ = 0" |
  "rank a ⟨ l, n, x, r ⟩ =
    (if a = x then n
     else if a > x then 1 + n + rank a r
     else rank a l)"
```

- Selection : select function

- Tree traversal : inorder function (in-order DFS)
- Getting rank : rank function w.r.t. the structure of the rank annotated tree

```
fun rank:: "'a::linorder ⇒ 'a rtree ⇒ nat" where
  "rank a ⟨⟩ = 0" |
  "rank a ⟨ l, n, x, r ⟩ =
    (if a = x then n
     else if a > x then 1 + n + rank a r
     else rank a l)"
```

- Selection : select function

(Selection)

```

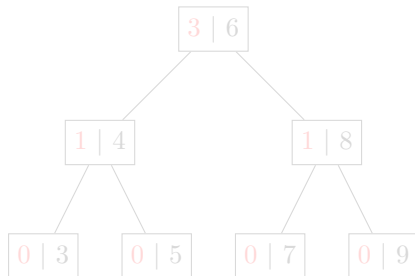
fun sel:: "nat ⇒ 'a::linorder rtree ⇒ 'a" where
  "sel _ ⟨⟩ = undefined" |
  "sel i ⟨l, n, x, r⟩ =
    (if i = n then x
     else if i < n then sel i l
     else sel (i - n - 1) r)"

```

sel 4 t

inorder t = $\underbrace{[3, 4, 5, 6]}_l, \underbrace{[7, 8, 9]}_r$

offset: n + 1



(Selection)

```

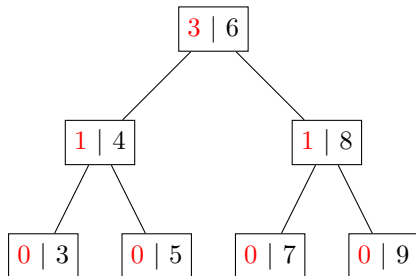
fun sel :: "nat ⇒ 'a::linorder rtree ⇒ 'a" where
  "sel _ ⟨⟩ = undefined" |
  "sel i ⟨l, n, x, r⟩ =
    (if i = n then x
     else if i < n then sel i l
     else sel (i - n - 1) r)"

```

sel 4 t

inorder t = $\underbrace{[3, 4, 5, 6]}_l, \underbrace{[7, 8, 9]}_r$

offset: $n + 1$



```
lemma select_correct:  
  "rbst t  $\implies$  i < length (inorder t)  $\implies$  sel i t = inorder t[i]"
```

Idea of the proof:

- By induction on t with i arbitrary
- Cases are split w.r.t. the body of the function `sel`
- In the third case, we show:
 - `sel i <l, n, x, r> = sel (i - n - 1) r`
 - `sel (i - n - 1) r = inorder r[i - n - 1]`

```
lemma rank_sel_id:  
  "rbst t  $\implies$  i < length (inorder t)  $\implies$  rank (sel i t) t = i"
```

Idea of the proof:

- By induction on `t` with `i` arbitrary
- Trivialized with lemma `select_correct`