



AVL

Beatriz Alves e Carolina Cruz



Definições criadas

Definição indutiva de **tree**

```
Inductive tree : Type :=  
  | Nil  
  | Node (v : nat) (l r : tree).
```

Definição indutiva de **BST**

```
Inductive BST : tree -> Prop :=  
  | BST_Empty: BST Nil  
  | BST_Node (v : nat) (l r : tree):  
    BST l ->  
    BST r ->  
    ForallT l (fun x => x < v) ->  
    ForallT r (fun x => x > v) ->  
    BST (Node v l r).
```

Definições criadas

Definição indutiva de **Balanced Tree**

```
Inductive BT : tree -> Prop :=
| BT_Empty: BT Nil
| BT_Node_Eq (v : nat) (l r : tree):
  BT l ->
  BT r ->
  height r = height l ->
  BT (Node v l r)
| BT_Node_R (v : nat) (l r : tree):
  BT l ->
  BT r ->
  height r = S (height l) ->
  BT (Node v l r)
| BT_Node_L (v : nat) (l r : tree):
  BT l ->
  BT r ->
  height l = S (height r) ->
  BT (Node v l r).
```

Definição de **AVL**

```
Definition AVL (t : tree) : Prop :=
  BST t /\ BT t.
```

Definição indutiva de **Diff**

```
Inductive Diff : Type :=
| Zero
| One
| MinusOne
| Two
| MinusTwo
| Impossible.
```

Funções criadas

- **Inserção** de árvore **BST** `Fixpoint insert (t : tree) (v: nat): tree`
- **Busca** de **BST** `Fixpoint search (t:tree) (v:nat) : bool`
- **Altura** de uma **tree** `Fixpoint height (t : tree) : nat`
- **Inserção** de árvore **AVL** `Fixpoint insertAVL (t : tree) (v: nat): tree`

Funções auxiliares importantes

- **ForallT** para definição de **BST** `Fixpoint ForallT (t : tree) (P : nat -> Prop) : Prop`
- **Presença** em **tree** `Fixpoint InT (t : tree) (v : nat) : Prop`
- **Diferença de altura** entre sub-árvores de uma **tree** `Definition diff (t : tree) : Diff`
- **Rotação** à esquerda e à direita para inserção em **AVL** `Definition left_rotate (t: tree) : tree`
- **Rebalanceamento** à esquerda e à direita para inserção em uma **AVL** `Definition rebalance_left (t : tree) : tree`
- **Rebalanceamento** geral para inserção em uma **AVL** `Definition rebalance (t : tree) : tree`

Principais Teoremas

Theorem **insertBST**: forall (v : nat) (t : tree),
BST t -> BST (insert t v).

Theorem **rebalance_BST**: forall (t : tree),
BST t -> BST (rebalance t).

Theorem **insertAVL_BST**: forall (t : tree) (v : nat),
BST t -> BST (insertAVL t v).

Theorem **searchBST**: forall (v : nat) (t : tree),
BST t -> search t v = true <-> InT t v.

Theorem **insertAVL_AVL**: forall (t : tree) (v : nat),
AVL t -> AVL (insertAVL t v).

Theorem **insertAVL_BT**: forall (t : tree) (v : nat),
BT t -> BT (insertAVL t v).

Lemmas auxiliares importantes

Lemma **left_rotate_BST**: forall (t : tree),
BST t -> BST (left_rotate t).

Lemma **rebalance_left_BST**: forall (t : tree),
BST t -> BST (rebalance_left t).

Lemma **rebalance_left_BT**: forall (l r : tree) (v : nat),
BT l ->
BT r ->
S (S (height l)) = height r ->
BT (rebalance_left (Node v l r)).

Lemmas auxiliares importantes - ForallT

Lemma **ForallT_InT**: forall (t : tree) (v : nat) (P : nat -> Prop),
ForallT t P -> InT t v -> P v.

Lemma **ForallT_left_rotate**: forall (P : nat -> Prop) (t : tree),
ForallT t P -> ForallT (left_rotate t) P.

Lemma **ForallT_rebalance_left**: forall (P : nat -> Prop) (t : tree),
ForallT t P -> ForallT (rebalance_left t) P.

Lemma **ForallT_rebalance**: forall (P : nat -> Prop) (t : tree),
ForallT t P -> ForallT (rebalance t) P.

Lemma **ForallT_insert**: forall (t : tree) (P : nat -> Prop) (v : nat),
P v -> ForallT t P -> ForallT (insert t v) P.

Lemma **ForallT_insertAVL**: forall (P : nat -> Prop) (t : tree) (v : nat),
P v -> ForallT t P -> ForallT (insertAVL t v) P.

Lemmas auxiliares importantes - diff

Lemma **diff_Zero** : forall v l r,
diff (Node v l r) = Zero <-> height l = height r.

Lemma **diff_One** : forall v l r,
diff (Node v l r) = One <-> height l = S (height r).

Lemma **diff_MinusOne** : forall v l r,
diff (Node v l r) = MinusOne <-> S (height l) =
height r.

Lemma **diff_Two** : forall v l r,
diff (Node v l r) = Two <-> height l = S (S (height
r)).

Lemma **diff_MinusTwo** : forall v l r,
diff (Node v l r) = MinusTwo <-> S (S (height l)) =
height r.

Lemmas auxiliares importantes

Lemma **insertAVL_height**: forall (t : tree) (v: nat),
BT t -> height (insertAVL t v) = S (height t) \vee height (insertAVL t v) = height t.

Dificuldades

- Achar uma alternativa para não precisar usar inteiros
- Prova de teoremas envolvendo **insertAVL** (*insertAVL_BT* e *insertAVL_BST*)
 - *insertAVL_BT* precisou de lemmas bem complicados, um deles (*insertAVL_height*) ficou com uma prova de mais de 200 linhas
 - *insertAVL_BST* precisou de muitos lemmas, um ForallT para cada função auxiliar



Obrigada!

CREDITS: This presentation template was created
by **Slidesgo**, including icons by **Flaticon**, and
infographics & images by **Freepik**


