Cours 4 Sophie Laplante 6/10/14

Propriété ABR

$\forall \texte{A}, v \in \texte{A}$

v < A val <=> c dans A.gauche v > A val <=> c dans A.droite

```
def insertion (A, v): if A.val == v: return A if A.val > v: .... if A.val < v: ....
```

1. Objectif

Démontré la complexité de l'algo d'insertion ABR

2. Quelques proposition & définition

2.1. définition

Pour un arbre binaire étiquetté

1. |A| = nombre de valeur dans A
2. 1. Pour tout sous arbre A' de A

Prof_A(A') = longueur du chemin de la racine de A jusqu'a la racine de A'

“

$$prof\_A(A)=0\ prof\_A(A.gauche)=1$$

1. Pour tout v dans A

“

$$Prof\_A(v) = Prof\_A(A')\ ou\ A'\ sous\ arbre\ A'.val = v$$

1. `h(A) = max( Prof_A(v))` v dans A ??? = -1 si A vide???

“

$$h(Arbre\ vide) = ??\ h(Feuille) = 0$$

On utilise un lemme qui exprime ces valeurs "recursivement", en fonction des sous-arbre gauche & droite

3. Lemme

1. |A| = 1 + |A.gauche| + |A.droite|
2. h(A) = 1 + max{h(A.gauche), h(A.droite)}

“

$$si\ A = Feuille\ h(A) = 0\ h(A.gauche) = -1\ car\ A.gauche\ vide$$

3. ``` Prof_A(v) = 0 si v = A.val 1 + Prof_A.gauche(v) si v < A.val 1 + Prof_A.droite(v) si v > A.val

Pour tout v dans A ```

4. Prop count = NB comparaison ABR

`Trouver ABR(A, v) <= 3(prof_A(v) + 1) si v dans A 3(h(v) + 1) si v pas dans A`

4.1. Preuve

On démontre l'énoncé pour tout A de taille n >= 1 Par réccurence sur n

- Cas de base n = 1 si v dans A : prof_A(v) = 0 On fait 1 comparaison hpr OK si v pas dans A si v < A: l'algo fait 2 comparaison '== et <' si v > A: l'algo fait 3 comparaison '==, < et >' Donc 3 comparaison Max hpr OK
- Hérédité Soit A un arbre et |A| = n+1 supposons l'énoncé vrai sur les arbres de taille <= 1
- Cas 1 v == A.val donc v dans A il faut montrer que cout <= 3(prof_A(v) + 1) 1 comparaison faite HP OK
- Cas 2 v < A.val Dans ce cas le #de comparaison faite par l'algo .....

```
def supprimerABR(A,v): si pas de sous arbre : suprimer si un sous arbre : on remonte le sous arbre sinon on remplace la valeur par son prédéseur (<) A'=trouverABR(A,v) if A' != None if A'.gauche supprimer(A') elif A'.gauche == None A'.gauche = A'.droite.gauche
```

```
A'.val = A'.droite.val A'.droite = A'.droite.droite elif A'.doite ==
None ... else A'.val = deleteMAx(A'.gauche)
```