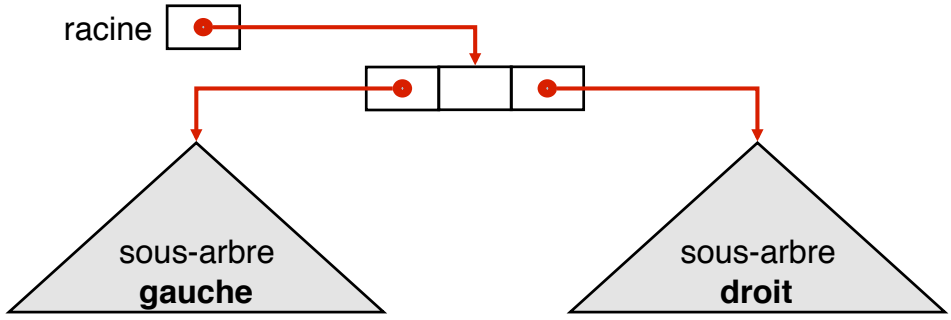
Algorithmique & programmation

Chapitre 5 : Arbres

Algorithmes sur les arbres

Calcul de la taille

- Vers le schéma récursif



- Schéma récursif

➤ racine⁺ = <> ⇨ résultat = 0 ✳

➤ racine⁺ ≠ <> ⇨

résultat = 1
+ taille (racine↑.gauche)
+ taille (racine↑.droit) ;

Calcul de la taille

fonction taille(d racine:pointeur):entier;
spécification *{}* ➔ *{résultat = nombre de noeuds de racine⁺}*
debfonc
    si racine = nil alors
        retour 0 ;
    sinon
        retour 1 + taille (racine↑.gauche) + taille (racine↑.droite) ;
    finsi ;
finfonc ;

- Parcours préfixé car traitement de la racine effectué avant les appels récursifs
- Parcours infixé
  retour taille(racine↑.gauche) + 1 + taille(racine↑.droit)
- Parcours postfixé
  retour taille (racine↑.gauche) + taille (racine↑.droit) + 1

Calcul de la taille (**ada**)

```
fonction taille (racine : in Ta_noeud) return integer is
--spec {} ➔ {résultat = nombre de noeuds de racine⁺}
begin
   if racine = null then
     return 0 ;
   else
     return 1 + taille (racine.all.gauche)
                              + taille (racine.all.droite) ;
   end if ;
end taille ;
```

Calcul du nombre de feuilles

- Feuille : nœud qui n'a pas de fils

- Déterminer si un noeud est une feuille

fonction feuille (d nœud : pointeur) : booléen ;
spécification *{noeud ≠ nil}* ➔ *{résultat = noeud est une feuille}*
debfonc
    retour (noeud↑.gauche = nil) et (noeud↑.droit = nil) ;
finfonc ;

Calcul du nombre de feuilles

- Schéma récursif

➤ racine⁺ = <> ⇨ résultat = 0 ✳

➤ racine⁺ ≠ <>

➤➤ racine⁺ est une feuille
⇨ résultat = 1 ✳

➤➤ racine⁺ n'est pas une feuille
⇨ résultat = nbfeuille (racine↑.gauche)
+ nbfeuille (racine↑.droit) ;

Calcul du nombre de feuilles

fonction nbfeuilles (d racine : pointeur) : entier;
spécification *{}* ➔ *{résultat = nombre de feuilles de racine⁺}*
debfonc
    si racine = nil alors
        retour 0 ;
    sinonsi feuille (racine) alors
        retour 1 ;
    sinon
        retour nbfeuille (racine↑.gauche) + nbfeuille (racine↑.droit) ;
    finsi ;
finfonc ;

Calcul du nombre de feuilles (**ada**)

```
function nbfeuilles (racine : in Ta_Noeud)
                                        returns integer is
--spec {} ➔ {résultat = nb de feuilles de racine+}
begin
   if racine = null then
     return 0 ;
   else if feuille (racine) then
     return 1 ;
   else
     return nbfeuille (racine.all.gauche)
                      + nbfeuille (racine.all.droit) ;
   end if ;
end nbfeuilles;
```

Recherche associative

fonction rech (d racine:pointeur; d val : t) : booléen ;
spécification *{ }* ➔ *{résultat= (val ∈ racine⁺)}*

☐ Schéma récursif

➤ racine⁺ = <> ⇨ résultat = faux ✳
➤ racine⁺ ≠ <> ⇨
➤➤ racine↑.info = val ⇨ résultat = vrai ✳
➤➤ racine↑.info ≠ val ⇨
➤➤➤ rech(racine↑.gauche,val)
⇨ résultat = vrai ✳
➤➤➤ non rech(racine↑.gauche,val)
⇨ résultat = rech(racine↑.droit,val) ;

Recherche associative

fonction rech (d racine : pointeur ; d val : t) : booléen ;
spécification *{ }* ➔ *{résultat= (val ∈ racine⁺)}*
debfonc
si racine = nil alors *{val ∉ racine⁺}*
retour faux ;
sinonsi racine↑.info = val alors *{val ∈ racine⁺}*
retour vrai ;
sinonsi rech(racine↑.gauche, val) alors
retour vrai ;
sinon
retour rech (racine↑.droit, val) ;
finsi ;
finfonc ;