

Objectifs

À la fin de cette séance, vous devriez être capable de :

- utiliser des algorithmes de parcours d’arbre binaire pour résoudre des problèmes simples ;
- déterminer le coût d’un algorithme de parcours d’arbre.

Exercice 1 : Échauffement

- Écrivez une fonction qui calcule le nombre de feuilles d’un arbre binaire.
- Écrivez une fonction qui renvoie l’arbre « miroir » d’un arbre binaire (dans lequel les fils gauche et droit de chaque nœud ont été inversés).

Pour écrire ces fonctions, vous avez besoin de manipuler un type abstrait `arbre`. Vous utiliserez les primitives d’accès définies en cours.

Exercice 2 : Feuille haute

Pour un arbre binaire A , écrivez une fonction qui calcule la hauteur de la feuille la plus haute (la feuille la plus proche de la racine)

1. par un parcours en profondeur d’abord récursif,
2. par un parcours en largeur d’abord.
3. Justifiez pourquoi ces deux algorithmes (récursif et itératif) renvoient bien la valeur demandée.

Correction de l’exercice 2

- ```

FeuilleHaute(a)
si a est une feuille :
 | Renvoyer 0
sinon si estVide(FilsGauche(a)) :
1. | Renvoyer 1 + FeuilleHaute(FilsDroit(a))
sinon si estVide(FilsDroit(a)) :
 | Renvoyer 1 + FeuilleHaute(FilsGauche(a))
sinon :
 | Renvoyer 1 + minimum(FeuilleHaute(FilsGauche(a)), FeuilleHaute(FilsDroit(a)))
2. Dans un parcours en largeur, la première feuille rencontrée est la moins profonde.

```

FeuilleHauteLargeur(  $a$  )

$F$  = FileVide de couples (noeud, entier)

$F$  = Enfiler ( $a,0$ ) dans  $F$

$x = a$

$h = 0$

**tant que**  $x$  n’est pas une feuille :

```

 | (x,h) = Tête(F)
 | F = Défiler(F)
 | si x n’est pas vide :
 | | F = Enfiler (FilsGauche(x), $h+1$) dans F
 | | F = Enfiler (FilsDroit(x), $h+1$) dans F

```

Renvoyer  $h$

3. a) arrêt :
  - version récursive : les appels récursifs sont faits sur des noeuds dont la distance aux feuilles décroît strictement (de 1). On arrivera donc forcément à une feuille et à l’arrêt de la récursivité. (car les arbres dont on parle sont finis).
  - version itérative : c’est un parcours par niveau de l’arbre qui se termine lorsqu’on rencontre la première feuille.

b) correction :

- version récursive : la procédure est correcte dans le cas de base (avec la convention qu’un arbre réduit à une feuille est de hauteur 0). En supposant que les appels récursifs sont corrects (ils renvoient la hauteur de la feuille la plus haute de chacun des sous-arbres de  $a$ ), alors le calcul  $1 + \min(\dots, \dots)$  donne bien la hauteur de la feuille la plus haute de  $a$ .
- version itérative : on enfile les couples (noeud, hauteur du noeud) par hauteur croissante. La première feuille défilée est la plus haute, et on récupère sa hauteur.

### Exercice 3 : Feuilles à la profondeur $p$

Pour un arbre binaire  $A$  et une profondeur  $p$ , écrivez une fonction qui calcule le nombre de feuilles à la profondeur  $p$

1. par un parcours en profondeur d’abord récursif,
2. par un parcours en largeur d’abord.

Pour chacun de ces deux algorithmes (récursif et itératif), que pouvez vous dire de son coût ? Y-a-t-il un algorithme moins coûteux que l’autre ?

### Correction de l’exercice 3

1. NbFeuillesProf( $a, p$ )

si  $p = 0$  :

    si  $a$  est une feuille :

        └ Renvoyer 1

    sinon :

        └ Renvoyer 0

sinon :

    si  $a$  est vide :

        └ Renvoyer 0

    sinon :

        └ Renvoyer NbFeuillesProf(FilsGauche( $a$ ),  $p-1$ ) + NbFeuillesProf(FilsDroite( $a$ ),  $p-1$ )

2. NbFeuillesProfLargeur(  $a, p$  )

$F$  = FileVide de couples (noeud, entier)

$F$  = Enfiler ( $a, p$ ) dans  $F$

$n = 0$

tant que  $F$  n’est pas vide :

    ( $x, p$ ) = Tête( $F$ )

$F$  = Défiler( $F$ )

    si  $p = 0$  :

        └  $n = n + 1$

    si  $x$  n’est pas vide :

        └  $F$  = Enfiler (FilsGauche( $x$ ),  $p-1$ ) dans  $F$

        └  $F$  = Enfiler (FilsDroite( $x$ ),  $p-1$ ) dans  $F$

Renvoyer  $n$

Le parcours en profondeur ne fait pas les appels récursifs lorsque son argument  $p$  vaut 0. Il ne parcourt donc que les nœuds dont la profondeur est inférieure ou égale à la valeur initiale de  $p$  : dans le pire des cas il y en a  $2^p - 1$ .

Le parcours en largeur, tel qu’il est écrit ici, parcourt systématiquement tout l’arbre, son coût est donc supérieur.

Cependant, il serait possible de l’améliorer en n’enfilant de nouveaux nœuds dans  $F$  que si  $p > 0$  : le coût serait alors le même que celui du parcours en profondeur.