Algorithmique
Arbres généraux : parcours

Plusieurs manières de parcourir (explorer tous les nœuds) un arbre :

— Le *parcours profondeur* (*depth-first traversal* en anglais), par nature récursif, qui consiste à avancer le plus loin possible.

— Le *parcours largeur* (*breadth-first traversal* en anglais), itératif, qui consiste à passer en revue tous les nœuds de l'arbre niveau par niveau.

1 Le parcours profondeur

"À la main"

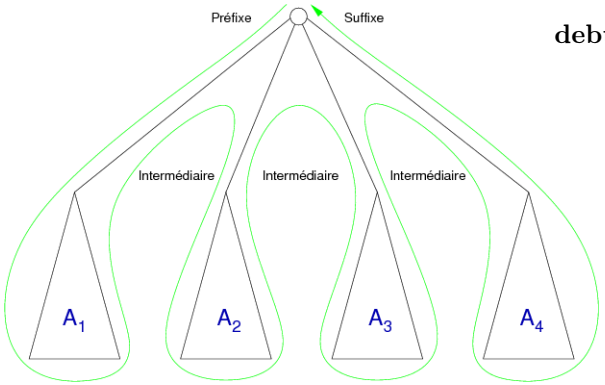
Parcourir un arbre en profondeur : on part de la racine et on suit le lien vers le premier fils, puis vers son premier fils et ainsi de suite le plus loin possible. Lorsqu'on arrive sur une feuille, on remonte vers son père pour passer à son frère s'il existe, sinon en remonte vers le père du père et ainsi de suite. Le parcours se termine lorsque la remontée nous ramène à la racine de l'arbre.

Un algorithme récursif

La manière la plus simple d'écrire ce parcours est récursive (il suit la structure récursive de l'arbre) :

— L'arbre réduit à une feuille sera un cas d'arrêt.

— Si l'arbre possède des sous-arbres, ils sont parcourus récursivement, de gauche à droite.




```
procedure profondeur(arbre A)
variables
    entier  i, nbFils
debut
    si  feuille(A) alors
        /* traitement feuille */
    sinon
        /* traitement préfixe */
        nbFils ← nbarbres(listearbres(A))
        pour i ← 1 jusqu'a nbFils - 1 faire
            profondeur(ieme(listearbres(A), i))
            /* traitement intermédiaire (i) */
        fin pour
        profondeur(ieme(listearbres(A), nbFils))
        /* traitement suffixe */
    fin si
fin
```

Ordres induits

Lors du parcours en profondeur, chaque nœud est rencontré $n+1$ fois avec n le nombre de fils.

— avant de descendre sur le premier fils : ordre **préfixe**,

— en remontant de chaque fils, avant de descendre au suivant : ordre **intermédiaire**,

— en remontant du dernier fils : ordre **suffixe**.

Les rencontre en préfixe et suffixe sont les *ordres induits* du parcours profondeur.

Pour éviter de faire un traitement intermédiaire de trop, après la remontée du dernier fils, il faut sortir le parcours de celui-ci de la boucle, comme dans l'algorithme présenté. S'il n'y a pas de traitement intermédiaire, on peut alors simplifier l'algorithme en laissant la boucle parcourir tous les fils. On peut également se passer du cas d'arrêt *feuille* lorsque celles-ci ne font pas l'objet d'un traitement particulier.

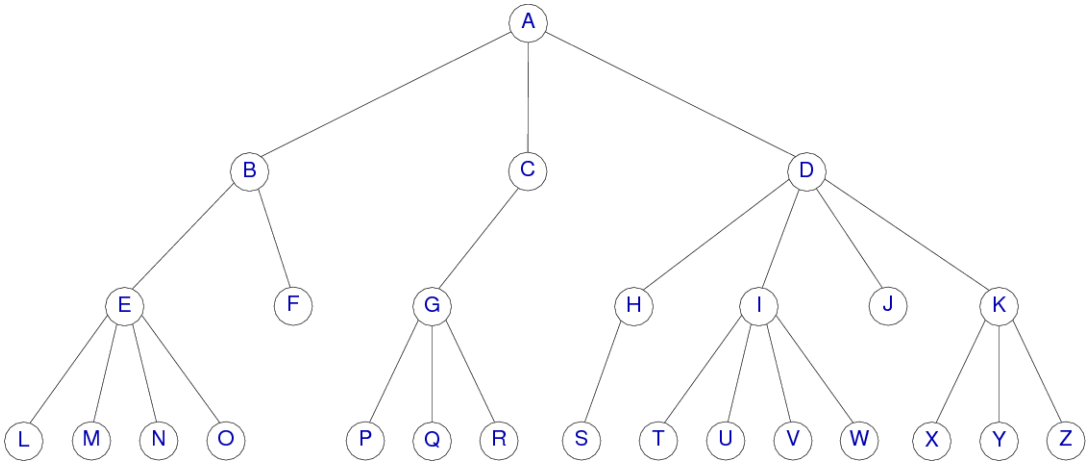
2 Le parcours largeur

L'algorithme classique utilise une *file* :

- on enfile la racine de l'arbre
- tant que la file n'est pas vide :
  - on récupère et on défile le premier élément de la file
  - on enfile chacun de ses fils, de gauche à droite.

```
procedure parcours_largeur(arbre A)
    variables
        file    f
        entier  i
debut
    f ← filevide
    f ← enfiler (A, f)
    tant que non estvide(f) faire
        A ← premier(f)
        f ← defiler (f)
        /* traitement contenu(racine(A)) */
        pour i ← 1 jusqu'a  nbarbres(listearbres(A) faire
            f ← enfiler(ieme(listearbres(A), f)
        fin pour
    fin tant que
fin
```

Annexe : exemples



Ordres induits du parcours profondeur

Préfixe : A B E L M N O F C G P Q R D H S I T U V W J K X Y Z

Suffixe : L M N O E F B P Q R G C S H T U V W I J X Y Z K D A

Parcours largeur :

De A à Z en ordre alphabétique !