Arbres binaires (parcours)

Vidéo 1 : https://www.youtube.com/watch?v=haPbqkccUaA

Vidéo 2 : https://www.youtube.com/watch?v=BKwAORXlXnI

Parcours profondeur (*depth-first traversal*)

Le parcours profondeur est un parcours par nature **récursive**, qui consiste à avancer le plus loin possible.

▼ **Parcours (récursif)**

cas d'arrêt : arbrevide

cas général :

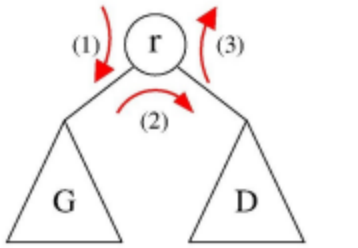
- **main gauche** → parcours(arbregauche), parcours(arbredroit)
- **main droite** → parcours(arbredroit), parcours(arbregauche)

▼ **Ordres induits**

Parcours préfixe : *racine* - SAG - SAD

Parcours infixe : SAG - *racine* - SAD

Parcours suffixe : SAG - SAD - *racine*



(1) : préfixe (2) : infixe (3) : suffixe

```
procedure profondeur(arbrebinaire B)
debut
    si B = arbrevide alors
        /* terminaison */
    sinon
        /* traitement préfixe (1) */
        profondeur(g(B))
        /* traitement infixe (2) */
        profondeur(d(B))
        /* traitement suffixe (3) */
    fin si
fin
```

Parcours largeur (breadth-*first traversal*)

Le parcours profondeur est un parcours par nature **itérative,** qui consiste à passer en revue tous
les nœuds de l'arbre niveau par niveau.

▼ Parcours (itératif)

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 (s'ils existent), d'abord le gauche puis le droit

```
procedure parcours_largeur(arbrebinaire B)
    variables
        file    f
debut
    si B <> arbrevide alors
        f ← filevide
        f ← enfiler (B,f)
        tant que non estvide(f) faire
            B ← premier(f)
            f ← defiler (f)
             /* traitement contenu(racine(B)) */
            si g(B) <> arbrevide alors
                f ← enfiler(g(B), f)
            fin si
            si d(B) <> arbrevide alors
                f ← enfiler(d(B), f)
            fin si
        fin tant que
    fin si
fin
```