Les arbres II : arbres binaires

Capacités attendues

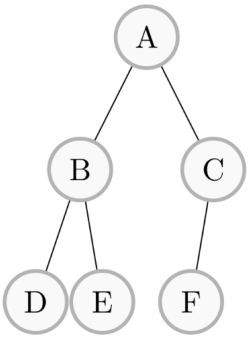
- ✓ Notions de nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits ;
- ✓ Évaluer quelques mesures des arbres binaires (taille, encadrement de la hauteur, etc.) ;
- ✓ Calculer la taille et la hauteur d'un arbre ;
- ✓ Parcourir un arbre de différentes façons (ordres infixe, préfixe ou suffixe ; ordre en largeur d'abord) ;
- ✓ Une structure de données récursive adaptée est utilisée ;
- ✓ L'exemple des arbres permet d'illustrer la programmation par classe.

Définition

En informatique, un **arbre binaire** est un arbre d'**arité** 2, ce qui signifie que chaque nœud admet au plus deux enfants, appelés **sous-arbre gauche** et **sous-arbre droit**.

Exercice 1

1. Entourer les sous-arbres gauche et droit de la racine de l'arbre binaire ci-dessous.



2. Dans le cours précédent intitulé « Arbres 1 : généralités », quels arbres sont des arbres binaires parmi ceux représentés en 1.1, 1.2 et 1.3 ?

1 Mesures

Vocabulaire

Ces définitions sont données à titre indicatif, pour les besoins des démonstrations suivantes.

- Un arbre binaire **parfait** est un arbre binaire dont toutes les feuilles ont la même profondeur et dont les nœuds qui ne sont pas des feuilles comptent exactement 2 enfants.
- Un arbre binaire **filiforme** est un arbre binaire dont tous les nœuds ont au plus un enfant.

1.1 Encadrement de la taille

Exercice 2

1. (a) Dessiner un arbre binaire parfait de hauteur 2.

(b) Exprimer sa taille t en fonction de sa hauteur h.

2. (a) Dessiner un arbre binaire filiforme de hauteur 2.

(b) Exprimer sa taille t en fonction de sa hauteur h.

Encadrement de la taille d'un arbre binaire

La taille t d'un arbre binaire de hauteur h est comprise entre $h+1$ et $2^{h+1}-1$.

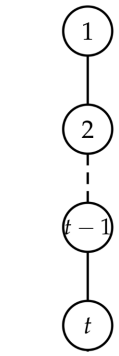
Son encadrement est donc $h+1 \leq t \leq 2^{h+1}-1$

Démonstration

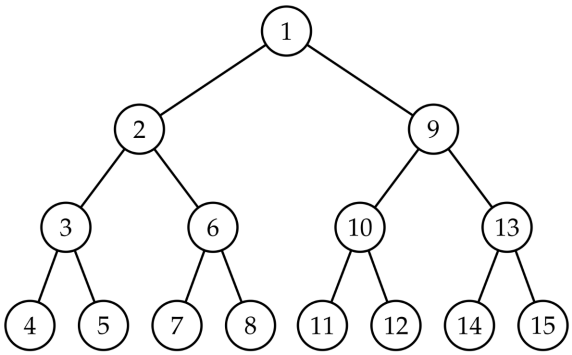
Pour une hauteur h donnée, la taille minimale d'un arbre (binaire ou non) est celle de l'arbre filiforme (dont tous les nœuds ont au plus un enfant). Sa taille t est égale à $h+1$.

La taille maximale d'un arbre binaire est celle de l'arbre binaire parfait de hauteur h dont tous les nœuds de profondeur inférieure à h ont deux enfants. Dans un tel arbre, il y a exactement 2^k nœuds de profondeur k, d'où une taille égale à

$$t = 1 + 2 + 2^2 + \cdots + 2^h = \sum_{k=0}^{h} 2^k = 2^{h+1} - 1\,.$$



Arbre filiforme
$t = h + 1$



Arbre parfait
$t = 2^{h+1} - 1$

1.2 Encadrement de la hauteur

Exercice 3

1. (a) Dessiner un arbre binaire parfait de taille 7.

(b) Quelle est sa hauteur ? *On ne demande pas de formule générale.*

2. (a) Dessiner un arbre binaire filiforme de taille 7.

(b) Exprimer sa hauteur h en fonction de sa taille t.

Encadrement de la hauteur d'un arbre binaire

La hauteur h d'un arbre binaire de taille t est comprise entre $\log(t)$ et $t-1$.

Son encadrement est donc $\log(t) \leq h \leq t-1$

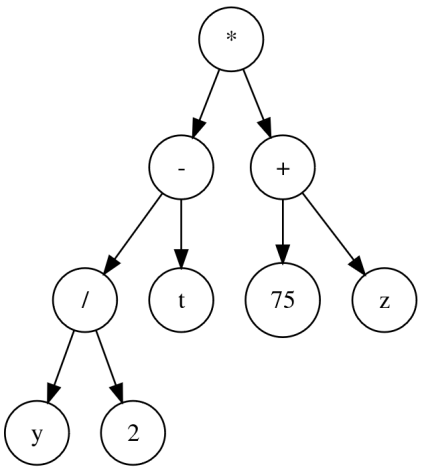
Démonstration

D'après la formule d'encadrement de la taille d'un arbre binaire, on a $h+1 \leq t$, donc $h \leq t-1$.

Par ailleurs, l'inégalité $t \leq 2^{h+1}-1$ s'écrit aussi $t < 2^{h+1}$. En appliquant le logarithme, il vient $\log(t) < h+1$, d'où $h > \log(t)-1$ et $h \geq \log(t)$.

Exercice 4

1. Quelle expression arithmétique l'arbre ci-dessous peut-t-il représenter ?



2. Dessiner l'arbre représentant $3 + (7/3 - 1)^3$

2 Algorithmes sur les arbres binaires

2.1 Calculs de mesures

Taille

La taille de l'arbre correspond à la **somme de la taille de ses sous-arbres**. On parcourt donc les sous-arbres gauche et droit récursivement, et on incrémente la taille à chaque fois que l'on rencontre un nouveau nœud. L'algorithme s'arrête lorsque l'appel récursif se fait sur un arbre vide.

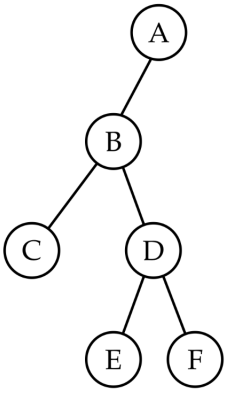
Hauteur

L'algorithme qui permet de calculer la hauteur d'un arbre binaire est assez similaire à celui utilisé pour la taille. La différence est qu'à chaque niveau de profondeur atteint (chaque nouvel appel récursif), on incrémente de 1 la hauteur par rapport au maximum des hauteurs entre le sous-arbre gauche et le sous-arbre droit.

Exercice 5

Récupérer sur notre site le code source arbres_b.zip nécessaire à faire les exercices suivants. Elle contient les fichiers `noeud.py` et `displayer.py`.

Décompresser l'archive à l'emplacement adéquat votre dossier de travail NSI (c'est-à-dire dans un sous-dossier dédié au chapitre courant).



1. Écrire un script qui permet de définir l'arbre ci-contre.

2. Compléter le code des méthodes `taille` et `hauteur`. (On peut utiliser la fonction `max` de Python pour sélectionner la plus grande des deux hauteurs.)
3. Écrire une méthode qui renvoie le nombre de feuilles d'un arbre binaire.

2.2 Les parcours

Parcours en profondeur d'abord

La parcours en profondeur d'abord *(Depth-First Search)* d'un arbre binaire consiste à explorer récursivement, en partant du nœud racine, les sous-arbres de chacun des nœuds. On donne ci-dessous une implémentation en langage naturel du parcours en profondeur d'abord.

```
fonction parcours_profondeur(noeud):
    si sous-arbre gauche est non vide :
        parcours_profondeur(sous-arbre gauche) # 1er appel récursif
    si sous-arbre droit est non vide :
        parcours_profondeur(sous-arbre droit) # 2ème appel récursif
```

Lors d'un tel parcours, il existe plusieurs façons de collecter les étiquettes des nœuds, selon que cette collecte se situe

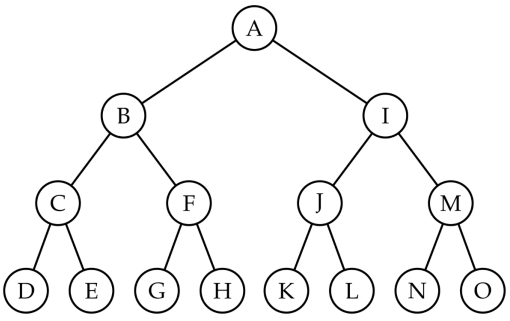
- avant le premier appel récursif : c'est l'ordre préfixe ;
- entre les deux appels récursifs : c'est l'ordre infixe ;
- après le deuxième appel récursif : c'est l'ordre suffixe (ou postfixe).

Parcours en largeur d'abord

Contrairement au précédent, ce parcours essaie toujours de visiter le nœud le plus proche de la racine qui n'a pas déjà été visité.
L'arbre est ainsi parcouru ligne par ligne ou étage par étage ou encore par niveaux de même hauteur.

Exercice 6

On considère l'arbre binaire ci-dessous.



Donner l'ordre d'affichage des étiquettes des nœuds de cet arbre pour les différents ordres possibles.

- Ordre préfixe : A, B, C, D, E, F, G, H, I, J, K, L, M, N, O
- Ordre infixe : D, C, E, B, G, F, H, A, K, J, L, I, N, M, O
- Ordre suffixe : D, E, C, G, H, F, B, K, L, J, N, O, M, I, A
- Ordre en largeur d'abord : A, B, I, C, F, J, M, D, E, G, H, K, L, N, O

Exercice 7

On propose le pseudo-code en langage naturel d'une fonction effectuant un des parcours en profondeur.

Cette fonction prend en argument le nœud racine d'un arbre et **renvoie** le tableau de type `list` des étiquettes de ses nœuds dans l'ordre de parcours.

Comme précédemment, les sous-arbres gauche et droit seront nommés respectivement `sag` et `sad`.

```
fonction parcours_mystere(Noeud noeud) :
    si noeud est vide
        renvoyer tableau vide
    fin si
    renvoyer parcours_mystere(sag) + [etiquette_noeud] + parcours_mystere(sad)
```

1. À quel parcours en profondeur ce programme correspond-il ? Justifier.

   Il s'agit du parcours infixe. On lit d'abord à gauche du nœud, puis le nœud et enfin à droite du nœud.

2. En utilisant le pseudo-code précédent, écrire en Python les fonctions :

   `parcours_prefixe`, `parcours_infixe`, `parcours_suffixe`

   qui prennent en argument le nœud racine d'un arbre et renvoient le tableau des étiquettes de ses nœuds selon les ordres préfixe, infixe et suffixe. **Tester** les fonctions avec les assertions pertinentes.

Exercice 8

```
fonction parcours_largeur(racine)
    si racine est vide
        renvoyer tableau vide
    fin si
    tableau res
    file = creer_file()
    file.enfiler(racine)

    tant que file non vide faire
        noeud = file.defile()
        ajouter noeud dans res
        si sag non vide
            file.enfile(sag)
        fin si
        si sad non vide
            file.enfile(sad)
        fin si
    fin tant que

    renvoyer res
```

1. En utilisant une file, écrire une fonction **itérative** `parcours_largeur` qui prend en argument le nœud racine d'un arbre et renvoie le tableau des étiquettes de ses nœuds (lecture de gauche à droite).

2. Copier le corps de la fonction précédente dans une nouvelle fonction `parcours_mystere_2` qui prend en paramètre un nœud appelé `racine` et remplacer la file par une pile puis traiter le sous-arbre droit en premier : quel type de parcours obtient-on ?

   On obtient un parcours en profondeur préfixe.