|  |
| --- |
|  |
| Structures de Données |
| *Problème 3 : Hauteur d'un arbre et nombre de nœuds* |
|  |
| **BARBESANGE Benjamin, GARÇON Benoît** |
| **mai 2015** |

|  |
| --- |
|  |

Table des matières

[Introduction 2](#_Toc418529512)

[Rappel du sujet 2](#_Toc418529513)

[Présentation du principe 2](#_Toc418529514)

[Schéma explicatif 2](#_Toc418529515)

[Algorithmes 3](#_Toc418529516)

[Principe 3](#_Toc418529517)

[Algorithme 4](#_Toc418529518)

[Lexique 4](#_Toc418529519)

[Trace de l’algorithme 5](#_Toc418529520)

[Cas 1 - Cas général 5](#_Toc418529521)

[Cas 2 - Arbre vide 7](#_Toc418529522)

[Cas 3 - Arbre avec seulement une feuille 8](#_Toc418529523)

[Cas 4 - Arbre avec seulement des liens verticaux 9](#_Toc418529524)

[Cas 5 - Arbre avec seulement un nœud 11](#_Toc418529525)

Introduction

# Rappel du sujet

Nous disposons d'un arbre à une seule racine implantée par lien vertical et lien horizontal. Chaque élément de l'arbre est un triplet composé de la valeur de l'élément, de l'adresse de sa liste chaînée de liens verticaux et l'adresse de la liste chaînée de liens horizontaux.

L'algorithme ci-présent a pour but de retourner la hauteur de l'arbre (c’est-à-dire le nombre maximum de niveaux) ainsi que le nombre de nœuds. Ce sera donc une procédure qui aura en sortie ces deux éléments.

Un nœud sera ici considéré comme un élément ayant au moins un successeur. Les feuilles n’étant donc pas considérées comme des nœuds.

# Présentation du principe

Le principe est d'effectuer un parcours en profondeur. A chaque fois que l'on empile, on compte un niveau d'arbre en plus car on ira systématiquement sur un lien vertical. Si le passage au lien vertical ne donne pas une feuille alors on a un nœud que l'on va compter.

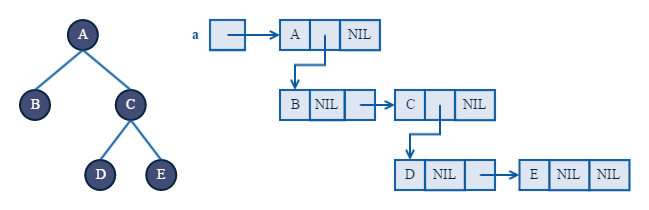
Lorsque l'on a terminé de parcourir le lien vertical, on compare notre niveau d'arbre actuel avec un maximum (qui sera initialisé à 0). Si cette valeur est supérieure au maximum alors le maximum change.

Ensuite tant que la pile n'est pas vide et que l'élément n'a pas de lien horizontal, on dépile et on passe au lien horizontal. On retranche également un niveau dans l'arbre à chaque fois.

L'algorithme se termine lorsque la pile est vide et que l'élément courant est à NIL. Ce qui signifie qu'il n'y a plus d'élément à traiter dans l'arbre.

## Schéma explicatif

On dispose d'un arbre à une racine implantée par lien vertical et horizontal.



L'algorithme renvoie 3 comme hauteur du niveau d'arbre et 2 comme nombre de nœuds.

Algorithmes

# Principe

Le principe de cet algorithme est simple : nous allons effectuer un parcours en profondeur itératif total de l’arbre, sans intégrer les feuilles à la pile, et nous allons simplement compter les données recherchées. Ici la taille de l’arbre (autrement dit le nombre d’éléments qu’il comporte) est supposée connue : effectivement on peut ainsi limiter la taille de la pile à N, N étant le nombre d’éléments dans l’arbre. Nous avons ainsi une complexité linéaire en O(N).

Nous supposerons dans cet exercice que tout ce qui concerne la pile (structure et algorithmes) est acquis et que ces travaux ne la concerne pas. C’est pourquoi nous admettons l’existence d’une structure de données « pile » ainsi que les algorithmes suivants : INIT(pile), LIBERER(pile), EMPILER(pile, valeur), DEPILER(pile), etc.

Voici le principe à proprement parlé :

* Initialisation de la pile
* Initialisation des variables
* Tant que l'élément courant n'est pas NIL [il y a encore des éléments à traiter]
  + Tant que l'élément courant n'est pas une feuille [on peut toujours descendre d'un niveau]
    - On empile son adresse [pour aller sur ses liens horizontaux plus tard]
    - On passe sur son lien vertical
    - On compte un niveau d'arbre en plus
    - On compte un nœud en plus
  + Fin Tant que
  + On compare le niveau actuel de l'arbre avec le maximum que l'on a pour le moment
  + Si le niveau d'arbre actuel est plus grand ou égal à notre niveau max Alors
    - Le max prend la valeur du niveau actuel+1
  + Fin Si
  + On passe sur le lien horizontal du courant
  + Tant que la pile n'est pas vide et que l'élément courant est NIL (pas de lien horizontal) Faire
    - On dépile un élément dans courant
    - On passe à son lien horizontal
    - On compte un niveau d'arbre en moins
  + Fin Tant que
* Fin Tant que
* On libère la pile
* Fin

# Algorithme

|  |
| --- |
| Procedure HAUTEUR\_NBNOEUDS(e : arb, taille ; s : nbNoeud, hauteur) |
| m(pile) := INIT(taille);  m(cour) := cm(arb);  m(hauteur) := 0;  m(level) := 0;  m(nbNoeud) := 0;  Tant que cm(cour) != NIL faire  Tant que cm(cm(cour)+1) != NIL faire  EMPILER(pile, cm(cour));  m(cour) := cm(cm(cour)+1);  m(level) := cm(level)+1;  m(nbNoeud) := cm(nbNoeud)+1;  fait;  Si cm(level) >= cm(hauteur) alors  m(hauteur) := cm(level)+1;  fsi;  m(cour) := cm(cm(cour)+2);  Tant que cm(cour) = NIL ET non(vide(pile)) faire  m(cour) := DEPILER(pile);  m(cour) := cm(cm(cour)+2);  m(level) := cm(level)-1;  fait;  fait;  LIBERER(pile); |
| FIN |

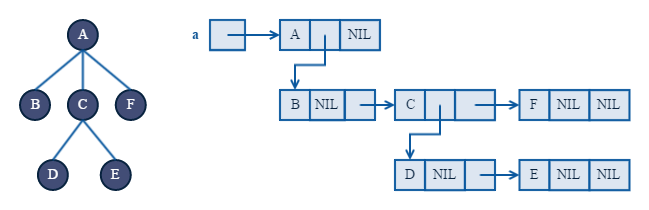
# Lexique

* pile : adresse du pointeur sur une structure pile pour le parcours de l'arbre
* arb : adresse du pointeur sur l'arbre
* taille : entier représentant le nombre d'éléments dans l'arbre
* nbNoeud : adresse de l'entier représentant le nombre de nœuds (racines et nœuds internes sans les feuilles)
* hauteur : adresse de l'entier représentant le nombre maximal de niveaux dans l'arbre
* level : adresse de l'entier pour le calcul de la hauteur
* cour : adresse du pointeur sur le nœud courant de l'arbre

Trace de l’algorithme

1. Cas général

Dans ce cas, nous disposons d'un arbre général avec une unique racine comme le stipule l’énoncé. Le lien horizontal de la racine est une liste vide. Sa représentation en mémoire est donc la suivante :



On définit l’adresse des points dans l’arbre ainsi : le point de valeur A est à l’adresse cm(a), le point de valeur B à l’adresse b, celui de valeur C à l’adresse c, celui de valeur D à l’adresse d et le dernier à l’adresse e.

Exécutons HAUTEUR\_NBNOEUDS(a, 5, nœud, max)

Initialisation de la pile à une taille de 5

m(cour) ≔ cm(a)

m(max) ≔ 0

m(level) ≔ 0

m(nœud) ≔ 0

cm(cour) ≠ NIL = **Vrai**

On **rentre** dans le **Tant que** principal

cm(cm(cour)+1) = b ≠ NIL = **Vrai**, on entre dans le second **Tant que**

on empile cm(a) dans la pile

m(cour) := cm(cm(a)+1) = b pour simplification

m(level) := 0 + 1 =1

m(nœud) := 0 + 1 = 1

cm(b+1) ≠ NIL = **Faux**, on sort du second **Tant que**

cm(level) = 1 >= cm(max) = 0 = **Vrai**, on **entre** dans le **Si**

m(max) ≔ cm(level) + 1 = 2

m(cour) := cm(cm(cour)+2) = cm(b+2) = c

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Faux ;** on n’**entre** **pas** dans le **Tant que**

cm(cour) = c ≠ NIL = **Vrai ;** On **entre à nouveau** dans le **Tant que principal**

cm(c+1) = d ≠ NIL = **Vrai**, on **entre** dans le second **Tant que**

on empile c dans la pile

m(cour) := cm(c+1) = d pour simplification

m(level) := 1 + 1 =2

m(nœud) := 1 + 1 = 2

cm(d+1) ≠ NIL = **Faux**, on sort du second **Tant que**

cm(level) = 2 >= cm(max) = 2 = **Vrai**, on **entre** dans le **Si**

m(max) ≔ cm(level) + 1 = 3

m(cour) := cm(cm(cour)+2) = cm(d+2) = e

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Faux ;** on n’entre pas dans le Tant que

cm(cour) = e ≠ NIL = **Vrai ;** On **entre à nouveau** dans le **Tant que principal**

cm(e+1) ≠ NIL = **Faux**, on n’entre pas dans le second **Tant que**

cm(level) = 2 >= cm(max) = 3 = **Faux**, on n’**entre pas** dans le **Si**

m(cour) := cm(cm(cour)+2) = cm(e+2) = NIL

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Vrai ;** on entre dans le dernier **Tant que**

m(cour) := depiler(pile) = c

m(cour) := cm(c+2) = f

m(level) := cm(level) – 1 = 2 – 1 = 1

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Faux ;** on sort du dernier **Tant que**

cm(cour) = f ≠ NIL = **Vrai ;** On **entre à nouveau** dans le **Tant que principal**

cm(f+1) ≠ NIL = **Faux**, on n’entre pas dans le second **Tant que**

cm(level) = 1 >= cm(max) = 3 = **Faux**, on n’**entre pas** dans le **Si**

m(cour) := cm(cm(cour)+2) = cm(f+2) = NIL

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Vrai ;** on entre dans le dernier **Tant que**

m(cour) := depiler(pile) = cm(a)

m(cour) := cm(cm(a)+2) = NIL

m(level) := cm(level) – 1 = 1 – 1 = 0

non(vide(pile)) = **Faux ;** cm(cour) = NIL = **Vrai ;** on sort du dernier **Tant que**

cm(cour) ≠ NIL = **Faux ;** On **sort** du **Tant que principal**

On libère la pile.

L’algorithme se termine.

L'algorithme donne donc :

**Hauteur** de l'arbre : **3**

**Nombre** de nœuds : **2**

1. Arbre vide

On dispose ici d'un arbre vide, c’est-à-dire que sa représentation en mémoire est la suivante :



Exécutons HAUTEUR\_NBNOEUDS(a, 0, nombre, hauteur)

Initialisation de la pile à une taille nulle (0) (la pile est inutile)

m(cour) ≔ cm(a) = NIL

m(hauteur) ≔ 0

m(level) ≔ 0

m(nombre) ≔ 0

cm(cour) ≠ NIL = **Faux**

On ne **rentre** **pas** dans le **Tant que** principal.

Libération de la pile

L’algorithme se termine.

L'algorithme donne donc :

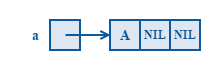
**Hauteur** de l'arbre : **0**

**Nombre** de nœuds : **0**

Ceci est cohérent pour un arbre vide.

1. Arbre avec seulement une feuille

Dans ce cas, nous disposons d'un arbre composé uniquement d'une racine. Le lien horizontal de la racine est une liste vide tout comme le lien vertical. Sa représentation en mémoire est donc la suivante :



Exécutons HAUTEUR\_NBNOEUDS(a, 1, nœud, max)

Initialisation de la pile à une taille de 1

m(cour) ≔ cm(a)

m(max) ≔ 0

m(level) ≔ 0

m(nœud) ≔ 0

cm(cour) ≠ NIL = **Vrai**

On **rentre** dans le **Tant que** principal

cm(cm(cour)+1) ≠ NIL = **Faux**, on ne rentre pas dans le second **Tant que**

cm(level) = 0 >= cm(max) = 0 = **Vrai**, on **entre** dans le **Si**

m(max) ≔ cm(level) + 1 = 1

m(cour) := cm(cm(cour)+2)

non(vide(pile)) = **Faux**

cm(cour) = NIL = **Vrai**

On n’entre pas dans le Tant que

cm(cour) ≠ NIL = **Faux**

On **sort** du **Tant que principal**

On libère la pile.

L’algorithme se termine.

L'algorithme donne donc :

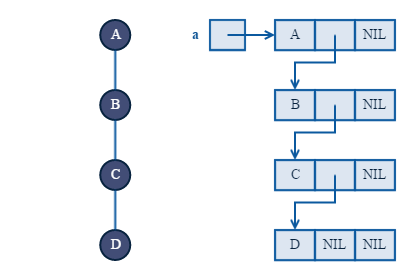
**Hauteur** de l'arbre : **1**

**Nombre** de nœuds : **0**

La racine à elle seule n'est pas un nœud et constitue un seul niveau d'arbre.

1. Arbre avec seulement des liens verticaux

Dans ce cas, nous disposons d'un arbre composé uniquement de liens verticaux. Les liens horizontaux sont donc tous des listes vides. Sa représentation en mémoire est donc la suivante :



On définit l’adresse des points dans l’arbre ainsi : le point de valeur A est à l’adresse cm(a), le point de valeur B à l’adresse b, celui de valeur C à l’adresse c et le dernier à l’adresse d.

Exécutons HAUTEUR\_NBNOEUDS(a, 4, nœud, max)

Initialisation de la pile à une taille de 4

m(cour) ≔ cm(a)

m(max) ≔ 0

m(level) ≔ 0

m(nœud) ≔ 0

cm(cour) ≠ NIL = **Vrai**

On **rentre** dans le **Tant que** principal

cm(cm(cour)+1) = b ≠ NIL = **Vrai**, on entre dans le second **Tant que**

on empile cm(a) dans la pile

m(cour) := cm(cm(cour)+1) = cm(cm(a)+1) = b pour simplification

m(level) := 0 + 1 = 1

m(nœud) := 0 + 1 = 1

cm(cm(cour)+1) = c ≠ NIL = **Vrai**, on rentre à nouveau dans le second **Tant que**

on empile b dans la pile

m(cour) := cm(cm(cour)+1) = cm(b+1) = c pour simplification

m(level) := 1 + 1 = 2

m(nœud) := 1 + 1 = 2

cm(cm(cour)+1) = d ≠ NIL = **Vrai**, on rentre à nouveau dans le second **Tant que**

on empile c dans la pile

m(cour) := cm(cm(cour)+1) = cm(c+1) = d pour simplification

m(level) := 2 + 1 = 3

m(nœud) := 2 + 1 = 3

cm(cm(cour)+1) ≠ NIL = **Faux**, on sort du second **Tant que**

cm(level) = 3 >= cm(max) = 0 = **Vrai**, on **entre** dans le **Si**

m(max) ≔ cm(level) + 1 = 4

m(cour) := cm(cm(cour)+2) = NIL

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Vrai ;** on entre dans le dernier **Tant que**

m(cour) := depiler(pile) = c

m(cour) := cm(c+2) = NIL

m(level) := cm(level) – 1 = 3 – 1 = 2

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Vrai ;** on rentre dans le dernier **Tant que**

m(cour) := depiler(pile) = b

m(cour) := cm(b+2) = NIL

m(level) := cm(level) – 1 = 2 – 1 = 1

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Vrai ;** on rentre dans le dernier **Tant que**

m(cour) := depiler(pile) = cm(a)

m(cour) := cm(cm(a)+2) = NIL

m(level) := cm(level) – 1 = 1 – 1 = 0

non(vide(pile)) = **Faux ;** cm(cour) = NIL = **Vrai ;** on sort du dernier **Tant que**

cm(cour) ≠ NIL = **Faux**

On **sort** du **Tant que principal**

On libère la pile.

L’algorithme se termine.

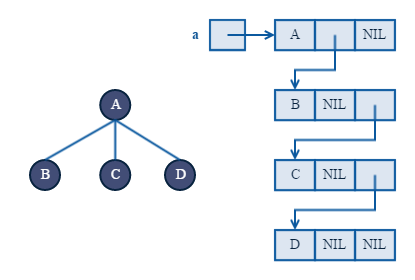
L'algorithme donne donc :

**Hauteur** de l'arbre : **4**

**Nombre** de nœuds : **3**

1. Arbre avec seulement un nœud

Dans ce cas, nous disposons d'un arbre composé uniquement d‘un seul nœud. Le lien horizontal de la racine est une liste vide. Sa représentation en mémoire est donc la suivante :



On définit l’adresse des points dans l’arbre ainsi : le point de valeur A est à l’adresse cm(a), le point de valeur B à l’adresse b, celui de valeur C à l’adresse c et le dernier à l’adresse d.

Exécutons HAUTEUR\_NBNOEUDS(a, 4, nœud, max)

Initialisation de la pile à une taille de 4

m(cour) ≔ cm(a)

m(max) ≔ 0

m(level) ≔ 0

m(nœud) ≔ 0

cm(cour) ≠ NIL = **Vrai**

On **rentre** dans le **Tant que** principal

cm(cm(cour)+1) = b ≠ NIL = **Vrai**, on entre dans le second **Tant que**

on empile cm(a) dans la pile

m(cour) := cm(cm(cour)+1) = cm(cm(a)+1) = b pour simplification

m(level) := 0 + 1 = 1

m(nœud) := 0 + 1 = 1

cm(cm(cour)+1) ≠ NIL = **Faux**, on sort du second **Tant que**

cm(level) = 1 >= cm(max) = 0 = **Vrai**, on **entre** dans le **Si**

m(max) ≔ cm(level) + 1 = 2

m(cour) := cm(cm(cour)+2) = c

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Faux ;** on n’entre pas dans le dernier **Tant que**

cm(cour) = c ≠ NIL = **Vrai**

On **rentre encore** dans le **Tant que principal**

cm(c+1) ≠ NIL = **Faux**, on n’entre pas dans le second **Tant que**

cm(level) = 1 >= cm(max) = 2 = **Faux**, on n’**entre pas** dans le **Si**

m(cour) := cm(cm(cour)+2) = d

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Faux ;** on n’entre pas dans le dernier **Tant que**

cm(cour) = d ≠ NIL = **Vrai**

On **rentre encore** dans le **Tant que principal**

cm(d+1) ≠ NIL = **Faux**, on n’entre pas dans le second **Tant que**

cm(level) = 1 >= cm(max) = 2 = **Faux**, on n’**entre pas** dans le **Si**

m(cour) := cm(cm(cour)+2) = NIL

non(vide(pile)) = **Vrai ;** cm(cour) = NIL = **Vrai ;** on entre dans le dernier **Tant que**

m(cour) := depiler(pile) = cm(a)

m(cour) := cm(cm(a)+2) = NIL

m(level) := cm(level) – 1 = 1 – 1 = 0

non(vide(pile)) = **Faux ;** cm(cour) = NIL = **Vrai ;** on sort du dernier **Tant que**

cm(cour) ≠ NIL = **Faux**

On **sort** du **Tant que principal**

On libère la pile.

L’algorithme se termine.

L'algorithme donne donc :

**Hauteur** de l'arbre : **2**

**Nombre** de nœuds : **1**