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

|  |
| --- |
|  |

Table des matières

[Introduction 2](#_Toc418192996)

[Rappel du sujet 2](#_Toc418192997)

[Présentation du principe 2](#_Toc418192998)

[Schéma explicatif 2](#_Toc418192999)

[Algorithmes 3](#_Toc418193000)

[Principe 3](#_Toc418193001)

[Algorithme 3](#_Toc418193002)

[Lexique 3](#_Toc418193003)

[Trace de l’algorithme 4](#_Toc418193004)

[Cas 1 - Cas général 4](#_Toc418193005)

Introduction

# Rappel du sujet

Nous disposons d'un arbre à une racine implantée par lien vertical et 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 ici présenté à pour but de retourner la hauteur de l'arbre (c’est-à-dire le nombre maximum de niveaux) ainsi que le nombre de nœuds.

Un nœud sera ici considéré comme un élément ayant au moins un successeur.

# 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 NIL alors on a un nœud que l'on va compter.

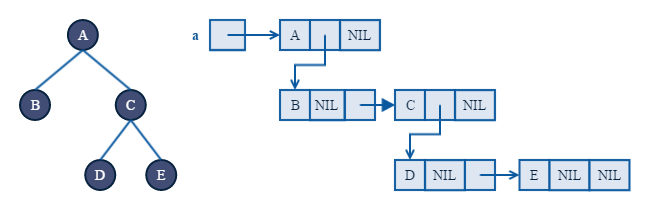
Lorsque l'on a terminé de parcourir le lien vertical, on compare notre niveau d'arbre actuel avec un maximum (qui sera initialisé à 0). Si cette valeur est supérieure au maximum alors le maximum change.

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

# Algorithme

# Lexique

Trace de l’algorithme

1. Cas général
2. Arbre vide

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



Exécutons hauteur\_nbnoeud(a, 10)

Initialisation de la pile

m(cour) ≔ cm(a) = NIL

m(max) ≔ 0

m(level) ≔ 0

m(nœud) ≔ 0

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

cm(cour) ≠ NIL = **Faux**

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

L'algorithme renvoie donc :

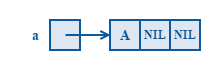
**Hauteur** de l'arbre : **0**

**Nombre** de nœuds : **0**

Ceci est cohérent pour un arbre vide.

1. Arbre avec seulement une racine

Dans ce cas, nous disposons d'un arbre composé uniquement d'une racine. Sa représentation en mémoire est donc la suivante.



Exécutons hauteur\_nbnoeud(a, 10)

Initialisation de la pile

m(cour) ≔ cm(a)

m(max) ≔ 0

m(level) ≔ 0

m(nœud) ≔ 0

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

cm(cour) ≠ NIL = **Vrai**

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

cm(cour) ≠ NIL = **Vrai**, on rentre dans le second **Tant que**

On empile cm(cour)

m(cour) ≔ cm(cm(cour) + 1) = NIL

cm(cour) ≠ NIL = **Faux**

On ne **rentre** **pas** dans le **Si**

m(level) ≔ cm(level) + 1 = 0 + 1 = 1

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

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

m(max) ≔ cm(level) = 1

non(vide(pile)) = **Vrai**

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

On entre dans le Tant que

On dépile dans cour

m(cour) ≔ cm(cm(cour) + 2) = NIL

m(level) ≔ cm(level) – 1 = 1 – 1 = 0

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

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

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

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

cm(cour) ≠ NIL = **Faux**

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

L'algorithme renvoie 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