

Graphe ch3 Arbre

Pierre-Yves BISCHOFF

IUT Informatique Graphique

2020

Sommaire

- 
- 1 Définition
 - 2 Parcours des arbres
 - 3 Arbre binaire

Sommaire de Définition

1 Définition

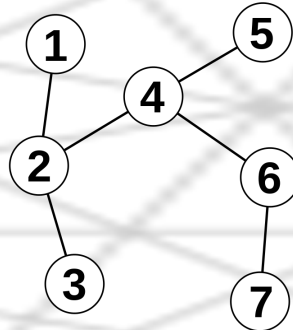
Arbre

arbre

Un graphe **non-orienté** $G = (S, A)$ est un arbre s'il est connexe sans cycle

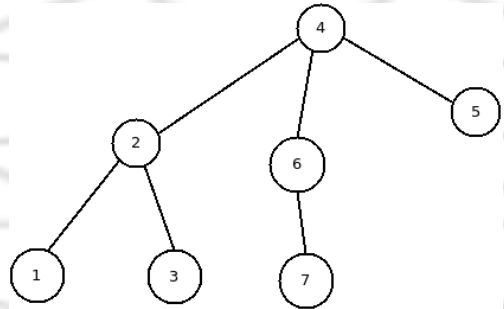
relation entre arête et sommet

le nombre de sommets excède le nombre d'arêtes d'une unité exactement



Enraciner

- ▶ Si on choisit un sommet r quelconque dans un arbre
- ▶ Enraciner l'arbre en r c'est :
- ▶ orienter toutes les arêtes de sorte que :
- ▶ il existe un chemin de r à tous les autres nœuds.



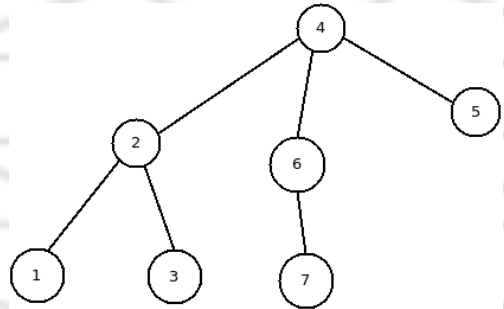
Feuilles

Feuilles

Dans un arbre les nœuds de degré 1 sont des feuilles

sommet interne

les autres sont des sommets internes



Parenté

Enfant

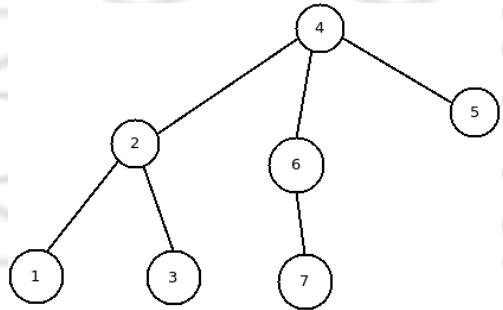
Les successeurs d'un sommet sont ses enfants

Frère

Les enfants d'un même sommet sont des frères

Parent

Le prédécesseur d'un sommet est un parent



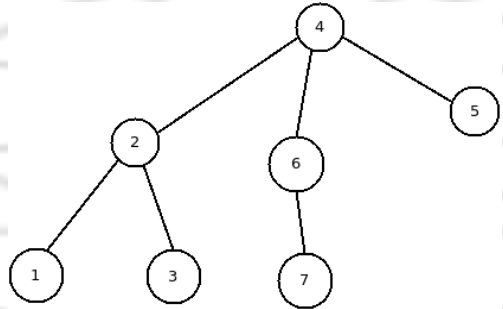
Hauteur ou profondeur

Profondeur d'un sommet

La profondeur d'un sommet s est le nombre de sommets du chemin rs sans compter le sommet de départ

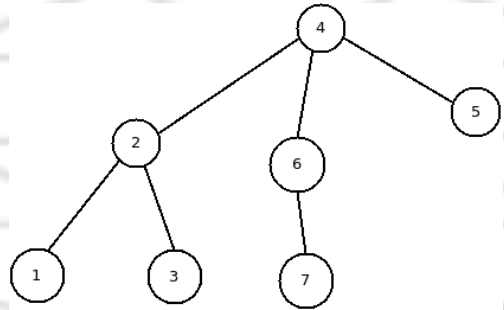
Profondeur d'un arbre

C'est le maximum des profondeurs des sommets de l'arbre




Hauteur ou profondeur

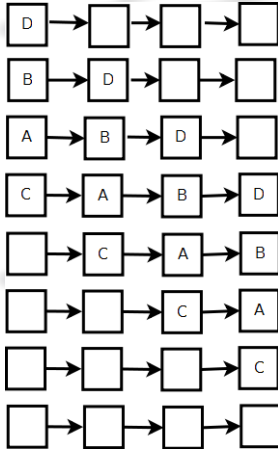
- ▶ ici 4 est à la profondeur 0
- ▶ 2, 6, 5 sont à la profondeur 1
- ▶ 1, 3 et 7 sont à la profondeur 2



Sommaire de Parcours des arbres

- 
- 2 Parcours des arbres
 - File
 - parcours en largeur
 - Pile
 - parcours en profondeur

File : définition



File = FIFO

Une file représente le même comportement qu'une file d'attente

FIFO = First In First Out

file = structure de données

basée sur le principe du premier entré, premier sorti

File : programmation

- ▶ Stockée dans une liste chaînée ou un tableau
- ▶ fonction *enqueue(s)* qui ajoute un élément à queue de la file
- ▶ fonction *dequeue()* qui enlève la tête de file
- ▶ fonction *nbElement()* qui renvoie le nombre d'éléments
- ▶ fonction *isEmpty()* qui renvoie vrai si file vide

Parcours d'un arbre

Parcours

Il existe 2 types de parcours

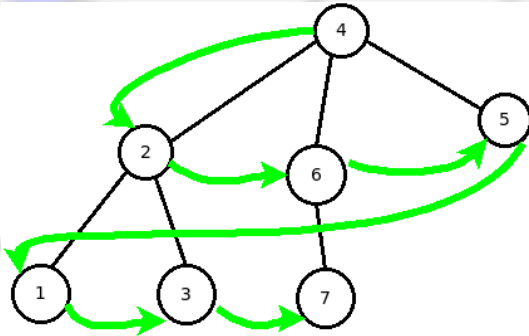
Parcours en largeur

le parcours en largeur va explorer les frères avant les enfants

Parcours en profondeur

le parcours en profondeur va explorer les enfants avant les frères

Parcours d'un arbre en largeur



Utilisation d'une file

On va enfiler les enfants de la racine

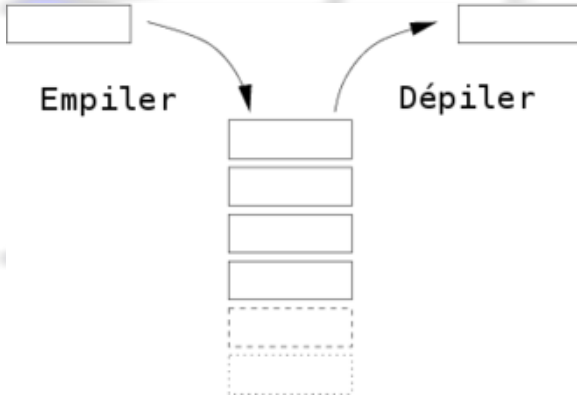
Boucle tant que la file n'est pas vide

on va défiler un élément et enfiler les enfants de cet élément

Ordre de visite

4, 2, 6, 5, 1, 3, 7

Pile : définition



Pile = LIFO

Une pile représente le même comportement qu'une pile d'objets
LIFO = Last In First Out

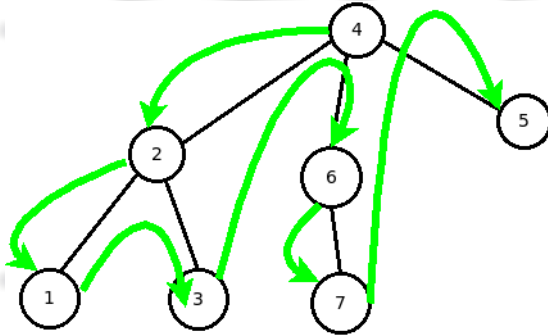
pile = structure de données

basée sur le principe du dernier entré, premier sorti

Pile : programmation

- ▶ Stockée dans une liste chaînée ou avec seulement 1 pointeur
- ▶ fonction *enpile(s)* qui ajoute un élément
- ▶ fonction *depile()* qui enlève un élément
- ▶ fonction *nbElement()* qui renvoie le nombre d'éléments
- ▶ fonction *estVide()* qui renvoie vrai si pile vide

Parcours d'un arbre en profondeur



- ▶ Utilisation d'une pile
- ▶ *empiler(racine)*
- ▶ *marquer(racine)*
- ▶ tant que la pile n'est pas vide
 - *s = depiler()*
 - Pour tout voisin *t* de *s*
 - Si *t* n'est pas marqué
 - *empiler(t)*
 - *marquer(t)*

Ordre de visite

4, 2, 1, 3, 6, 7, 5

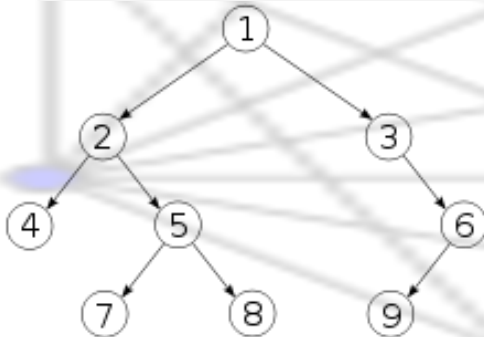
Sommaire de Arbre binaire

- 3 Arbre binaire
 - Définitions
 - Tas
 - Arbre Binaire de Recherche

Définition

Arbre binaire

Dans un arbre binaire les nœuds possèdent au maximum 2 fils

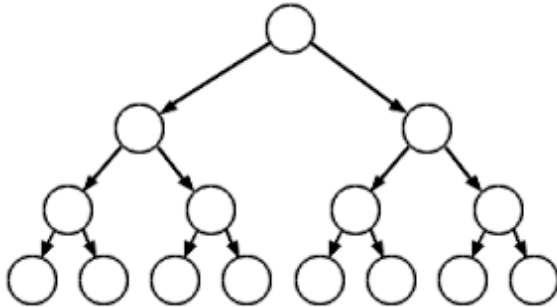


- ▶ On appelle Gauche et Droite les 2 fils d'un nœud
- ▶ À la profondeur 0 il y a le nœud racine
- ▶ À la profondeur n il y a au plus 2^n nœud
- ▶ Un nœud sans fils est une feuille

Définition

Arbre binaire parfait

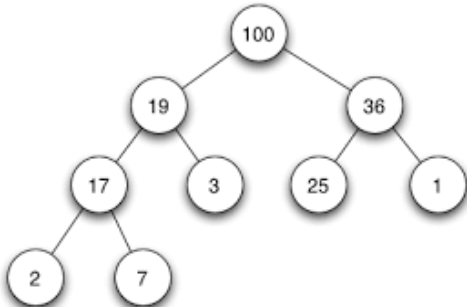
Arbre binaire dont toutes les feuilles sont à la même profondeur



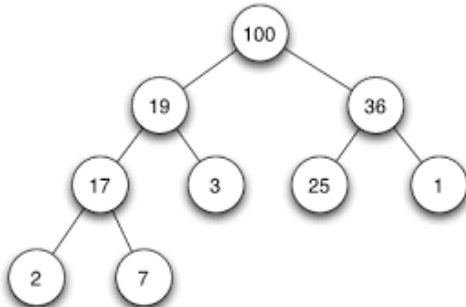
Définition

Arbre binaire complet

Arbre binaire de profondeur d dont les $d - 1$ premiers niveaux sont complets et les feuilles du dernier sont à gauche



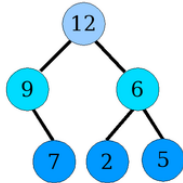
Tas binaire



- ▶ un tas est un arbre binaire complet
- ▶ la clé de chaque nœud \geq (resp. \leq) aux clés de ses 2 fils
- ▶ C'est un tas-max (resp. tas-min)

Exemple d'arbres qui ne sont pas des tas

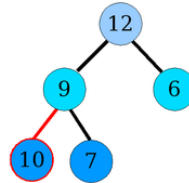
Cet arbre n'est pas un tas, car il n'est pas complet calé à gauche :



La représentation en tableau comporte un « trou », ce qui est interdit :

0	1	2	3	4	5	6
12	9	6	-	7	2	5

Cet arbre n'est pas un tas, car il viole la propriété de tas : un noeud de valeur 10 ne devrait pas être fils d'un noeud de valeur 9.



Représentation en tableau :
 $\text{valeur}(\text{filsgauche}(1)) > \text{valeur}(1)$

0	1	2	3	4
12	9	6	10	7



département
informatique graphique

Propriétés des tas

Propriété

Pour tous les nœuds s et t de l'arbre tels que t est un fils de s Alors

$$\text{valeur}(t) \leq \text{valeur}(s)$$

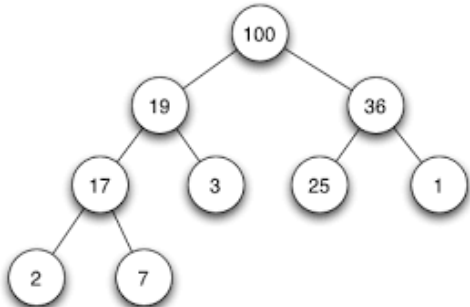
Conséquences on a $T = (G, D, r)$

- ▶ $\max(T) = r$
- ▶ le plus grand élément du tas est à la racine
- ▶ Permet une insertion d'un élément en temps logarithmique
- ▶ Accès direct au maximum



département
informatique graphique

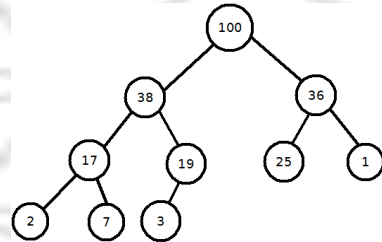
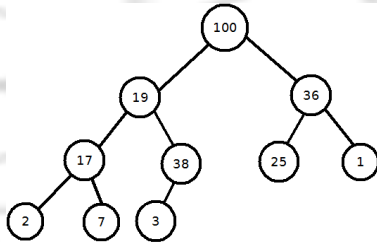
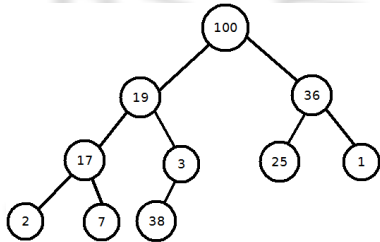
Algorithme d'insertion dans un tas



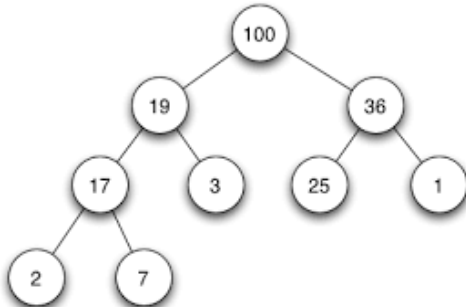
Insertion de l'élément 38

- ▶ On insère 38 à la première place libre
- ▶ Puis si $38 > \text{pere}(38)$ on échange 38 et son père
- ▶ Et ainsi de suite
- ▶ Remarque : au pire on remonte jusqu'à la racine et on fait $\log(n)$ échanges

Déroulement de l'algorithme d'insertion



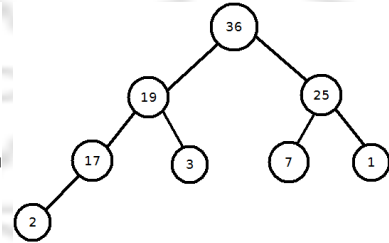
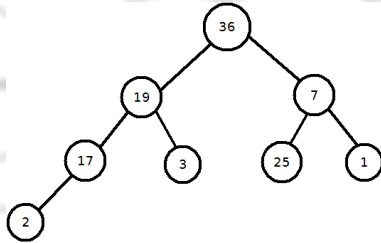
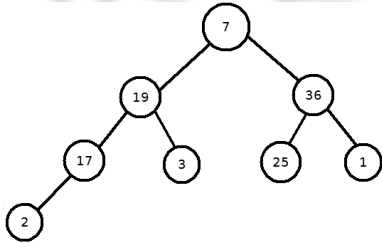
Algorithme pour retirer la racine dans un tas



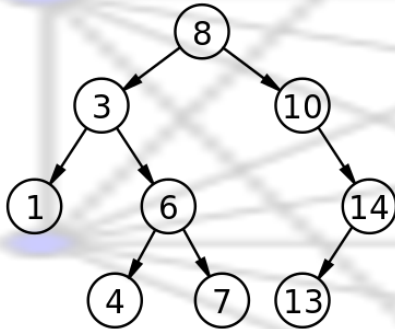
Suppression de l'élément 100

- ▶ On remplace 100 par le dernier nœud
- ▶ Puis on échange cet élément avec son plus grand fils
- ▶ Et ainsi de suite
- ▶ Remarque : au pire on descend jusqu'à une feuille et on fait $\log(n)$ échanges

Déroulement de suppression de la racine

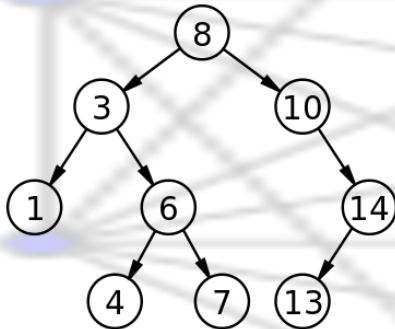


ABR



- ▶ un ABR est un arbre binaire
- ▶ pour tous les nœuds :
- ▶ les clés du sous arbre gauche \leq à sa clé
- ▶ les clés du sous arbre droit \geq à sa clé

Algorithme de recherche dans un ABR



On recherche la clé 7

- ▶ algorithme récursif $7 \neq 8$ et $7 < 8$
- ▶ Donc on recherche 7 dans le fils gauche de 8
- ▶ $7 \neq 3$ et $7 > 3$ donc on recherche dans fils droit de 7
- ▶ etc ... si filsG ou filsD vide retourne faux

performance algorithme de recherche dans un ABR

performance moyenne

Cette opération requiert un temps en $O(\log(n))$ si l'arbre est équilibré

pire performance

Par contre si l'arbre est dégénéré en une liste chaînée : $O(n)$

Rééquilibrage d'arbre binaire

- ▶ Pour garder l'arbre équilibré on peut faire une rotation
- ▶ Rotation à droite
- ▶ Rotation à gauche
- ▶ Cela va permettre de réduire la hauteur d'un côté et donc ajouter de l'autre

Rotation dans un arbre

p, q sont des sommets, U, V, W des sous-ABR

