Cours de Programmation C

Mendy Fatnassi

10 décembre 2020

Table des matières

Chapitre 1 : Arbre Binaire,N-aire,ARN,AVL

1.1 Generalité sur les Arbres

En informatique un arbre est un ensemble de sommets organisé de facons hiéarchique tel qu'il existe un unique sommet appelé racine et qui n'a donc pas de superieure (père).
Tous les autres sommets sont atteints a partir de la racine par un unique chemins .

De facons recursive (et constructive) un arbre est la donnée d'une racine et d'une liste d'arbres disjoints appelés sous-arbres.
Un sommet de l'arbre est la racine d'un sous-arbre.

1.1.1 Principe

Les arbres sont pratique pour la mise en place d'une structure de parcoure. Nous connaisons les structure de donnéé linéaire tels que les pile,file et liste maintenant nous allons voir des structure un peux plus complexe comme les arbres et les graphes (Cf.Cours$_graphe$).

Lesarbres, commeleslistes, permettentdereprsenterunnombrevariablededonnes.Leprincipalavantag

1.1.2 Définition

-Un arbre est soit un noeud, soit un arbre vide.

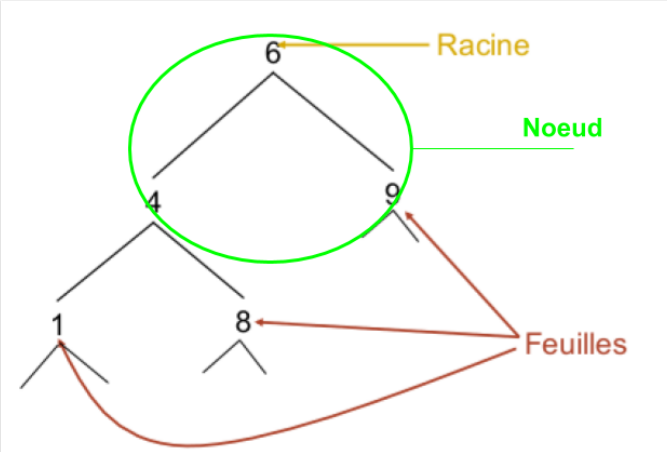
-Un noeud a des fils qui sont eux aussi des arbres.

-Si tous les fils d'un noeud sont vides, alors le noeud est qualifié de feuille.

-Les noeuds portent des valeurs, ce sont les données que l'on veut stocker.

-Si tous les nœuds de l'arbre ont n fils, alors l'arbre est dit n-aire.

Exemple :



Un arbre d'un point de vie informatique et une structure de recherche ayant une valeur et 2 sous arbres (Gauche-Droit), mais en theorie des graphe,un arbre est un graphe non orienté connexe et sans cycle.

Terminologie

Terminologie généalogique :

-**père** : Prédécesseur d'un sommet.
-**fils** : Successeur d'un sommet.
-**frères** : Sommets qui ont le meme père.
On peut y ajouter les notions de (grand-père,descendants,etc.).

Terminologie forestière :

-**racine** : Noeud qui ne possede pas de père.
-**feuilles** : Noeud (externes ou terminaux) qui n'ont pas de fils.
-**branches** : Généralement un chemin depuis un noeud racine (quelque fois jusqu'a une feuille).

1.1.3 Mesures sur les Arbres

La mesures sur les arbres sont tres utiles pour connaitre certainne caractéristique de l'arbre et permet aussi la mesure de la complexité des algorithmes sur les parcours d'arbres par exemple.

-**Profondeur**(Niveau) d'un noeud : Correspond a la longueur de la branche qui la joint a la racine (nombre d'arcs).

-**Hauteur** d'un noeud : C'est la longueur de la plus longue branche de ce noeud jusqu'a une feuille.

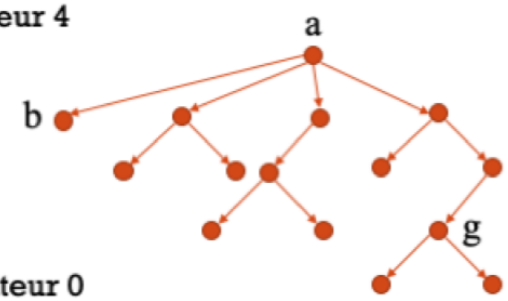
-**Hauteur** d'un arbre : C'est la longueur de la branche la plus longue (Hauteur de la racine).

-**Taille** d'un arbre : Nombre de ses sommets.

-**Longueur** totale : La somme des longueurs des branches issues de la racine.

Voci un petit exemple :

- Arbre de Hauteur 4
- de taille 15



- b niveau 1 hauteur 0
- a niveau 0 hauteur 4
- g niveau 3 hauteur 1

Chapitre 2 : Parcours d'arbre

Prefixe/Infixe/Postfixe
Il s'agit de parcours en longueur , on va en voir 3 type :

Parcours Prefixe (Racine Gauche Droit) : le noeud racine est traité au premier passage avant le parcours des sous-arbres .

Parcours Infixe (ou Symetrique) (Gauche Racine Droit) : On commenca par examiner le noeud gauche puis la racine et on finipar sson noeud droit.

Parcours Postfixe (Gauche Droit Racine) : Meme principe.

2.0.1 Largeur

Le parcours en largeur , consiste a traiter chaque noeud de gauche a droite sur chque ligne de l'arbre.

Chapitre 3 : Arbre Binaire

-Arbre binaire complet : Chaque noeud a 0 ou 2 fils .
-Arbre binaire parfait : Arbre complet + Toutes les feuilles sont a la meme hauteur.
-Arbre binaire quasi-parfait : Arbre parfait sauf qu'il peux manqué des feuilles a droite.

Un arbre binaire est :

-soit l'arbre vide
-soit un noeud qui a exactement deux fils (éventuellement vides)

Pour manipuler les arbres binaires, on a besoin de primitives d'**accès** :
valeur : retourne la valeur d'un arbre non vide
fils-g : retourne le fils de gauche d'un arbre non vide
fils-d : retourne le fils de droite d'un arbre non vide

de primitives de **test** :
vide ? : retourne vrai si un arbre donné est vide
arbre= ? : retourne vrai si deux arbres donnés sont égaux

de primitives de **construction** :
creerArbre : retourne un arbre vide
cons-binaire : crée un arbre avec une valeur donnée et deux arbres donnés qui seront ses deux uniques fils .

3.1 Implementation par tas

Implementation par tas d'un arbre.

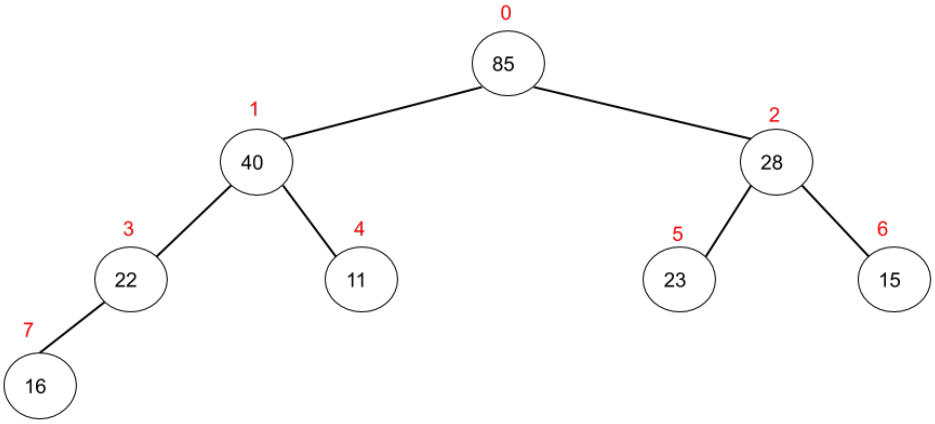
typedef enum tas$_type MIN, MAX t_t as_t ype$;
typedef struct tas int *tas ; int nb$_v al, nc, max_v al; t_t as_t ypetype; t_t as$;
typedef struct tas * t$_t as$;

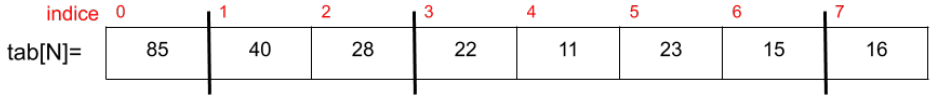
3.1.1 Primitive

/*****Primitive de Creation*****/ $t_tascreer_tas(intnbv)t_tas * t = malloc(sizeof(t_tas)); t- > tas$

-Pour trouvé un fils gauche(fg) ou un fils droite(fd) dans un arbre qui est representer par un tableau on peux appliquer les formule suivante :
fg=2n+1 et fd=2n+2 (ou n est le numeros du noeud a l'indice i)

Exemple :





/*****Primitive de Test*****/

Chapitre 4 : Arbre ARN

Chapitre 5 : Arbre N-aire