

Ch4 - Les piles et les files

~~Introduction:~~

1) La Notion de Pile

on considère une pile d'entiers :

cellule : Enregistrement

val : Entier

suiv : *cellule

Fin Enreg

Pile : Enregistrement

sommet : *cellule

Fin Enreg

Procédure Init (P: *Pile)

debut

P → sommet ← NULL

fin

Fonction Est vide (P: Pile): booléen

debut

Estvide ← (P.sommet = NULL)

fin

Fonction valsommet (P: Pile): entier

debut

valsommet ← P.sommet → val

Procédure Empiler (P: *Pile, x: entier)

var

q: *cellule

debut

q ← Allouer(1)

q → val ← x

q → suiv ← P → sommet

P → sommet ← q

fin

Fonction Depiler (P: *Pile): Entier

var

x: Entier

q: *cellule

debut

x ← P → sommet → val

q ← P → sommet

P → sommet ← P → sommet → suiv

liberer(q)

Depiler ← x

Fin

Ex:

Ecrire une procédure permettant de trier une pile d'entiers dans l'ordre croissant (plus petit est en sommet); en utilisant une pile intermédiaire et une variable simple

Procédure Tri (P: *Pile)

var

Q: Pile

x: entier

Debut

Init(Q)

tantque (Estvide(*P) = faux) faire

x ← Depiler(P)

tantque (Estvide(Q) = faux

Et valsommet(Q) > x) faire

Empiler(P, Depiler(Q))

fin tantque

Empiler(Q, x)

fin tantque

tantque (Estvide(Q) = faux) faire

Empiler(P, Depiler(Q))

fin tantque

fin

Expression Null

- Infixe $A+B$
- Postfixe $AB+$
- Préfixe $+AB$

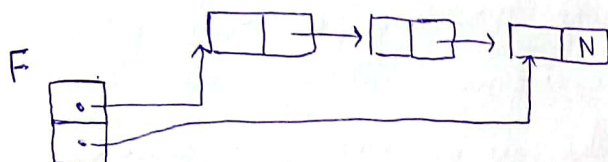
L'algo de conversion Inf - Postf

Inf: $A+B-(C*(2/4*5))\%2*E$



output: $AB+C24/5**2\%E*-$

La notion de File :



Cellule : Enregistrement
 Val : Entier
 suiv : *cellule

Fin enreg

File : Enregistrement
 Tete : *cellule
 Queue : *cellule

Fin enreg

Procédure Init(F ; *File)

Debut

$F \rightarrow Tete \leftarrow NULL$

$F \rightarrow queue \leftarrow NULL$

Fin

Fonction Estvide (F ; File); boolean

debut

$Estvide \leftarrow (F.Tete = NULL)$

fin

Procédure Enfiler(F ; *File, x : entier)

Var

q : *cellule

debut

$q \leftarrow Allouer(1)$

$q \rightarrow val \leftarrow x$

$q \rightarrow suiv \leftarrow NULL$

si ($Estvide(*F) = vrai$) Alors

$F \rightarrow Tete \leftarrow q$

sinon

$F \rightarrow queue \rightarrow suiv \leftarrow q$

fin si

$F \rightarrow queue \leftarrow q$

fin

Fonction defiler (F ; *File) *Entier

Var

x : Entier

q : *cellule

debut

$x \leftarrow F \rightarrow Tete \rightarrow val$

$q \leftarrow F \rightarrow Tete$

$F \rightarrow Tete \leftarrow F \rightarrow Tete \rightarrow suiv$

si ($F \rightarrow Tete = NULL$) Alors

$F \rightarrow queue \leftarrow NULL$

fin si

liberer(q)

defiler $\leftarrow x$

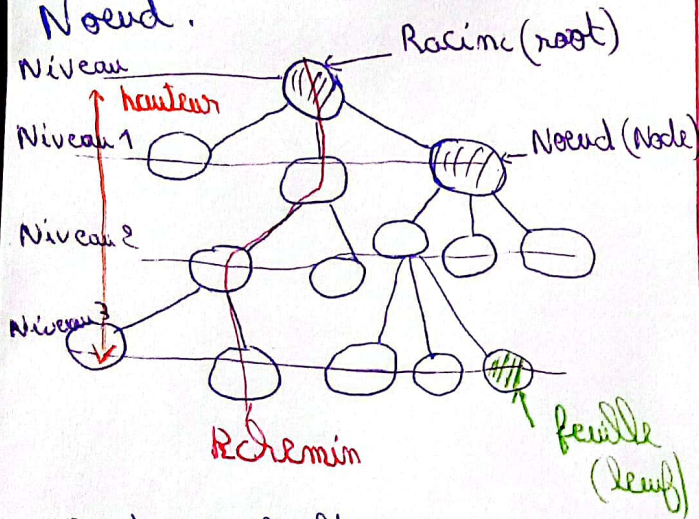
fin

Ch 5 - Les arbres

Def: un arbre est SD permettant de regrouper plusieurs ^{éléments} identiques

Les éléments peuvent être alloués d'une manière non contigue.

Chaque elt possède un seul prédécesseur (père), sauf le premier (Racine) et zéro ou plusieurs successeur (fils). un élément dans un arbre est appelé Noeud.



Poids: 2 val-elt

Noeud: Enregistrement

val: type

AF1: *Noeud

AF2: *Noeud

⋮

AFN: *Noeud

fin enreg

nb max de fils = 2: Arbre binaire
=> un arbre binaire est un arbre dont chaque Noeud peut avoir un maximum 2 fils

1) Arbre binaire:

on considère un binaire d'entiers:

Noeud: Enregistrement

Val: Entier

FG: *Noeud

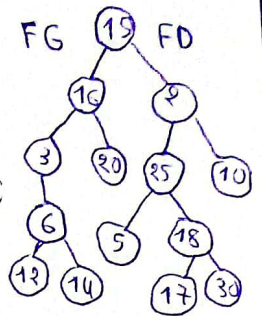
FD: *Noeud

fin Enreg

Arbre: Enregistrement

Racine: *Noeud

fin Enreg



~~INIT~~

Initialiser un arbre vide:

Procedure Init (A: *Arbre)

debut

A ↔ Racine ← NULL

fin

verifier si l'arbre est vide ou non

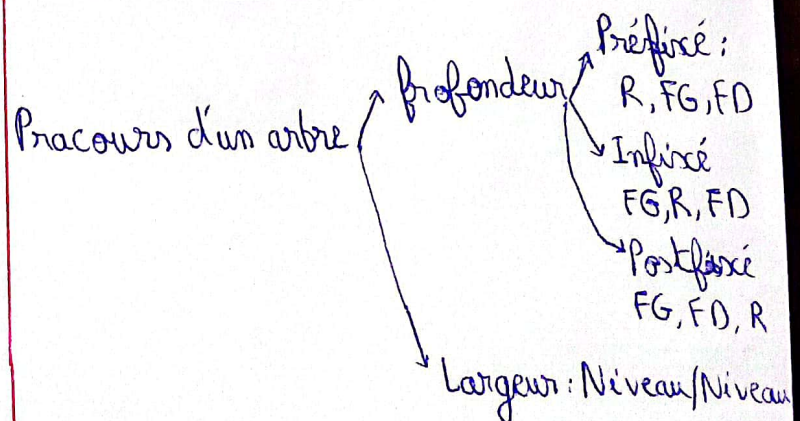
~~fonction~~

fonction Estvide (A: Arbre): booléen

debut

Estvide ← (A.Racine = NULL)

fin



Procédure Préfixe (A: Arbre)

Var

AG, AD: Arbre

debut

si (Est-vidé(A) = faux) Alors

AG.Racine \leftarrow A.Racine \rightarrow FG

AD.Racine \leftarrow A.Racine \rightarrow FD

Ecrire(A.Racine \rightarrow val)

Préfixé(AG)

Préfixé(AD)

fin
fin si

Préfixé: 15-16-3-6-12-14-20-2-25-5
-18-17-30-10

Procédure infixe (A: Arbre)

Var

AG, AD: Arbre

Debut

si (Est-vidé(A) = faux) Alors

AG.Racine \leftarrow A.Racine \rightarrow FG

AD.Racine \leftarrow A.Racine \rightarrow FD

Postfixé

1 | Préfixé(AG)

3 | Ecrire(A.Racine \rightarrow val)

2 | Préfixé(AD)

fin
fin si

Infixe: 3-12-6-14-16-20-15-5-25-
17-18-30-2-10

Postfixe: 12-14-6-3-20-16-5-17-30
-18-25-10-2-15

Procédure Longueur (A: Arbre)

Var

F: File de Noeud

X: Noeud

debut

Si (Est-vidé(A) = faux) Alors

Enfiler(&F, *(A.Racine))

tant que (Est-vidé(F) = faux) faire

X \leftarrow Defiler(&F)

Ecrire(X.val)

si (X.FG \neq NULL) Alors

Enfiler(&F, *(X.FG))

fin si

si (X.FD \neq NULL) Alors

Enfiler(&F, *(X.FD))

fin si
fin tant que

fin

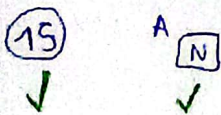
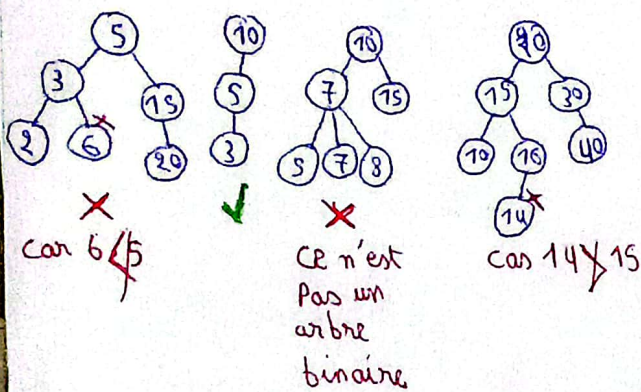
15 16 23 20 25 10 6 5 18 12 14 17 30 10
Sortie: 15 16 23 20 25 10 6 5 18 12 14
17 30

2) Arbre Binaire de Recherche ABR:

def: un ABR est un arbre binaire dont chaque NOEUD doit vérifier la propriété suivante:

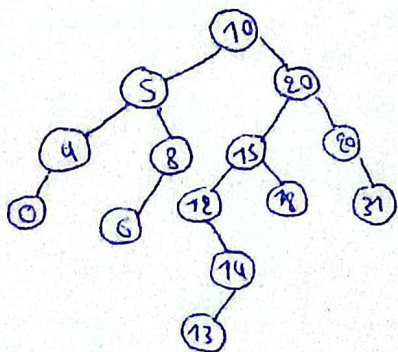
Tout elt $x \in \text{SAG} \leftarrow \text{val de Noeud} (= \text{tout elt} \in \text{SAD})$

Exp. parmi les arbres suivants identifier les ABR:



Les principales opérations:

Ajout d'un elt:



$q \leftarrow \text{Allow}(1)$
 $q \rightarrow \text{val} \leftarrow x$
 $A \rightarrow \text{FG/FD} \leftarrow q$

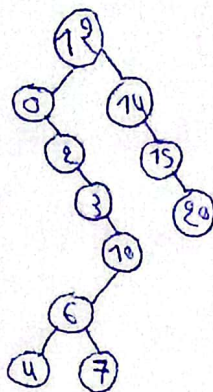
Supp. d'un elt à partir d'un ABR

on distingue 3 cas:

- Noeud à supprimer est une feuille
- Noeud à supprimer possède 1 seul fils (FC/FD)
- Noeud à supprimer possède 2 fils

Ex:

1) construire un ABR avec l'ensemble suivant:
 12 - 14 - 0 - 2 - 3 - 10 - 15 - 6 - 20 - 7 - 4



2) Ajouter l'elt $x = 13$

3) suppr les elts: 2, 20, 12

on considère un AB dont le nb de Noeud = n
 le nb de feuille = f
 l'hauteur = h

1) Quelle est l'hauteur maximale de l'arbre?
 $\Rightarrow h_{\max} = n$

2) Quel est le nb max feuille en fonction de h ? $\Rightarrow f_{\max} = 2^h$