Algorithmique & Structures de données

Les arbres binaires

hepia, HES-SO - ITI

1. Définition d'un arbre

Un arbre est une structure de données qui peut être:

- soit un ensemble vide ;
- soit un ensemble comportant un élément de type T appelé nœud auquel est associé un nombre fini d'arbres appelés sous-arbres, dont les nœuds sont de type T.

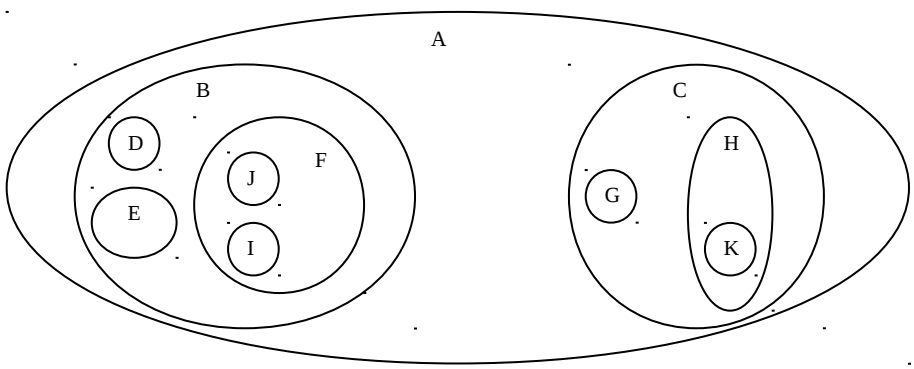
Autres définitions

- Le premier nœud d'un arbre s'appelle la ***racine*** de l'arbre.
- Un nœud directement associé à un autre s'appelle un ***descendant*** direct ou ***fils*** de ce dernier.
- Un nœud sans descendant s'appelle une ***feuille*** ou ***nœud terminal***.
- Le nombre de descendants directs associés à un même nœud s'appelle le ***degré*** de ce nœud. Le degré de l'arbre est le maximum des degrés des nœuds de l'arbre.
- Le ***niveau*** de la racine d'un arbre est 1 (ou 0 selon la manière de compter) ; le niveau du descendant direct d'un nœud est celui de ce nœud plus 1.
- Un arbre ***ordonné*** est un arbre dans lequel l'ordre des sous-arbres de chaque nœud est fixé.
- Un arbre ***binaire*** est un arbre ordonné de degré deux.

2. Les représentations d'un arbre

On peut donner diverses représentations pour schématiser les arbres. Voici quelques représentations externes (ne faisant intervenir aucune structure informatique particulière).

a) Représentation ensembliste



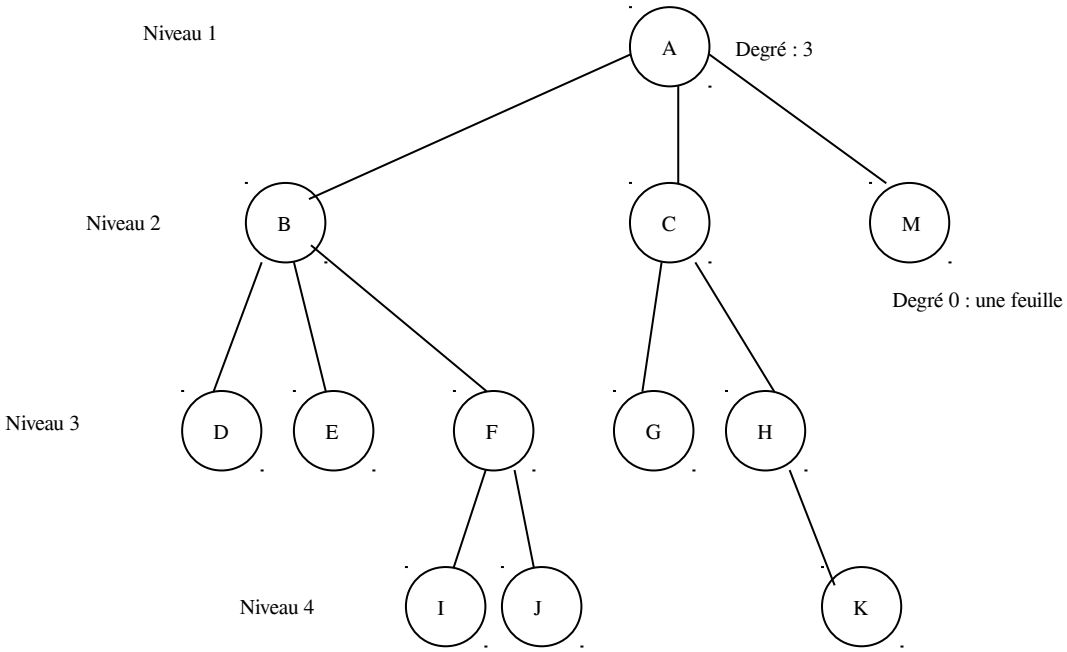
b) Représentation sous forme de liste

```
(A(B(D)(E)(F(I)(J)))(C(G)(H(K))))
```

c) Représentation par niveaux

```
1       2       3       4
A
        B
                D
                E
                F
                        I
                        J
        C
                G
                H
                        K
```

d) Représentation sous forme d'arbre (botanique)



3. La spécification formelle d'un arbre binaire

La spécification formelle d'un arbre binaire définit un minimum d'opérations. En particulier, aucune opération ne permet d'ajouter ou de supprimer un élément d'un arbre sans préciser à quel endroit doit s'effectuer cette adjonction ou cette suppression. Cela dépend de ce que représente l'arbre.

Nous donnerons ici une spécification formelle minimale sachant que dans d'autres présentations, l'interface peut être plus complet. Pour simplifier l'écriture de la spécification formelle de cette structure, nous introduisons les notations suivantes.

Notation

Un arbre `A` est noté : **`<r,A1,A2>`** où:

`r` est la racine de l'arbre **`A`**

`A1` est le sous-arbre gauche de **`A`** (l'arbre est ordonné)

`A2` est le sous-arbre droite de **`A`**

4. La représentation interne d'un arbre binaire

L'arbre est essentiellement (mais pas exclusivement) une structure dynamique. On le définit en fonction des nœuds qui le compose. Un nœud consiste en un article avec un champ pour la donnée stockée dans l'arbre et des champs pour accéder aux sous-arbres associés à ce nœud.

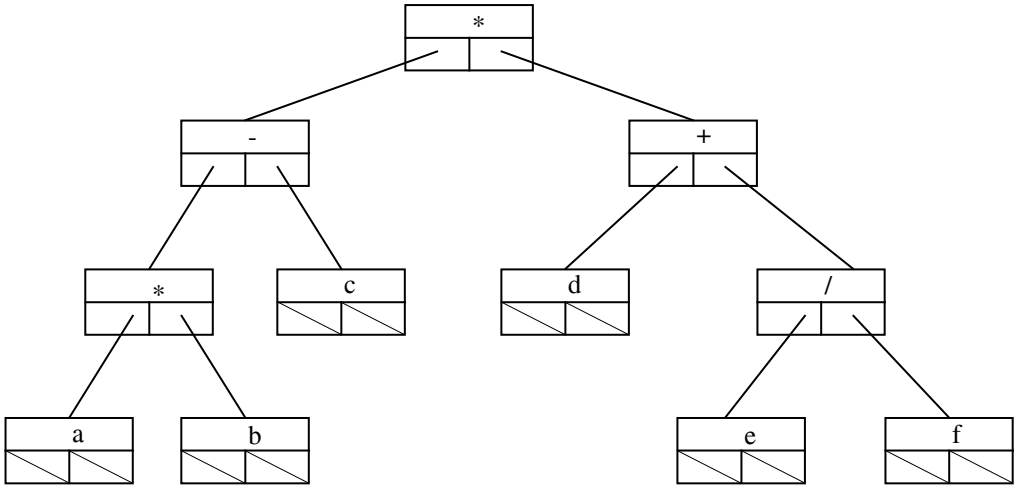
Typiquement, la déclaration d'une structure d'arbre aura l'allure suivante :

```
type T_Donnee is ...    -- ici description de la donnée ou type privé
type T_Noeud;             -- prédéclaration
type T_Arbre is access T_Noeud;
type T_Noeud is record
   Valeur        : T_Donnee;
   Gauche,Droite : T_Arbre;
end record;
```

Il existe d'autres représentations internes qui s'adaptent mieux à des situations particulières.
On peut implanter un arbre avec un tableau par exemple.

Exemple d'implantation d'un arbre

Une expression arithmétique comme **`(a*b-c)*(d+e/f)`**
admet une représentation interne comme :



Remarquez que les nœuds de cet arbre ne contiennent pas tous des informations du même type. Les feuilles contiennent des opérandes et les nœuds internes contiennent des opérateurs. On parle alors d'arbre hétérogène

a) Autre exemple

Si le langage utilisé ne permet pas l'utilisation du type accès, on peut se servir d'un tableau pour représenter un arbre binaire.
Avec l'exemple précédent on obtient la table suivante:

1	*	2	3
2	-	4	5
3	+	6	7
4	*	8	9
5	c	0	0
6	d	0	0
7	/	10	11
8	a	0	0
9	b	0	0
10	e	0	0
11	f	0	0

Comme l'arbre est ici hétérogène, il faut stocker des informations de types différents dans chaque ligne ou si ce n'est pas possible, utiliser un codage approprié.

5. Les traitements

a) Les parcours d'un arbre binaire

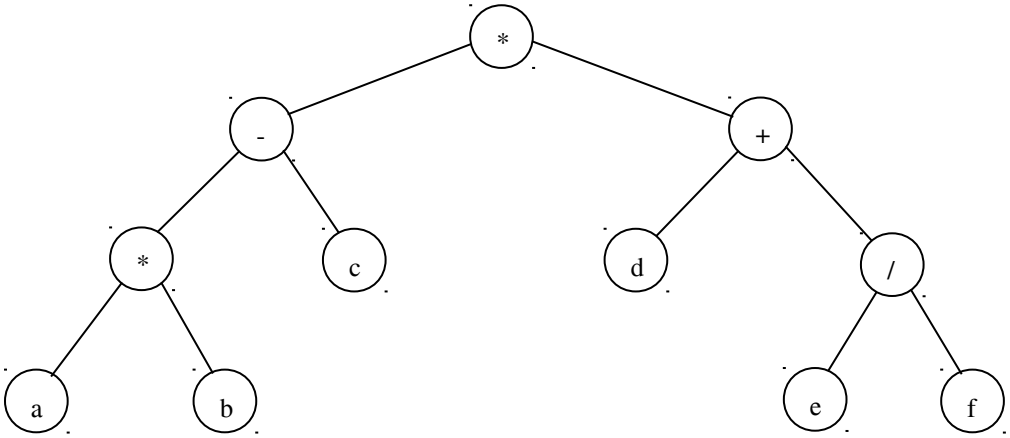
Le traitement d'un arbre binaire revient le plus souvent à appliquer une opération (notons la OP) à tous les nœuds de l'arbre. Pratiquement, cela revient à parcourir la structure et à appliquer l'opération OP à tous les nœuds rencontrés. Parcourir l'arbre impose l'examen de tous ses nœuds et demande de définir un ordre des éléments dans l'arbre.
Pour un arbre binaire, il y a principalement 3 ordres.
Notons par la lettre R la racine de l'arbre à parcourir, G le sous-arbre gauche et D le sous-arbre droit. On peut alors décrire ces ordres de la façon suivante :

1) Parcours **PREORDRE** ou ***préfixe*** (**R, G, D**)
2) Parcours **SYMETRIQUE** ou ***infixe*** (**G, R, D**)
3) Parcours **POSTORDRE** ou ***postfixe*** (**G, D, R**)

Voici un algorithme en Ada pour le parcours infixe d'un arbre.

```
-- Les déclarations pour le type arbre
     type T_Noeud ;
     type  T_Donnee is Integer;
     type T_Arbre is access T_Noeud
     type T_Noeud is record
        Info  : T_Donne;
        SAG   : T_Arbre;
        SAD   : T_Arbre;
     end record;

-- La procédure de traitement: ici l'opération OP est un affichage
     procedure Infixe (A : in  T_Arbre) is
     begin
        if A /= Null then
           Infixe(A.SAG);
           Put(A.INFO);
           Infixe (A.SAD);
        end if;
     end Infixe;
```

Exemple de parcours appliqué à l'arbre de l'expression arithmétique : **`(a*b-c)*(d+e/f)`**

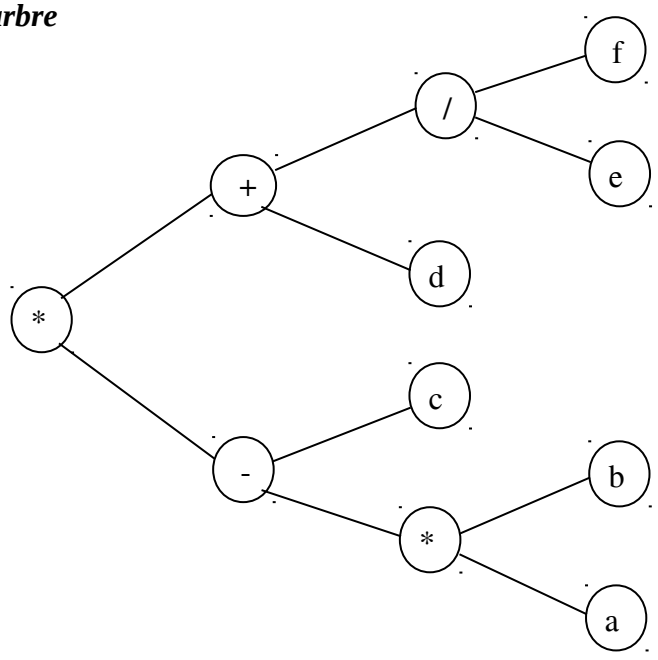


1) Parcours préfixe (**R, G, D**) : `* - * a b c + d / e f`
2) Parcours infixe (**G, R, D**) : `a * b - c * d + e / f`
3) Parcours postfixe (**G, D, R**) : `a b * c - d e f / + *`

Autre exemple de parcours : algorithme simple d'impression d'un arbre

```
-- Les déclarations pour le type arbre
type T_Noeud ;
type T_Donnee is Integer;
type T_Arbre is access T_Noeud
type T_Noeud is record
   Info   : T_Donnee;
   Gauche : T_Arbre;
   Droite : T_Arbre;
end record;

-- La procédure de traitement
procedure Imprime ( A : in  T_Arbre; N : in Positive) is
   I : Positive;
begin
   if A /= Null then
      Imprime (A.Droite, N+1);
      for i in 1..N loop
         Put("    ");
      end loop;
      Put (A.Info);
      Put_Line;
      Imprime (A.Gauche, N+1);
   end if;
end Imprime;
```

Impression d'un arbre



b) La recherche dans un arbre binaire

Les arbres binaires sont souvent utilisés pour stocker de l'information à retrouver plus tard.

Exemple : les arbres lexicographiques

Chaque nœud contient un champ information dont une partie est d'un type ordonné que l'on appelle la clé et dont le but est de mettre les nœuds dans un ordre donné.
Les arbres lexicographiques sont conçus de telle façon que pour tout nœud N, toutes les clés du sous-arbre gauche sont inférieures à celle du nœud N et toutes les clés du sous-arbre droit sont supérieures à la clé du nœud N.
Voici un exemple d'algorithme qui recherche un élément dans un tel arbre.

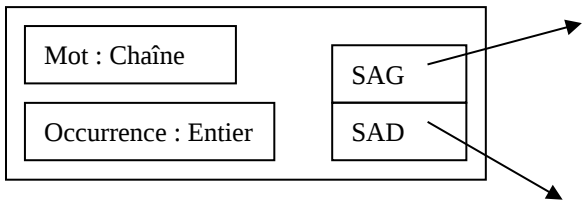
```
type T_Noeud ;
subtype T_Cle is Integer;
type T_Arbre is access T_Noeud;
type T_Noeud is record
   Cle    : T_Cle;
   Gauche : T_Arbre;
   Droite : T_Arbre;
end record;

function Cherche(X : T_Cle; A: T_Arbre) return T_Arbre is
   Succes  : Boolean := False;
   Courant : T_Arbre := A;
begin
   while Courant /= NULL and not Succes loop
      if Courant.Cle = X then
         Succes := True;
      elsif Courant.Cle > X then
         Courant := Courant.Gauche;
      else
         Courant := Courant.Droite;
      end if;
   end loop;
   return Courant;
end Cherche;
```

c) L'insertion dans un arbre binaire

Exemple: la construction d'un arbre lexicographique à partir de la lecture d'un texte

Chaque nœud comporte un mot du texte ainsi que le nombre des apparitions de ce mot dans le texte (nombre d'occurrences de chaque mot). La procédure d'insertion cherche si l'élément à insérer se trouve déjà dans l'arbre, auquel cas on incrémente le compteur associé à ce mot, sinon elle place le mot dans l'arbre.

La structure de données utilisée est la suivante:



Exercice

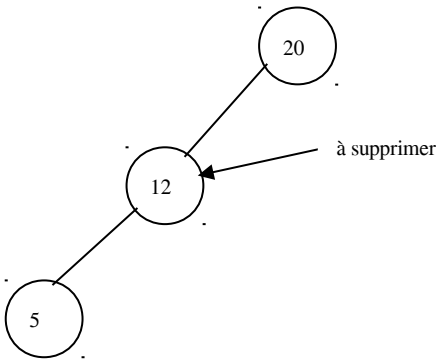
Ecrire une procédure qui place des mots dans un arbre et ensuite affiche cet arbre.

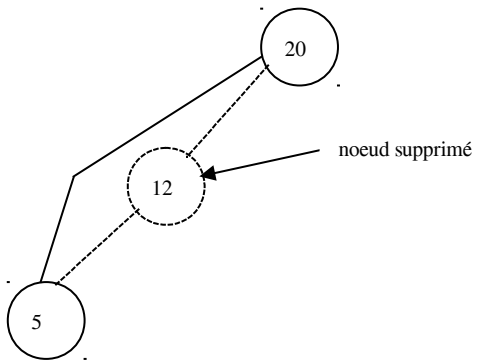
d) La suppression d'un nœud dans un arbre binaire

Plusieurs cas doivent être envisagés selon la position du nœud à supprimer.

Cas simples

- Le nœud à supprimer est une feuille (nœud terminal) : on le supprime simplement.
- Le nœud à supprimer a un seul fils : on rattache le nœud parent au nœud fils, puis on effectue la suppression.

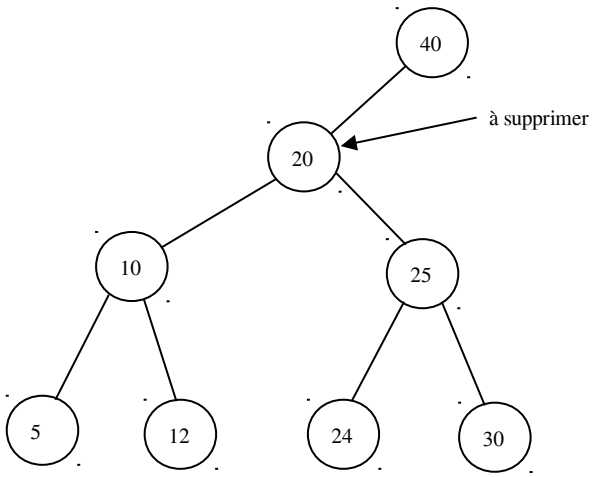




Cas non triviaux

- Le nœud à supprimer a deux descendants : sa valeur est remplacée par celle d'un autre nœud, lequel est ensuite supprimé ; le problème consiste à choisir cet autre nœud.

Exemple



Principe

On remplace l'élément à supprimer par :

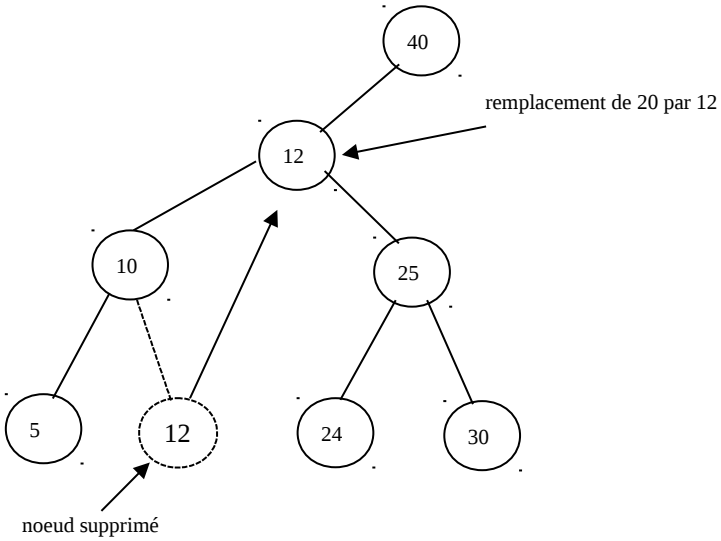
1. le plus grand élément du sous-arbre gauche (c.-à-d. le nœud le plus à droite du sous-arbre gauche) ;

ou bien

2. le plus petit élément du sous-arbre droite (c.-à-d. le nœud le plus à gauche du sous-arbre droite).

Finalement, on supprime le nœud de remplacement (on se retrouve en fait dans un des deux cas simples pour la suppression).

Exemple : solution 1)



Autre exemple