Nom : Sujet : Jury : Page : /

10.- Arbres : représentations et applications

Prérequis : Induction + Programmation OCaml et C Niveau : MP2I

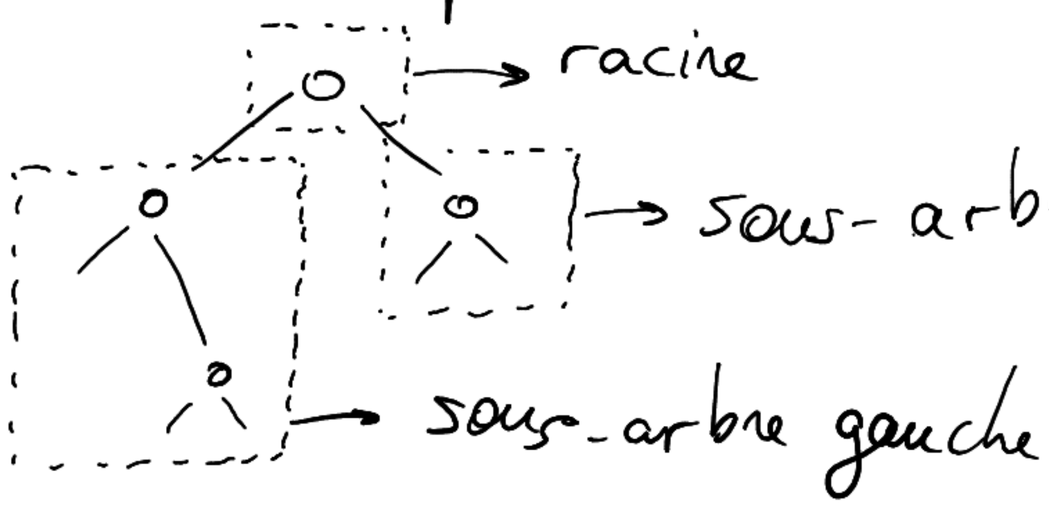
Motivation : Les arbres sont l'une des structure de données fondamentales. Ils permettent de hiérarchiser les données, permettant une optimisation dans un nombre incalculable de problèmes où les structures séquentielles sont moins efficaces. Ils sont omniprésents et d'eux dépendent bon nombre de structures plus riches.

I) Introduction aux arbres binaires

Définition 1 : Un arbre binaire est un ensemble de noeuds organisés hiérarchiquement. On les définit inductivement comme étant :

- Soit un arbre vide, noté E, ne contenant aucun noeud ;
- Soit un noeud étiqueté, appelé racine, relié à exactement deux arbres binaires l et r, appelés respectivement sous-arbres gauche et droit. On note $N(l, x, r)$ un tel arbre dont la racine est étiquetée par x.

Exemple 2 :



Remarque 3 : Les arbres binaires  et  sont distincts.

Définition 4 : Dans un arbre binaire, lorsqu'un noeud a possède un sous-arbre b non vide, on dit que a est le père de b, et que b est le fils de a.

Définition 5 : Dans un arbre binaire, un noeud dont les deux sous-arbres sont vides est appelé une feuille. Les noeuds n'étant pas des feuilles sont appelés les noeuds internes.

Définition 6 : La hauteur d'un arbre binaire t, usuellement notée $h(t)$, est définie récursivement par : $h(E) = -1$ et $h(N(l, a, r)) = 1 + \max(h(l), h(r))$.

Propriété 7 : Soient t un arbre binaire, n son nombre de noeuds et h sa hauteur. Alors, on a les propriétés suivantes :
- $h+1 \leqslant n \leqslant 2^{h+1} - 1$;
- le nombre de sous-arbres vides est $n+1$.

Représentations 8 : Représentation des arbres binaires en OCaml et C

C

```
typedef struct arbre_bin {
    struct bintree *gauche;
    int valeur;
    struct bintree *droit;
} arbre_bin_t;
```

OCaml

```
type 'a arbre_bin =
  | E
  | N of 'a arbre_bin * 'a * 'a arbre_bin
```

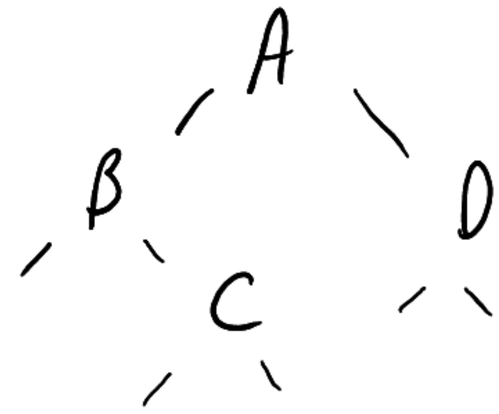
Définition 9 : On dit qu'un arbre binaire est parfait si tous ses niveaux sont complètement remplis. Lorsque tous les niveaux sont remplis à l'exception du dernier, et que celui-ci est rempli à partir de la gauche, on dit qu'il est complet.

Remarque 10 : Il est possible de représenter efficacement en mémoire les arbres binaires complets (et donc également les parfaits) à l'aide d'un tableau.

Définition 11 : On appelle parcours d'un arbre binaire une énumération de ses sommets où chaque sommet apparaît une et une seule fois. On désigne également par métonymie un algorithme visitant chaque noeud de l'arbre une unique fois et en effectuant un traitement.

Proposition 12: Trois parcours canoniques se dégagent :
- le parcours préfixe, visitant le noeud, son sous-arbre gauche puis son sous-arbre droit ;
- le parcours infixe, visitant le sous-arbre gauche, le noeud puis le sous-arbre droit ;
- le parcours suffixe, visitant le sous-arbre gauche, le sous-arbre droit puis le noeud.

Exemple 13 :



→ Parcours préfixe : A → B → C → D

→ Parcours infixe : B → C → A → D

→ Parcours suffixe : C → B → D → A.

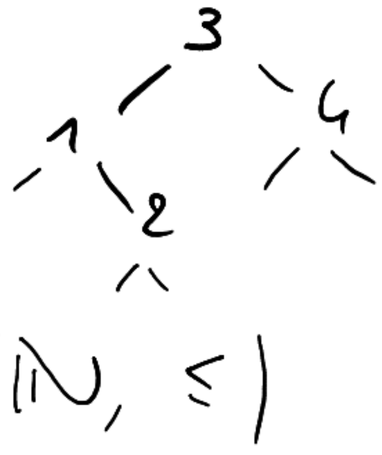
Activité 14 : TP d'implémentation des arbres binaires, des opérations usuelles (hauteur, nombre de noeuds internes et de feuilles) et des parcours sur des exemples donnés en C et OCaml.

II) Structures sur des arbres binaires :

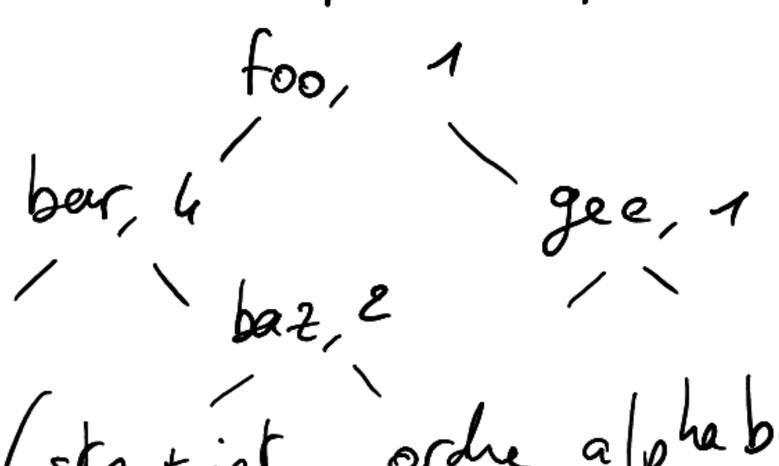
1.) Arbres binaires de recherche

Définition 15 : Un arbre binaire de recherche est un arbre binaire où les éléments sont munis d'un ordre total et, pour chaque sous-arbre $N(l, x, r)$, x est plus grand que tous les éléments de l, et plus petit que ceux de r.

Exemples 16 :



$(\mathbb{N}, \leq)$



(str * int, ordre alphabétique sur les str)

Propriété 17 : Les arbres binaires de recherche permettent d'implémenter les tableaux associatifs.

Nom : Sujet : Jury : Page : /

Activité 18 : Implémentation des tableaux associatifs à l'aide d'arbre binaire de recherche en C et OCaml.

C

```
typedef struct abr {
    struct abr *gauche;
    char *clef;
    int valeur;
    struct abr *droit;
} 
```

OCaml

```
type ('c, 'v) abr =
  | E
  | N of ('c, 'v) abr * 'c * 'v * ('c, 'v) abr
```

Définition 19 : Soit S un ensemble d'arbres binaires. On dit que les arbres de S sont équilibrés s'il existe une constante C telle que pour tout arbre non vide t de S, on a :
$h(t) \leq C \times \log(n(t))$.

DEV : arbres rouge-noir

2.) Tas et files de priorité

Définition 20 : Une file de priorité est une structure de données permettant de manipuler des multiensembles totalement ordonnés. Les opérations élémentaires sur les files de priorités sont :
- ajouter un élément avec une priorité donnée dans la file de priorité.
- retirer l'élément avec la priorité la plus basse de la file de priorité.

Définition 21 : Un tas (min) est un arbre binaire t tel que t est E ou de la forme $N(l, x, r)$ où l et r sont des tas et x est inférieur à tous les éléments de l et r.

Propriété 22 : Les tas permettent d'implémenter les files de priorité.

Remarque 23: On peut définir les tas-max de manière similaire aux tas-min en remplaçant "inférieur" par "supérieur" dans la définition des tas. On peut facilement passer de l'un à l'autre en considérant les ordres inverses.

Propriété 24: Dans un tas comprenant n éléments, l'insertion d'un nouvel élément et la suppression de la racine s'effectuent en temps $O(\log(n))$.

DEV: tri par tas

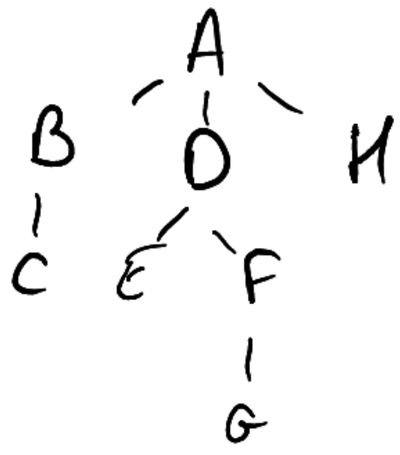
Activité 25: Implémentation des tas et du tri par tas en OCaml.

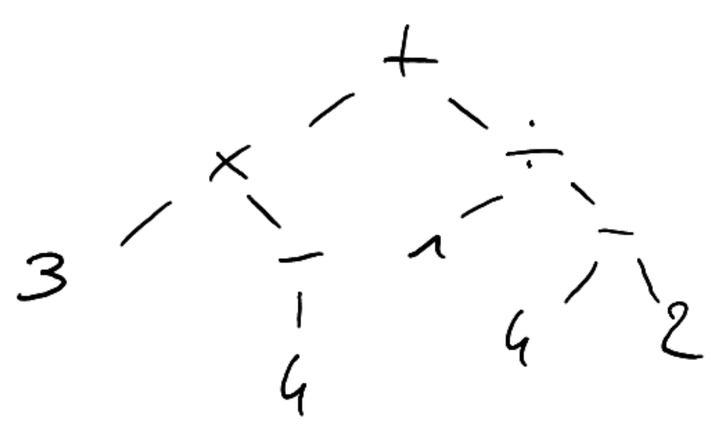
III) Généralisation :

1.) Arbres

Définition 26: Un arbre est une structure de données hiérarchique représentant un ensemble de $n \geq 1$ noeuds tel que :
- Un noeud est désigné racine de l'arbre ;
- les $n-1$ noeuds restants sont partitionnés en k sous-ensembles disjoints et ordonnés formant autant d'arbres, appelés sous-arbres de r ;
- la racine r est liée à la racine de chacun des k sous-arbres.

Exemple 27:





Remarque 27 : Les arbres binaires ne sont pas des arbres : un arbre ne peut pas être vide et il y a une distinction entre sous-arbre gauche et droit : il y a deux arbres binaires de 2 noeuds (et) mais un seul arbre de 2 noeuds ( ).

Définition 28 : La hauteur d'un arbre est définie comme la longueur d'un chemin entre la racine et un noeud de l'arbre. On considère que la hauteur d'un arbre à 1 noeud est 0.

Proposition 29 : Il est possible de convertir un arbre en un arbre binaire et réciproquement. On utilise pour cela l'existence d'un isomorphisme entre :

- Un arbre binaire et un ensemble d'arbres,
- Un arbre binaire non vide dont le sous-arbre droit est vide et un arbre.

2.) Arbres préfixes

Définition 31 : Un arbre préfixe (ou trié) est une structure permettant de représenter un tableau associatif dont les clés sont des chaînes de caractères. Dans un arbre préfixe, chaque branche est étiquetée par une lettre, et chaque noeud contient une valeur si la séquence de lettres menant à la racine est une entrée du tableau associatif.

Exemple 32 :