

## Algorithmique 1

### TD 4 Arbres binaires

On considère le type abstrait `arbreBinaire` de `type_predefini` avec les primitives suivantes :

```

fonction valeurSommet(val A: arbreBinaire de type_predefini,
                      val S: sommet): type_predefini;
fonction racine(val A: arbreBinaire de type_predefini): sommet;
fonction filsGauche(val A: arbreBinaire de type_predefini, val S: sommet): sommet;
fonction filsDroit(val A: arbreBinaire de type_predefini, val S: sommet): sommet;
fonction pere(val A: arbreBinaire de type_predefini, val S: sommet): sommet;
fonction CreerArbreBinaire(ref A: arbreBinaire de type_predefini,
                           val racine: type_predefini): vide;
fonction ajouterFilsGauche(val x: type_predefini,
                           ref A: arbreBinaire de type_predefini, val S: sommet): vide;
fonction ajouterFilsDroit(val x: type_predefini,
                           ref A: arbreBinaire de type_predefini, val S: sommet): vide;
fonction supprimerFilsGauche(ref A: arbreBinaire de type_predefini, val S: sommet): vide;
fonction supprimerFilsDroit(ref A: arbreBinaire de type_predefini, val S: sommet): vide;

```

#### Exercice 4.1 *Parcours*

Soit l'arbre binaire dont les feuilles sont étiquetées avec les nombres naturels illustré sur Fig.1.

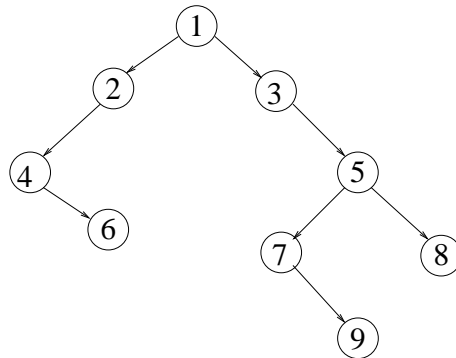


FIG. 1 – Arbre binaire

1. Donner les mots correspondants resp. à son parcours préfixe, infixé et postfixé.
2. Existe-t-il un arbre binaire à 8 sommets dont le mot préfixe est 12345678 et le mot suffixe est
  - (a) 53247681 ?
  - (b) 43527681 ?
3. Donner le principe d'un algorithme pour reconstruire un arbre binaire à partir de ces mots préfixe et suffixe.

### Exercice 4.2 *Parcours hiérarchique*

Soit le tableau d'entiers  $\text{Tab}=\{0, 2, 3, 4, 6, 7, 9, 10, 12, 13, 14, 16, 20\}$  et la fonction `remplirTableauArbre` vue en cours. Dessiner l'arbre binaire produit par l'appel à `remplirTableauArbre(Tab)`.

```
fonction remplirTableauArbre(  
    val T: tableau[1..N] d'entiers): arbreBinaire d'entiers;  
var A: arbreBinaire d'entiers;  
var F: file de sommets;  
var S: sommet;  
var i, tmp, s: entier;  
debut  
    creerFile(F);  
    creerArbreBinaire(A, T[1]);  
    enfiler(F, racine(A));  
    tmp= 2;  
    tantque 2*tmp-1 <= N faire  
        pour i allant de tmp a 2*tmp - 1 par pas de 2 faire  
            S= valeur(F);  
            defiler(F);  
            ajouterFilsGauche(T[i], A, S);  
            enfiler(filsGauche(A, S));  
            ajouterFilsDroit(T[i+1], A, S);  
            enfiler(filsDroit(A, S));  
        finpour  
        tmp= tmp * 2;  
    fintantque  
    pour i allant de tmp a N par pas de 2 faire  
        S= valeur(F);  
        defiler(F);  
        ajouterFilsGauche(T[i], A, S);  
        ajouterFilsDroit(T[i+1], A, S);  
    finpour  
    retourner(A);  
fin  
finfonction
```

### Exercice 4.3 *Parcours en profondeur*

Utiliser un parcours en profondeur pour écrire les fonctions :

1. `compter (val A : arbreBinaire): entier` qui compte le nombre de feuilles dans A.
2. `hauteur (val A : arbreBinaire): entier` qui calcule la hauteur de A. .
3. `valeur_minimum(val A : arbreBinaire): entier` qui retourne le minimum des valeurs contenues dans A.

**Exercice 4.4** *Appartenance, égalité*

1. Ecrire une fonction qui teste si un élément appartient à un arbre binaire.
2. Ecrire une fonction qui teste l'égalité de deux arbres binaires.
3. Ecrire une fonction qui teste si un arbre binaire est un sous arbre d'un autre arbre binaire.

**Exercice 4.5** *Arbre binaire complet*

On appelle arbre binaire complet un arbre binaire tel que chaque sommet possède 0 ou 2 fils.

1. Donner des exemples d'arbres binaires complets.
2. Ecrire une fonction qui teste si un arbre binaire est complet  
`arbreBinaire_complet(val A:arbreBinaire ): boolean;`

**Exercice 4.6** *Arbre binaire parfait*

On appelle arbre binaire parfait un arbre binaire (complet) tel que chaque sommet soit père de deux sous arbres de même hauteur.

1. Donner des exemples d'arbres binaires parfaits.
2. Ecrire une fonction qui teste si un arbre binaire est parfait.  
`arbreBinaire_parfait(val A:arbreBinaire ): boolean;`

**Exercice 4.7** *Arbre binaire quasi-parfait*

On appelle arbre binaire quasi-parfait un arbre binaire parfait éventuellement grignoté d'un étage en bas à droite.

1. Donner des exemples d'arbres binaires quasi-parfaits.
2. Ecrire une fonction qui teste si un arbre binaire est quasi-parfait  
`arbreBinaire_quasi_parfait(val A:arbreBinaire ): boolean;`