Algorithmique 1

DS 3 : Arbres Binaires et ABR

Documents **non** autorisés

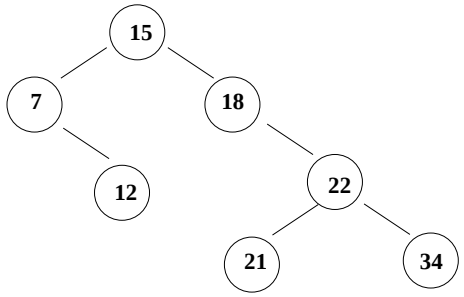
Exercice 3.1

Dans l'exercice suivant, ou considère la fonction `mystere1` définie de la manière suivante :

```
fonction mystere1(ref A : ABR, ref T : tableau[1..N] d'entier) vide;
debut
    si A!= NIL alors
    mystere1-aux (A, T, 0);
    fsi
fin;

fonction mystere1-aux(ref A : sommet, ref T : tableau[1..N] d'entier, ref i:entier) vide;
debut
     si filsGauche(A) != NIL alors
         mystere1-aux(filsGauche(A),T, I)
     fsi;
     I= I+1;
     T[I] = A ;
     si filsDroit(A) != NIL alors
     mystere1-aux(filsDroit(A),T, I)
     fsi;
     N=I;
fin
```

1. Quel type de parcours est utilisé dans la fonction `mystere1` ?
   Justifier votre réponse.



2. L'arbre binaire dessiné ci-dessus est-il un arbre binaire de recherche ? Justifier votre réponse.

3. Ajouter successivement à l'arbre binaire dessiné ci-dessus les valeurs : $19, 13, 15, 7, 30, 1$
4. Quelle est le résultat de l'application de `mystere1` sur l'arbre ainsi obtenu ?
5. De manière générale, que fait `mystere1` ?

Exercice 3.2

Dans l'exercice suivant, on demande d'écrire la fonction inverse de `mystere1` dont la signature est la suivante : `mystere2(ref T: tableau[1..N], ref A: arbre binaire): vide`.

Exercice 3.3

1. Ecrire une fonction qui vérifie si un arbre binaire est inclu dans un arbre binaire de recherche.
2. Quelle est la compléxité de votre fonction ? Justifier votre réponse.

Annexe C : Implémentation du type abstrait arbreBinaire.

```
cellule=structure
         info:objet;
         gauche: sommet;
         droit: sommet;
         pere: sommet;
       finstructure
sommet=^cellule;
arbreBinaire= sommet;
```

Annexe

Primitives du type abstrait arbre binaire d'objet

```
fonction creerArbreBinaire(val Racine:objet):sommet;
fonction getValeur(val S:sommet):objet;
fonction filsGauche(val S:sommet):sommet;
fonction filsDroit(val S:sommet):sommet;
fonction pere(val S:sommet):sommet;
fonction setValeur(ref S:sommet, val x:objet):vide;
fonction ajouterFilsGauche(ref S:sommet, val x:objet):vide;
fonction ajouterFilsDroit(ref S:sommet, x:objet):vide;
fonction supprimerFilsGauche(ref S:sommet):vide;
fonction supprimerFilsDroit(ref S:sommet):vide;
fonction detruireSommet(ref S:sommet):vide;
fonction creerArbreBinaire(val Racine:objet):sommet;
```