

Algorithmique et Structures de données

Feuille 5 : Arbres binaires

On considère le type abstrait `arbreBinaire` d'objet défini en cours.
 Pour rappel voir annexe A.

Exercice 5.1 *Implémentation du type abstrait `arbreBinaire`*
 (rappel du cours en Annexe B)

- Soit l'arbre de la Fig.1(a). Son implémentation dynamique est illustrée sur la Fig.1(b).
 On ajoute une feuille au dernier niveau. Dessiner la nouvelle structure.

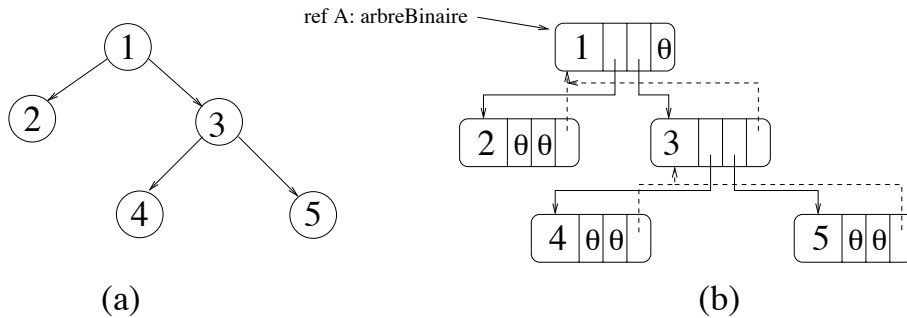


FIGURE 1 – Arbre binaire complet

- Compléter l'implémentation vue en cours par les primitives :
`filesDroit`, `pere`, `ajouterFilesDroit`, `supprimerFilesDroit`.

Exercice 5.2 *Parcours*

Soit l'arbre binaire dont les feuilles sont étiquetées avec les nombres naturels illustré sur Fig.2.

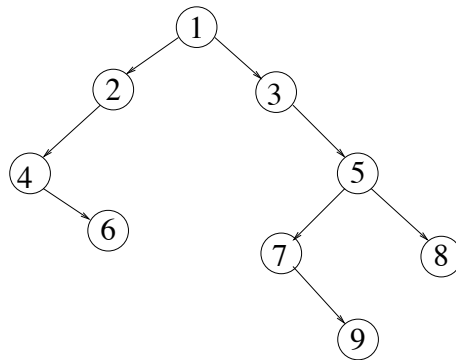


FIGURE 2 – Arbre binaire

1. Donner les mots correspondants resp. à son parcours préfixe, infixé et suffixé.

2. Existe-t-il un arbre binaire à 8 sommets dont le mot préfixe est 12345678 et le mot suffixe est

(a) 53247681 ?

(b) 43527861 ?

On supposera que les feuilles sont étiquetées avec les nombres naturels n , $n \leq 8$.

3. Donner le principe d'un algorithme pour reconstruire un arbre binaire à partir de ses mots infixes, préfixe et suffixe.

Exercice 5.3 *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 (annexe C).

Dessiner l'arbre binaire produit par l'appel à `remplirTableauArbre(Tab)`.

Exercice 5.4 *Parcours en profondeur*

En utilisant l'algorithme de parcours en profondeur :

1. écrire une fonction qui compte le nombre de feuilles dans un arbre binaire.
 2. écrire une fonction qui calcule la hauteur d'un arbre binaire.
 3. écrire une fonction qui retourne le minimum des valeurs contenues dans un arbre binaire.
 4. écrire une fonction qui teste si un élément appartient à un arbre binaire.
- endenumerate

Exercice 5.5 *Arbre binaire complet*

On appelle **arbre binaire complet** un arbre binaire tel que chaque sommet interne a exactement 2 fils.

1. Donner des exemples d'arbres binaires complets.
2. Ecrire une fonction qui teste si un arbre binaire est complet.

Exercice 5.6 *Arbre binaire parfait*

On appelle **arbre binaire parfait** un arbre binaire complet dans lequel toutes les feuilles sont à la même hauteur dans l'arbre.

1. Donner des exemples d'arbres binaires parfaits.
2. Ecrire une fonction qui teste si un arbre binaire est parfait.

Exercice 5.7 *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.

Exercice 5.8 *Evaluation d'expressions arithmétiques*

On suppose donné un arbre binaire représentant une expression arithmétique contenant des entiers et des opérateurs binaires `+` et `*`. Si l'expression est bien formée (correcte) alors

l'arbre binaire est complet. Chaque feuille de cet arbre contient donc un entier et chaque noeud interne un opérateur (l'un des symboles + ou *).

On adopte comme convention que le symbole + est codé par -1 et * par -2.

Ecrire une fonction `valeurExpression(val A: arbreBinaire)` qui évalue la valeur de l'expression représentée par un arbre binaire si l'arbre est complet sinon renvoie `NIL`, en visitant une et une seule fois tous les noeuds de l'arbre (un seul parcours de l'arbre).

Problème récurrent (notre fil d'ariane)

Exercice 5.9 Gestion d'une piste d'atterrissage des avions

Un avion est un enregistrement contenant :

- l'indicatif (6 caractères)
- la destination (30 caractères)
- l'autonomie résiduelle de carburant comptée en heures de vol (entier)
- deux booléens indiquant s'il y a un pirate à bord et s'il y a le feu.

1. Définir les structures de données nécessaires.
2. Ecrire la fonction `Priorité` ainsi que la gestion complète de la piste.
3. Envisager le cas de suppression d'un élément quelconque de la file lorsque le pirate a mis sa menace de détournement à exécution.

Quelles sont les notions que vous venez de voir qui peuvent permettre d'amorcer le fil d'ariane ?

ANNEXE A Type abstrait *arbreBinaire*

```
arbreBinaire= curseur;  
sommet= curseur;
```

– Création

```
fonction creerArbreBinaire(val Racine:objet):sommet;
```

– Accès

```
fonction getValeur(val S:sommet):objet;  
fonction filsGauche(val S:sommet):sommet;  
fonction filsDroit(val S:sommet):sommet;  
fonction pere(val S:sommet):sommet;
```

– Modification

```
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;
```

ANNEXE B Implémentation du type abstrait *arbreBinaire*

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

ANNEXE C Construction d'un arbre binaire à partir d'un tableau

```
fonction remplirTableauArbre(ref T:tableau[1..N] d'entier):
    arbreBinaire d'entier;
var A:arbreBinaire d'entier;
var F: file de sommet;
var s:sommet;
var i:entier;
debut
    creerFile(F);
    A= creerArbreBinaire(T[1]);
    enfiler(F, A);
    tmp= 2;
    tantque 2*tmp-1 <= N faire
        pour i= tmp a 2*tmp-1 par pas de 2 faire
            s= getValeur(F);
            defiler(F);
            ajouterFilsGauche(s, T[i]);
            enfiler(F, filsGauche(s));
            ajouterFilsDroit(s, T[i+1]);
            enfiler(F, filsDroit(s));
        finpour;
        tmp= tmp*2;
    fintantque
    pour i=tmp a N-1 par pas de 2 faire
        s= getValeur(F);
        defiler(F);
        ajouterFilsGauche(s, T[i]);
        ajouterFilsDroit(s, T[i+1]);
    finpour;
    si N mod 2 == 0 alors
        ajouterFilsGauche(getValeur(F), T[N])
    finsi
    detruireFile(F);
    retourner(A);
fin
```