

Algorithmique 1

TD 6 Arbres binaires : implémentation dynamique

Soit le type abstrait `arbreBinaire` de `type_predefini` représenté comme il suit :

```

type sommet= structure
    valeur: type_predefini;
    gauche: ^sommet;
    droit: ^sommet;
    pere: ^sommet;
finstructure
type arbreBinaire= ^sommet;
    
```

On dispose des primitives suivantes :

```

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

Exercice 6.1 *Structures de données*

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

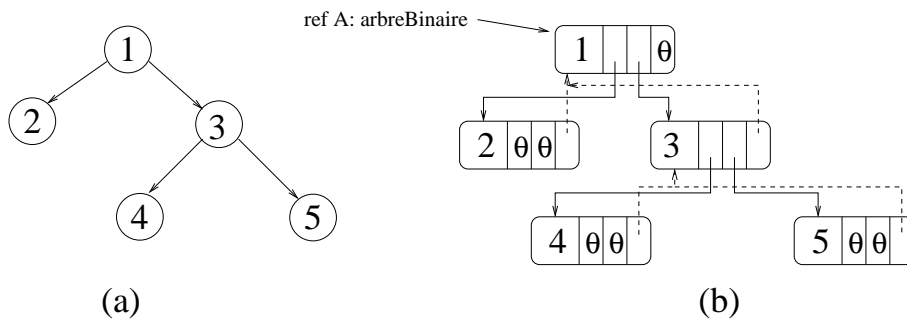


FIG. 1 – Arbre binaire complet

Exercice 6.2 *Construction*

Dessiner l'arbre binaire construit par la fonction `creer_arbre_non_quasi_parfait`. Illustrer son implémentation dynamique construite par la fonction suivante :

```

fonction creer_arbre_non_quasi_parfait(ref a: arbreBinaire):vide;
var s1, s2, s3, s4, s6: sommet;
  debut
    CreerArbreBinaire(a, 1);
    sommet s1= racine(a);
    ajouterFilsGauche(2, a, s1);
    sommet s2= filsGauche(a, s1);
    ajouterFilsDroit(3, a, s1);
    sommet s3= filsDroit(a, s1);
    ajouterFilsGauche(4, a, s2);
    sommet s4= filsGauche(a, s2);
    ajouterFilsDroit(5, a, s2);
    ajouterFilsGauche(6, a, s3);
    sommet s6= filsGauche(a, s3);
    ajouterFilsDroit(7, a, s3);
    ajouterFilsGauche(8, a, s4);
    ajouterFilsDroit(9, a, s4);
    ajouterFilsGauche(10, a, s6);
  fin
finfonction

```

Exercice 6.3 *Primitives*

Compléter l'implémentation vue en cours par les primitives :

`filsDroit`, `pere`, `ajouterFilsDroit`, `supprimerFilsDroit`

Exercice 6.4 *Exemples*

Écrire les fonctions :

```

fonction est_feuille(val A: arbreBinaire, val s: sommet): boolean;
fonction hauteur(val A: arbreBinaire): entier;
fonction afficher_arbre_parcours_prefixe(val A: arbreBinaire): vide;

```

Exercice 6.5 *Même squelette*

Écrire une fonction `meme_squelette` qui teste si deux arbres binaires sont semblables à la valeur près des valeurs contenues dans les sommets.

Exercice 6.6 *Arbre binaire de Calder*

Écrire une fonction `poids` qui calcule la somme des valeurs contenues dans les sommets d'un arbre binaire. En déduire un prédicat `est_Calder` qui teste si, en tous les noeuds, les deux fils ont le même poids.