Université de Bordeaux **Semestre 3 2015-2016**
Licence Informatique et Math-Informatique **Algorithmique 1**

TD 6 Arbres planaires

Soit la définition du type abstrait `sommetArbrePlanaire` vue en cours.
Pour rappel voir annexe A.

Exercice 6.1 *Exemples*

Soit l'arbre planaire illustré sur la figure 1.

1. Donner les suites de sommets correspondant respectivement aux parcours préfixe, postfixe et hiérarchique.
2. Tout noeud d'un arbre est la racine du sous-arbre constitué par sa descendance et lui-même. Dénombrer l'ensemble de sous-arbres pour l'arbre planaire de la figure 1 .

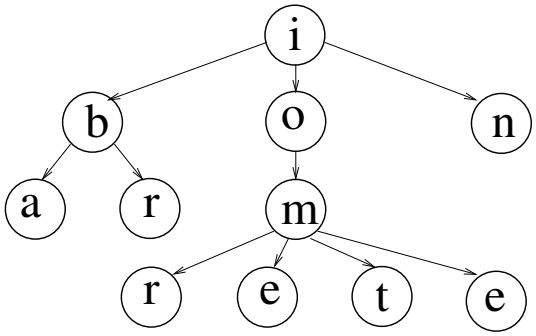

FIGURE 1 – Arbre planaire

Exercice 6.2 *Construction*

En utilisant les primitives du type `sommetArbrePlanaire` écrire une fonction qui construit l'arbre planaire de la figure 2(a).

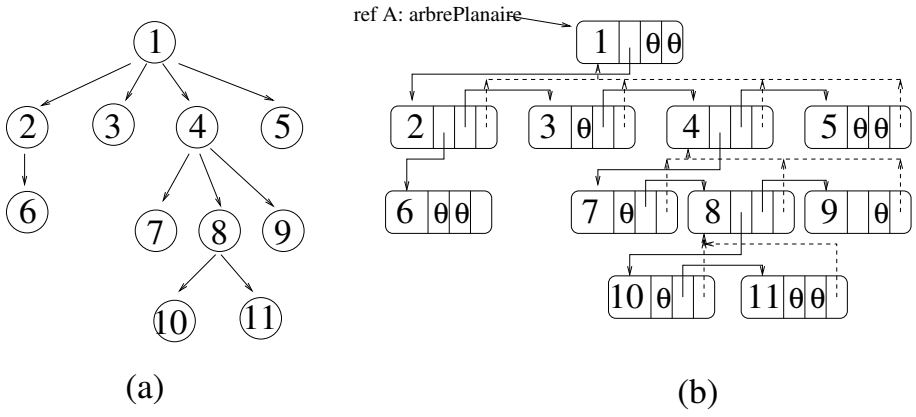

FIGURE 2 – Représentation d'un arbre planaire

Exercice 6.3 *Représentation binaire d'un arbre planaire*

Tout arbre planaire peut être représenté par un arbre binaire en prenant comme lien gauche de tout noeud le lien vers son premier fils, et comme lien droit le lien vers son frère de droite.

1. Soit l'arbre planaire de la figure 2(a). Dessiner sa représentation binaire.
2. Dessiner l'arbre planaire qui correspond à la représentation binaire illustrée sur la figure 3.
3. Démontrer que les primitives du type `arbreBinaire` sont réalisables par les primitives du type `sommetArbrePlanaire` et vice versa.

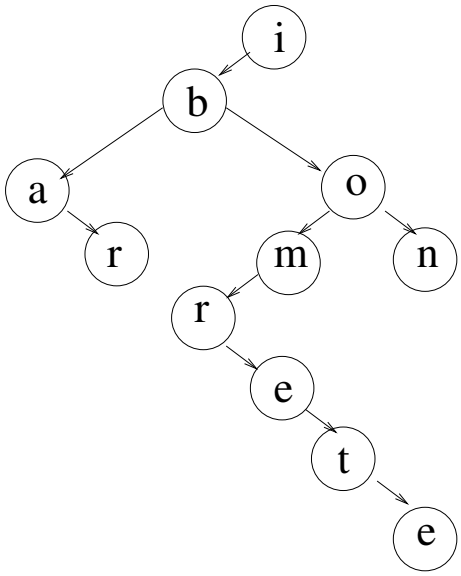

FIGURE 3 – Représentation binaire d'un arbre planaire

Exercice 6.4

Soit l'implémentation de `sommetArbrePlanaire` vue en cours et illustrée sur la figure 2(a).
Pour rappel voir annexe B.
Ecrire les primitives `ajouterFils` et `supprimerSommet`

Exercice 6.5 *Parcours itératif*

1. Ecrire une fonction de parcours préfixe itératif d'un arbre planaire en utilisant une pile.
2. Ecrire une fonction de parcours hiérachique itératif d'un arbre planaire en utilisant une file.

Problème récurrent

Exercice 6.6 Gestion d'une piste d'atterrissage

Un avion est caractérisé par un enregistrement contenant :
— un indicatif (6 caractères)
— sa destination (30 caractères)
— son 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.

Le problème consiste à

1. définir les structures de données nécessaires à la gestion d'une piste d'atterrissage ;
2. définir et écrire une fonction calculant la priorité d'un avion pour l'utilisation de la piste ;
3. définir et écrire les fonctions nécessaires à la gestion complète de la piste (on envisagera la suppression d'un avion piraté de la file d'attente lorsque le pirate a mis sa menace de détournement à exécution).

Quelles notions vues dans ce TD peuvent permettre d'amorcer la résolution du problème ?

Annexe A **Type abstrait** *sommetArbrePlanaire*

```
sommetArbrePlanaire= curseur;
```

- **Création**
  `fonction creerArbrePlanaire(val Racine:objet):sommetArbrePlanaire;`
- **Accès**
  `fonction valeur(val S:sommetArbrePlanaire):objet;`
  `fonction premierFils(val S:sommetArbrePlanaire):sommetArbrePlanaire;`
  `fonction frere(val S:sommetArbrePlanaire):sommetArbrePlanaire;`
  `fonction pere(val S:sommetArbrePlanaire):sommetArbrePlanaire;`
- **Modification**
  `fonction setValeur(ref S:sommetArbrePlanaire, val x:objet):vide;`
  `fonction ajouterFils(ref S:sommetArbrePlanaire, val x:objet):vide;`
  `fonction supprimerSommet(ref S:sommetArbrePlanaire):vide;`
  `fonction detruireArbrePlanaire(ref S:sommetArbrePlanaire):vide;`

Annexe B **Implémentation du type abstrait** *sommetArbrePlanaire*

```
cellule= structure
          info: objet;
          premierFils: sommet
          frere: sommet;
          pere: sommet
         finstructure
sommet= ^cellule;
sommetArbrePlanaire= sommet;
```