TP 3 – Arbres Binaires

Un arbre binaire est une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément est appelé noeud, le noeud initial étant appelé racine. Dans un arbre binaire, chaque élément possède au plus deux éléments fils au niveau inférieur, habituellement appelés gauche et droit. Du point de vue de ces éléments fils, l'élément dont ils sont issus au niveau supérieur est appelé père. Au niveau le plus élevé il y a donc un noeud racine. Au niveau directement inférieur, il y a au plus deux noeuds fils. En continuant à descendre aux niveaux inférieurs, on peut en avoir quatre, puis huit, seize, etc. c'est-'a-dire la suite des puissances de deux. Un noeud n'ayant aucun fils est appelé feuille, sinon il est appelé noeud interne. Le nombre de niveaux total, autrement dit la distance entre la feuille la plus éloignée et la racine, est appelé hauteur de l'arbre. Le niveau d'un noeud est appelé profondeur.

Exercice 1. Un type pour les arbres binaires
En utilisant les types sommes, définir le type `bintree` permettant de représenter les arbres binaires définis sur des entiers. Utiliser ce type pour définir l'arbre binaire représenté en Figure 1.

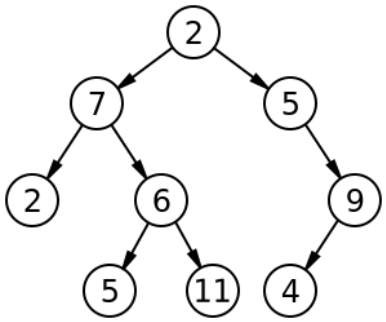

FIGURE 1 – Exemple d'arbre binaire

Exercice 2. Navigation dans l'arbre binaire
1. Écrire la fonction `bintree_count_nodes` qui calcule le nombre de noeuds dans un arbre binaire.
2. Écrire la fonction `bintree_count_leaves` qui calcule le nombre de feuilles dans un arbre binaire.
3. Écrire la fonction `bintree_count_internal_nodes` qui calcule le nombre de noeuds internes dans un arbre binaire.

```
# bintree_count_nodes;;
- : bintree -> int = <fun>
# bintree_count_nodes example_tree;;
- : int = 9
# bintree_count_leaves;;
- : bintree -> int = <fun>
# bintree_count_leaves example_tree;;
- : int = 4
# bintree_count_internal_nodes;;
- : bintree -> int = <fun>
# bintree_count_internal_nodes example_tree;;
- : int = 5
```

Exercice 3. Propriétés

1. Écrire la fonction `bintree_height` qui retourne la hauteur d'un arbre binaire.

```
# bintree_height;;
- : bintree -> int = <fun>
# bintree_height example_tree;;
- : int = 4
```

Un arbre binaire est symétrique s'il est possible de tracer une ligne verticale passant par la racine telle que le sous-arbre gauche est l'image miroir du sous-arbre droit.

2. Écrire la fonction `bintree_is_mirror` qui retourne vrai si un arbre binaire est l'image mirroir d'un autre arbre binaire. Nous nous intéressons ici à la structure des deux arbres et pas aux valeurs des noeuds.

3. Écrire la fonction `bintree_is_symetric` qui retourne vrai si un abre binaire est symétrique.

```
# bintree_is_mirror;;
- : bintree -> bintree -> bool = <fun>
# bintree_is_mirror Empty Empty;;
- : bool = true
# bintree_is_mirror (Node(1, Empty, Empty)) (Node(2, Empty, Empty));;
- : bool = true
# bintree_is_mirror (Node(1, Empty, Node(3, Empty, Empty)))
                    (Node(2, Node(4, Empty, Empty), Empty));;
- : bool = true
# bintree_is_symmetric;;
- : bintree -> bool = <fun>
# bintree_is_symmetric Empty;;
- : bool = true
# bintree_is_symmetric (Node(1, Empty, Empty));;
- : bool = true
# bintree_is_symmetric (Node(1, Node(2, Empty, Empty), Node(2, Empty, Empty)));;
- : bool = true
# bintree_is_symmetric (Node(1, Node(2, Empty, Empty), Node(3, Empty, Empty)));;
- : bool = true
# bintree_is_symmetric (Node(1, Node(2, Empty, Empty), Empty));;
- : bool = false
```

Exercice 4. Visiter

1. Écrire la fonction `bintree_pre` qui retourne la liste des entiers rencontrés dans un arbre binaire lors d'un parcours préfixé.

2. Écrire la fonction `bintree_post` qui retourne la liste des entiers rencontrés dans un arbre binaire lors d'un parcours postfixé.

3. Écrire la fonction `bintree_in` qui retourne la liste des entiers rencontrés dans un arbre binaire lors d'un parcours infixé.

```
# bintree_visit_pre;;
- : bintree -> int list = <fun>
# bintree_visit_pre example_tree;;
- : int list = [2; 7; 2; 6; 5; 11; 5; 9; 4]
# bintree_visit_post;;
- : bintree -> int list = <fun>
# bintree_visit_post example_tree;;
- : int list = [2; 5; 11; 6; 7; 4; 9; 5; 2]
# bintree_visit_in;;
- : bintree -> int list = <fun>
# bintree_visit_in example_tree;;
- : int list = [2; 7; 5; 6; 11; 2; 5; 9; 4]
```

Exercice 5. Arbre binaire de recherche

Un arbre binaire de recherche est un arbre binaire tel que pour chaque neoud, les entiers stockés dans son sous-arbre gauche sont inférieurs ou égaux, et les entiers stockés dans son sous-arbre droit sont supérieurs

1. Écrire la fonction `bintree_insert` qui insert un entier dans un arbre binaire de recherche.

2. Écrire la fonction `bintree_search` qui retourne vrai si un entier donné est dans un arbre binaire de recherche.

```
# bintree_insert;;
- : bintree -> int -> bintree = <fun>
# bintree_insert Empty 1;;
- : bintree = Node (1, Empty, Empty)
# bintree_insert (bintree_insert Empty 1) 2;;
- : bintree = Node (1, Empty, Node (2, Empty, Empty))
# bintree_insert (bintree_insert (bintree_insert Empty 1) 2) 3;;
- : bintree = Node (1, Empty, Node (2, Empty, Node (3, Empty, Empty)))
# bintree_insert (bintree_insert (bintree_insert Empty 1) 2) 0;;
- : bintree = Node (1, Node (0, Empty, Empty), Node (2, Empty, Empty))
# bintree_search;;
- : bintree -> int -> bool = <fun>
```