TD 5 - Les arbres

1 Les arbres

Exercice 1

Implémentez les arbres étiquetés. Pour cela, vous implémenterez les fonctions suivantes :

```
def fils(A, p)
    # renvoie la liste des fils du sommet p dans l'arbre A
def pere(A, f)
    # renvoie le père du sommet f dans l'arbre A, ou None
    # si f est la racine
def racine(A, p)
    # renvoie la racine de l'arbre A
def etiquette(A, f):
    # renvoie l'etiquette de f dans A
```

Exercice 2

Écrivez un algorithme `taille_arbre(A,p)` qui prends un arbre A et qui renvoie le nombre de sommets de A.

Exercice 3

Écrivez un algorithme `parcours_arbre(A,p)` qui prends un arbre A et une liste p, qui parcours l'arbre A et remplie la liste p avec les sommets de l'arbre.

Exercice 4

Écrivez l'algorithme `parcours_niveau(A,h,p)` qui qui prends un arbre A et une hauteur h, qui parcours l'arbre A et qui remplie la liste p avec les sommets de l'arbre situés à hauteur h.

Exercice 5

Écrivez l'algorithme `sommet_a_distance(A,s,h)` qui renvoie la liste des sommets à profondeur h du sommet s dans A.

Exercice 6

Écrivez l'algorithme `parcours_feuilles(A,p)` qui qui prends un arbre A qui parcours l'arbre A et qui remplie la liste p avec les sommets de l'arbre qui sont des feuilles.

Exercice 7

Écrivez l'algorithme `parcours_sommets_internes(A,p)` qui qui prends un arbre A qui parcours l'arbre A et qui remplie la liste p avec les sommets de l'arbre qui sont des somets internes.

Exercice 8

Écrivez une fonction `afficher_arbre(A)` qui affiche l'arbre A à l'écran. Pour cela, vous allez utiliser Graphviz (voir www.graphviz.org) qui est un outil de visualisation des graphes. Pour ce faire, votre fonction convertira l'arbre en un fichier utilisant le langage DOT, il lancera le visualisateur dotty pour faire apparaître à l'écran l'arbre présent dans le fichier précédement créé.

Voici un exemple d'arbre écrit au format DOT :

```
digraph G{
  graph [ordering="out"];
  n1 [ label="2" ];
  n2 [ label="2" ];
  n3 [ label="-2" ];
  n4 [ label="2" ];
  n5 [ label="3" ];
  n6 [ label="2" ];

  n1 -> n2;
  n1 -> n3;
  n1 -> n4;
  n2 -> n5;
  n3 -> n6;
  n5 -> n7;
  n5 -> n8;
  n6 -> n9;
}
```

La ligne `1 -> 2` signifie que le sommet 1 est le père de 2.

2 Les arbres binaires

Exercice 9

Implémentez les arbres binaires étiquetés. Pour cela, vous implémenterez les fonctions suivantes :

```
def racine(A, p)
    # renvoie la racine de l'arbre A
def fils(A,p)
    # renvoie une liste des fils de A
def pere(A,p)
    # renvoie le père du sommet p dans l'arbre A ou None
    # si c'est la racine
def fils_gauche(A, p)
```

```
    # renvoie le fils gauche du sommet p dans l'arbre A
def fils_droit(A, p)
    # renvoie le fils droit du sommet p dans l'arbre A
```

Exercice 10

Écrivez les algorithmes `parcours_infixe(B,p)`, `parcours_prefixe(B,p)`, `parcours_postfixe(B,p)` qui prends un arbre binaire B et une liste p, qui parcours l'arbre B selon respectivement le parcours infixe, préfixe et postfixe et qui remplie la liste p avec les sommets de l'arbre dans l'ordre du parcours de l'arbre.

Exercice 11

Écrivez une fonction `afficher_arbre_binaire(A)` qui affiche l'arbre A à l'écran. Pour cela, quand un sommet n'a pas de fils gauche (resp. fils droit) vous dessinerez un sommet contenant un point à la place.

Voici un exemple d'arbre binaire écrit au format DOT :

```
digraph G{
  graph [ordering="out"];

  n1 [label="2"];
  n2 [label="5"];
  n3 [label="5"];
  n4 [label="3"];
  n5 [label="."];
  n6 [label="."];
  n7 [label="2"];
  n8 [label="."];
  n9 [label="."];
  n10 [label="."];
  n11 [label="."];

  n1 -> n2;
  n1 -> n3;
  n2 -> n4;
  n2 -> n5;
  n3 -> n6;
  n3 -> n7;
  n4 -> n8;
  n4 -> n9;
  n7 -> n10;
  n7 -> n11;
}
```

Exercice 12

Écrivez un algorithme `est_parfait(B)` qui prends un arbre binaire B et qui renvoie vrai si, tous les niveaux sont remplis sauf éventuellement le dernier niveau où toutes les feuilles sont tassées à gauches.

Exercice 13

Écrivez un algorithme `est_complet(B)` si toutes les feuilles de l'arbre de l'arbre binaire B sont à la même hauteur.

Exercice 14: Difficile

Écrivez un algorithme `arbres_binaires_de_taille_fixe(n)` qui renvoie la liste de tous les arbres binaires de taille n.

3 Les arbres binaires de recherche

Exercice 15

Écrivez un algorithme `est_arbre_binaire_de_recherche(B)` qui prends un arbre binaire B étiqueté et qui renvoie vraie si c'est un arbre binaire de recherche.

Exercice 16

Écrivez un algorithme `inserer_element_dans_arbre_binaire_de_recherche(B,e)` qui prends en paramètre un arbre binaire de recherche B et qui insère l'élément e dans l'arbre B.

Exercice 17

Écrivez un algorithme `creer_arbre_binaire_de_recherche(l)` qui prends en paramètre une liste d'éléments et qui construit un arbre binaire de recherche ayant pour étiquettes les éléments de la liste l.

Exercice 18

Écrivez un algorithme `rechercher_element(B, e)` qui prends un arbre binaire de recherche B et qui renvoie vrai si l'arbre contient une étiquette égale à e.

4 Les tas

Exercice 19

Écrivez un algorithme `est_un_tas(B)` qui prends un arbre binaire B étiqueté et qui renvoie vraie si c'est un tas. C'est à dire si B est un arbre parfait et que les étiquettes des pères sont toujours plus petites que celles de leurs fils.