DM – Arbres et graphes – Master 1 Bio-informatique

2020 – version 2

Exercice 1

Dans la définition des arbres, les fonctions pere(), fils() et racine() vérifient 4 propriétés. Pour chacune de ces propriétés p, donnez des exemples où pere(), fils() et racine() vérifient les 4 propriétés à l'exception de la propriété p.

Exercice 2

On suppose que vous disposez d'un module python implémentant les arbres binaires qui contient les fonctions suivantes :

```
def racine(A)
    # renvoie la racine de l'arbre 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
    # ou None si il n'a pas de fils gauche.
def fils_droit(A, p)
    # renvoie le fils droit du sommet p dans l'arbre A
    # ou None si il n'a pas de fils droit.
```

1. Écrivez une fonction `nombe_fils_gauches( arbre )` qui prend en paramètre un arbre et qui renvoie le nombre de sommets de l'arbre qui sont des fils gauches ;
2. Faites de même pour compter le nombre de fils droit ;
3. On dit qu'un arbre est équilibré si, pour chaque sommet s, la différence entre la hauteur du sous-arbre droit de s et la hauteur du sous-arbre gauche de s vaut -1, 0 ou 1. Écrivez, en python, la fonction `est_equilibre(arbre)` qui renvoie `True` si l'arbre passé en paramètre est équilibré et `False` sinon.

Exercice 3: Le parcours de l'ivrogne

L'objectif de cet exercice est d'implémenter un nouveau parcours, appelé le parcours de l'ivrogne. Soit A l'arbre binaire dont on cherche à obtenir un parcours de l'ivrogne. Avant d'implémenter ce parcours, il est nécessaire d'implémenter deux procédures intermédiaires :

1. Implémentez une fonction ~~descenteBouree(A, s)~~ descenteBouree(A, s, d) qui prend en paramètre un arbre A, un sommet s et une direction d ('gauche' ou 'droite'), qui descend dans l'arbre, tant que c'est possible, en commençant par l'arc de direction d (le fils d) et en alternant à chaque étape entre fils gauche et fils droit. L'algorithme s'arrête dès qu'il

ne peut plus descendre en suivant cette logique. La fonction renvoie alors le sommet d'arrivée et la direction ~~du dernier arc parcouru~~finale qui a forcé l'arrêt de la descente bourrée.

2. Implémentez la fonction repriseDeConscience(A, s, d) qui prend en paramètre un arbre, un sommet et une direction et qui remonte l'arbre à partir du sommet s à l'aide de la relation pere, tant que l'arc utilisé pour remonter est un arc de direction d. La fonction renvoie alors le sommet d'~~arrivé~~ arrivée et la direction courante.

Le parcours de l'ivrogne est un parcours qui commence avec la racine et une direction d donnée. Ensuite, il itère, jusqu'à avoir parcouru tous les sommets de l'arbre, les quatre opérations suivantes :

a) on commence évidement par changer de direction ;

b) on réalise ensuite une descente bourrée à partir du sommet courant et de la nouvelle direction ;

c) en conservant la dernière direction obtenu, on fait une reprise de conscience.

d) enfin, on note dans le parcours de l'ivrogne le sommet obtenu.

Par exemple, l'exécution du parcours de l'ivrogne, à partir du sommet 4 et de la direction 'droite' sur l'arbre A de la figure 1 donne le résultat suivant :

```
parcoursDeLivrogne(A, 'droite') :
  sommet de départ : racine(A) = 4
  direction de départ : 'droite'
  ----------------------------
  changement de direction : 'gauche'
  descenteBouree(A, 4, 'gauche') -> 3, 'droite'
     └ (4, 'gauche') - (3, 'droite')
  repriseDeConscience(A, 3, 'droite') -> 3
     └ 3
  on note 3
  ----------------------------
  changement de direction : 'gauche'
  descenteBouree(A, 3, 'gauche') -> 3, 'gauche'
     └ (3, 'gauche')
  repriseDeConscience(A, 3, 'gauche') -> 4
     └ 3 - 4
  on note 4
  ----------------------------
  changement de direction : 'droite'
  descenteBouree(A, 4, 'droite') -> 7, 'droite'
     └ (4, 'droite') - (8, 'gauche') - (5, 'droite') -
       (1, 'gauche') - (7, 'droite')
  repriseDeConscience(A, 7, 'droite') -> 7
     └ 7
  on note 7
  ----------------------------
```

```
changement de direction : 'gauche'
descenteBouree(A, 7, 'gauche') -> 7, 'gauche'
   └ (7, 'gauche')
repriseDeConscience(A, 7, 'gauche') -> 1
   └ 7 - 1
on note 1
-----------------------------
changement de direction : 'droite'
descenteBouree(A, 1, 'droite') -> 2, 'gauche'
   └ (1, 'droite') - (2, gauche)
repriseDeConscience(A, 2, 'gauche') -> 2
   └ 2
on note 2
-----------------------------
changement de direction : 'droite'
descenteBouree(A, 2, 'droite') -> 2, 'droite'
   └ (2, 'droite')
repriseDeConscience(A, 2, 'droite') -> 5
   └ 2 - 1 - 5
on note 5
-----------------------------
changement de direction : 'gauche'
descenteBouree(A, 5, 'gauche') -> 5, 'gauche'
   └ (5, 'gauche')
repriseDeConscience(A, 5, 'gauche') -> 8
   └ 5 - 8
on note 8
-----------------------------
Tous les sommets ont été visités, fin du parcours.
```

Le résultat du parcours de l'ivrogne est donc : [3, 4, 7, 1, 2, 5, 8].
Vous êtes fin prêt pour expérimenter le parcours de l'ivrogne !

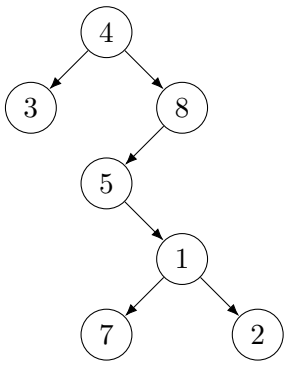

FIGURE 1 – Un arbre binaire

4. Implémentez le parcours de l'ivrogne.

5. (question optionnelle) Est-ce que le parcours de l'ivrogne passe par tous les sommets ? Justifiez votre réponse.

Exercice 4

1. Choisissez une structure de donnée pour implémenter des graphes non orientés. Proposez ensuite une implémentation des graphes en python.
2. Proposez un algorithme qui prend en paramètre un graphe non orienté et qui renvoie le nombre de composantes connexes du graphes.
3. Proposer un algorithme qui détermine si le graphe passé en paramètre contient un cycle.