NSI - Terminale

Structures de données - Graphes - TD : algorithmes

qkzk

2020/10/11

Compétence : *Savoir parcourir un graphe*

Exercice 1

À partir de la classe à la fin de l'exercice, on donne ci-dessous la représentation informatique de l'arbre `a`. On rappelle que la création d'un arbre crée simplement un noeud racine dont la clé est définie en paramètre et dont les sous arbres gauches et droits sont vides

```
a = Arbre(37)
a.sag = Arbre(41)
a.sag.sag = Arbre(13)
a.sag.sag.sad = Arbre(3)
a.sag.sag.sad.sag = Arbre(5)
a.sag.sag.sad.sad = Arbre(23)
a.sad = Arbre(2)
a.sad.sag = Arbre(7)
a.sad.sad = Arbre(11)
a.sad.sad.sag = Arbre(19)
```

```
class Arbre:
    def __init__(cle):
        self.cle = cle
        self.sag = None
        self.sad = None
```

1. Représenter graphiquement cet arbre `a`.
2. Donner le résultat du parcours de cet arbre en largeur d'abord.
3. Donner le résultat du parcours en préfixe, en infixe, en suffixe.

Exercice 2

Compétence : *Savoir rechercher et insérer une clé dans un arbre binaire de recherche.*

Toujours à partir de la classe présentée plus haut, on suppose cette fois disposer d'une méthode `inserer_cle` qui prend un paramètre entier en entrée et insère la clé dans l'arbre en respectant la propriété "l'arbre est un arbre binaire de recherche."

On dispose aussi d'une méthode `rechercher_cle` qui prend en paramètre d'entrée une clé entière et retourne un booléen Vrai si, et seulement si, la clé est présente dans l'arbre.

On donne ci-dessous la représentation informatique de l'arbre binaire de recherche `abr` :

```
liste_cles = [3, 10, 1, 6, 14, 4, 7, 4, 13]

abr = Arbre(8)
for cle in liste_cles:
    if not abr.rechercher_cle(cle):
        abr.inserer_cle(cle)
```

1. Représenter graphiquement cet arbre binaire de rechercher `abr`.
2. Donner le résultat du parcours de cet arbre en infixe. Que remarque-t-on ?

Compétence : *Savoir parcourir un graphe*

Exercice 3

On donne ci-dessous la représentation informatique du graphe *non orienté* `g` :

```
g = Graphe()
liste_cle = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
for cle in liste_cle:
    g.ajouter_sommet(cle)

g.ajouter_arete('A', 'B')
g.ajouter_arete('B', 'A')
g.ajouter_arete('B', 'C')
g.ajouter_arete('B', 'D')
g.ajouter_arete('B', 'G')
g.ajouter_arete('C', 'B')
g.ajouter_arete('C', 'E')
g.ajouter_arete('D', 'B')
g.ajouter_arete('D', 'i')
g.ajouter_arete('E', 'C')
g.ajouter_arete('E', 'I')
g.ajouter_arete('F', 'A')
g.ajouter_arete('F', 'G')
g.ajouter_arete('F', 'H')
g.ajouter_arete('G', 'B')
g.ajouter_arete('G', 'F')
g.ajouter_arete('G', 'I')
g.ajouter_arete('H', 'F')
g.ajouter_arete('H', 'I')
g.ajouter_arete('I', 'D')
g.ajouter_arete('I', 'E')
g.ajouter_arete('I', 'G')
g.ajouter_arete('I', 'H')
```

1. Représenter graphiquement ce graphe.
2. Donner la liste d'adjacence correspondante.
3. Donner le résultat du parcours de ce graphe en largeur d'abord à partir du sommet de la clé `'B'`.
4. Donner le résultat du parcours de ce graphe en profondeur d'abord à partir du sommet `'C'`.

Compétence : *Savoir détecter un cycle dans un graphe. Savoir rechercher un chemin dans un graphe.*

Exercice 4

Reprenons le graphe défini dans l'exercice précédent.

1. Ce graphe possède-t-il au moins un cycle ? Si on applique l'algorithme de recherche de cycle au départ du sommet `A` quel est le sommet qui en permet l'arrêt ?
2. Donner une chaîne de "trajet" entre les sommets de clé `'E'` et `'H'` en appliquant l'algorithme de recherche vu en cours.