TP - Graphes et Langages

6 septembre 2016

L'objectif de ces TP est de programmer des fonctions classiques sur les graphes finis orientés.

1 Graphes en Python

1.1 Les dictionnaires en Python

En Python, un dictionnaire est un tableau associatif qui a chaque clé du tableau associe une valeur. Un dictionnaire se déclare ainsi :
`{cle1:valeur1,cle2,valeur2,.... }`
Par exemple, le dictionnaire `D=1:2, ''titi'':3` a deux clés qui sont `1` et `''titi''`. La valeur `2` est associée à `1` et la valeur `3` est associée à la clé `''titi''`. Les valeurs peuvent être de n'importe quel type, mais il y a des restrictions sur les clés possibles (qu'on ne décrira pas ici, mais une liste classique par exemple ne peut pas être une clé).

L'accès à une donnée se fait par des `[]`, comme pour une liste. On a, par exemple ici, `D[1]` qui vaut `2`.

Pour avoir la liste des clés on utilise ma méthode `keys()`, ici par exemple `D.keys()` et pour avoir la liste des valeurs `D.values()`. Vous pourrez avoir de nombreuses informations sur internet, par exemple sur
`https://openclassrooms.com/courses/apprenez-a-programmer-en-python/les-dictionnaires-2`

1. Créer un dictionnaire dont l'ensemble des clés est `1,2,4,6,7,9,12,15,18,20` et la valeur de chaque clé est la liste de tous ses diviseurs premier positifs.
2. Afficher la liste des clés.
3. Afficher la liste des valeurs.
4. On rappelle qu'un nombre est premier s'il a exactement deux diviseurs (qui sont forcément 1 et lui-même). Ecrire une fonction qui à partir d'une liste de diviseurs, teste si le nombre correspondant est premier.
5. Écrire une fonction qui retourne la liste des clé du dictionnaire qui sont premiers.

1.2 Les graphes par listes d'adjacence

On propose d'utiliser qu'un graphe soit un tuple de longueur 2, (V, E), où V est une liste triée de sommets (des entiers pour plus de simplicité) et E est un dictionnaire, qui à chaque élément de V associe la liste triée de ses voisins.

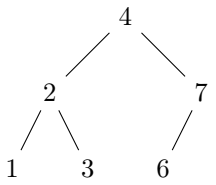
Par exemple, le codage `([1,2,5,6], {1:[],2:[1,2,6],5:[6],6:[1,5]})` représente le graphe dont l'ensemble des sommets est $\{1, 2, 5, 6\}$ et l'ensemble des arêtes est $\{(2, 1), (2, 2), (2, 6), (5, 6), (6, 1), (6, 5)\}$.

1. Écrire une fonction affiche en mode texte un graphe. exemple.
2. Écrire une fonction qui calcule le nombre de sommet d'un graphe.
3. Écrire une fonction qui calcule le nombre d'arête d'un graphe.
4. Écrire une fonction qui supprime une arête à un graphe.
5. Écrire une fonction qui ajouter une arête à un graphe.
6. Écrire une fonction qui transforme un graphe en sa structure en `Networkx`.
7. En utilisant la fonction précédente, écrire une fonction qui permet d'afficher un dessin d'un graphe.
8. Programmer l'algorithme de parcours en largeur.
9. Programmer l'algorithme de parcours en profondeur.
10. En utilisant/modifiant le parcours en largeur, programmer une fonction qui étant donnés deux sommets x et y d'un graphe donne, s'il existe, un chemin le plus court possible allant de x à y.

2 Arbres binaires de recherche

On va maintenant coder les arbres binaires de recherche. Par simplicité, on n'accepte pas les doublons dans les ABR.

Afin de bénéficier de l'aspect récursif, les ABR sont codé de la façon suivante, illustré sur la figure 1 :



`[4,[2,[[1,[],[]],[3,[],[]]],[7,[[6,[],[]]],[]]]`

FIGURE 1 – Exemple de codage d'un ABR.

— L'arbre vide est codé par la liste vide `[]`,
— Un arbre non vide est codé par `[valeur,SousArbreGauche,SousArbreDroit]`.

1. Dessiner l'arbre codé par `[10,[8,[],[]],[12,[],[]]]`.
2. Dessiner l'arbre codé par `[10,[8,[],[[9,[],[]]],[12,[[11,[],[]],[]]]]`.
3. Dessiner l'ABR obtenu en insérant, dans l'ordre, 8,4,5,2,12,17. Donner son codage.
4. Coder l'insertion dans un ABR.
5. Coder la recherche d'un élément dans un ABR.
6. Coder une fonction qui calcule la hauteur d'un ABR.
7. Coder la fonction de suppression d'une feuille dans un ABR.
8. Coder une fonction qui supprime un sommet de l'arbre qui n'a qu'un fils.
9. Coder une fonction qui supprime la valeur à la racine d'un ABR.
10. Coder la fonction de suppression dans un ABR.