Université Bordeaux 1
Licence, module J1MI1003, 2013/2014

TP Noté

Listes-Graphes-Chaînes

Tous documents autorisés. Lundi 9 Décembre 2013
Durée : 1h 20.
Vous devez écrire vos solutions dans un fichier tpnote.py et envoyer ce fichier à l'adresse : `ges@labri.fr` avec comme sujet "TPNOTE".

Partie 1 : listes.

1- Ecrire, en Python, une fonction `lunion(l,n)` qui retourne une liste comportant, dans un ordre quelconque, les éléments apparaissant au moins une fois dans `l` ou dans `n`. Lorsque `l` et `n` sont sans répétition, le résultat retourné doit ne pas avoir de répétition non-plus.
Par exemple, on doit avoir

```
>>> l=lunion([1,2,3,4],[2,3,4,5])
>>> print(l)
[1, 2, 3, 4, 5]
```

2- Ecrire une fonction `inclusion(l,n)` qui prend en argument deux listes `l`, `n` et :
renvoie `True` si l' ensemble des éléments de la liste `l` est inclus dans l' ensemble des éléments de la liste `n`.
renvoie `False` sinon.
Par exemple :

```
>>> inclusion([1,2,3],[1,3,4])
False
>>> inclusion([1,2,3],[3,4,1,2])
True
```

3- Ecrire une fonction `setequal(l,n)` qui prend en argument deux listes `l`, `n` et :
renvoie `True` si l' ensemble des éléments de la liste `l` est égal à l' ensemble des éléments de la liste `n`.
renvoie `False` sinon.
4- Ecrire une fonction `inclusionstrict(l,n)` qui prend en argument deux listes `l`, `n` et :
renvoie `True` si l' ensemble des éléments de la liste `l` est strictement inclus dans l' ensemble des éléments de la liste `n`.
renvoie `False` sinon.
Par exemple :

```
>>>inclusionstrict([1,3,4],[4,2,1,3])
True
>>>inclusionstrict([1,2,3,4],[4,2,1,3])
False
```

Partie 2 :connexité-distance

On cherche à calculer la composante connexe d'un sommet s d' un graphe par une variante de la méthode vue en TP : au lieu de propager des "marques" depuis s tant qu'on le peut, on ajoute des sommets du graphe dans une liste, tant qu'on le peut. On obtient la fonction suivante.

```
def compconnexe(s):
    lcour=[s]
    lnext=lunion(lcour,listeVoisins(s))
    while inclusionstrict(lcour,lnext):
        lcour=lnext
        for u in lcour:
            lnext=lunion(lnext,listeVoisins(u))
    return lcour
```

5- Tester cette fonction sur le graphe `tgv2005`.
6- On appelle S-chaîne dans le graphe G, une suite de sommets $(s_0, s_1, \ldots, s_k, s_{k+1}, \ldots, s_n)$ telle que, pour tout $k \leq n-1$, s_k et s_{k+1} sont voisins. La *longueur* de la S-chaîne est l'entier n. La *distance* entre un sommet s et un sommet t d'un graphe G est la longueur de la *plus courte* S-chaîne de s à t (et ∞ si s et t ne sont connectés par aucune chaîne).
Ecrire une fonction `distance(s,t)` qui prend en argument deux sommets `s`, `t` et renvoie la distance de `s` à `t` (ou alors `"infty"` si ces sommets ne sont pas connectés). Par exemple :

```
>>> distance(sommetNom(tgv2005,"Bordeaux"),sommetNom(tgv2005,"Marseille"))
2
>>> distance(sommetNom(tgv2005,"Bordeaux"),sommetNom(tgv2005,"Strasbourg"))
'infty'
>>> distance(sommetNom(fig22,"B"),sommetNom(fig22,"D"))
3
```

Aide : on pourra ajouter, dans le texte de la fonction `compconnexe(s)`, un *compteur* du nombre de passages dans la boucle `while`.

Partie 3 : S-chaînes minimales

7- Ecrire une fonction `premiersom(s,t)` qui prend en argument deux sommets `s`, `t` et renvoie un sommet qui est le suivant de `s` dans une S-chaîne minimale de `s` à `t` (ou `None` si `s`, `t` ne sont pas connectés). Par exemple :

```
>>> premiersom(sommetNom(tgv2005,"Bordeaux"),sommetNom(tgv2005,"Marseille"))
<sommet: 'Lille', 'white', False>
>>> print( premiersom(sommetNom(tgv2005,"Bordeaux"),sommetNom(tgv2005,"Strasbourg")))
None
>>> premiersom(sommetNom(fig22,"B"),sommetNom(fig22,"D"))
<sommet: 'A', 'white', False>
```

8- Ecrire une fonction `Schainemin(s,t)` qui prend en argument deux sommets `s`, `t` et renvoie une S-chaîne minimale allant de `s` à `t` (ou `None` si `s`, `t` ne sont pas connectés). Par exemple :

```
>>> Schainemin(sommetNom(tgv2005,"Bordeaux"),sommetNom(tgv2005,"Marseille"))
[<sommet: 'Bordeaux', 'white', False>, <sommet: 'Lille', 'white', False>,
<sommet: 'Marseille', 'white', False>]
>>> Schainemin(sommetNom(fig22,"B"),sommetNom(fig22,"D"))
[<sommet: 'B', 'white', False>, <sommet: 'A', 'white', False>,
 <sommet: 'C', 'white', False>, <sommet: 'D', 'white', False>]
```

9- Ecrire une fonction `marquermin(s,t)`: qui marque tous les sommets d' une S-chaîne minimale allant du sommet `s` vers le sommet `t`.
10- Ecrire une fonction `testvisu(G,nom1,nom2)` qui prend en argument un graphe `G` et des chaînes de caractères `nom1`, `nom2` et dessine sur votre écran le graphe `G` avec des marques sur tous les sommets d'une S-chaîne minimale allant du sommet de nom `nom1` vers le sommet de nom `nom2` . Tester sur les exemples :

```
testvisu(tgv2005,"Bordeaux","Marseille")
testvisu(fig22,"B","D")
```