Travaux pratique énoncé 2 : clôture transitive

Similairement au sujet précédent, on gardera ouvert deux documents :

anneau02.py

relation.py

Dans le second document, vous avez déjà *composition()* et *union*.

L'objet d'étude de ce TP est une carte représentée par un graphe. S'il existe un chemin entre deux point, l'arête est pondérée par la distance entre ces points.Sinon, l'arête est pondérée par $+\infty$.

Anneau min-plus

Dans un premier temps, codons les fonctions de bases de l'anneau. On utilise l'anneau min plus : sur l'ensemble $\mathbb{N} \cup \{\infty\}$, avec l'opération somme étant le minimum, d'élément neutre $+\infty$ et l'opération produit étant la somme classique, d'élément neutre 0. On constate qu'on a bien

$$\forall a, b, c,\ x + \min(y, z) = \min(x + y, x + z)$$

On notera ∞ par -1.
En code, dans anneau02.py, ajouter:

```
def unit():
    return -1

def produit(a,b):
    if (a==-1 or b==-1)
        return -1
    else
        return a+b
```

```
def zero():
    return 0

def somme(a,b):
    if (a==-1 and b==-1)
        return -1
    else
        return min(a,b)
```

Pourrions nous inverser le rôle des opérations sommes et produit (en changeant également unit et zero) ?

Réutiliser la fonction *affiche* du Tp1 en la modifiant pour remplacer les nombres négatifs ("-1") par des espaces(" "). Elle prend désormais en entrée une relation pondérée (liste de liste d'entiers), elle ne retourne toujours pas de sortie et elle imprime un tableau.
Vous pourrez vous référer à celle que vous avez écrite pour le TP1, en n'oubliant pas de modifier le contenu des print().

L' entrée

```
[[−1, 1 , 1 , 1 ,−1],
 [ 6 ,−1, 6 ,−1,−1],
 [−1,−1, 4 ,−1, 0 ],
 [−1,−1,−1, 7 ,−1]]
```

Devrait donner

```
111
6 6
 4 0
 7
```

La somme de deux élément consiste en une comparaison de deux chemins parallèles et le choix du plus court. Le produit consiste a prendre un chemin en deux parties, et calculer la distance totale.

Opérations sur relations pondérées

Ordre sur les relations On considère l'ordre suivant : R est plus petite que S si

- toute arête de R appartient aussi à S
- Pour toute arc de R, son coefficient est plus grand que son équivalent dans S.

Par exemple, R:=[[-1,0,1][2,7,5]] est plus petite que S:=[[-1,0,0][2,5,2][-1,1,18]], En effet : les liaisons de la première sont :

- (0,1) de valeur 0 et (1,0), de valeur 2 apparaissent aussi avec les même valeurs pour la seconde relation.
- (0,2) de valeur 1 , (1,1) de valeur 7 et (1,2) de valeur 5 apparaissent aussi dans la seconde relation, et ceux de la première sont des entier plus *petit* que ceux de la seconde

[[ ]] est la plus petite relation.
$0^{N\times M}$ (tableau de taille $N \times M$ remplis par des 0) est la plus grande relation parmi celles de taille inférieures ou égales à $N \times M$

Symétrie *Une relation pondérée est symétrique si pour toute arête de la relation, l'arête inverse appartient également à la relation et possède le même coefficient.*

Écrire une fonction *testSymetrie(R)* qui prend en entrée une relation pondérée R et retourne un booléen indiquant si R est symétrique.
On se rappelle du TD2, parcourir les paires de sommets et tester la propriété sur chaque. Est il nécessaire de tester toutes les arêtes ?

Écrire une fonction *symetrie(R,somme)* qui prend en entrée une relation R et retourne la plus petite relation S symétrique plus grande que R.
On rappelle que Somme(a, b) retourne ici le plus grand entier inférieur à a et à b.
Deux manières sont possible : comparer chaque arête avec son inverse,

ou écrire une fonction *inverse(R)* qui prend en entrée une relation R et retourne R^{-1}, puis appeler symetrie(R, somme) := union(R, inverse(R), somme)

Facultatif : écrire une fonction *symetrie2(R)* qui prend en entrée une relation R et retourne la plus grande relation S symétrique plus petite que R. vous pourrez utiliser max et ferez attention aux infinis .

Réflexivité *Une relation est réflexive s'il existe une boucle de poids zero sur tout sommet*
Écrire une fonction testReflexivite(R) qui prend une relation en entrée et retourne un booléen vrai si la relation est réflexive et faux sinon. (Vous pouvez vous référer au TD).

Écrire une fonction reflexive(R) qui prend une fonction en entrée et retourne la plus petite fonction réflexive qui la contient.
Alternativement, écrire une fonction *identite(n)* prenant en argument une taille d'ensemble n et retournant la relation identité sur $[1, n]$, puis appeler reflexive(R, somme) = union(R,identite(len(R)),somme)

Tester les deux fonctions sur la relation vide de taille 3 ([[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]]).

Éxécuter testreflexive sur reflexive([[-1,-1,-1],[-1,-1,-1],[-1,-1,-1]]).

Génération aléatoire Importer random et initialiser la graine (random.seed()). *Vous savez que random.randint(a,b) retourne un nombre aléatoire uniforme compris entre a et b tous deux inclus*

Écrire une fonction mapgen(s,m=10,p=1,q=1) qui prend en entrée quatre entiers :

Le nombre de sommets, ie la taille de la relation

Une borne sur les distance

Deux entiers codant ensemble un rationnel $\frac{p}{q}$ est la proportion de distances non infinies ($\neq -1$)

et retourne une relation de l'ensemble $[0, s-1]$ dans lui même dont chaque élément a une probabilité $\frac{p}{q}$ d'être choisi uniformément dans $[0, m]$ et vaut ∞ sinon.

Tester et afficher votre fonctions.

Clôtures

La clôture réflexive et transitive d'une relation R est la plus petite relation qui contient R et est réflexive et transitive. Puisque la relation pleine (toute paire de sommet est indexée par 0) contient toute les autres et est réflexive et transitive, et que l'intersection de deux relations réflexives et transitive l'est également, une telle clôture existe toujours et est bien définie

Suite récurrente Récupérer ou recoder les fonctions Union et composition du TP précédent.

Vérifier la compatibilité avec l'anneau de ce TP.

Quel est la taille du plus long chemin simple (ne passe pas deux fois par un même sommet) possible dans un graphe?

Si vous avez un graphe G, que vous ajoutez à G les fusions d'arêtes, sans retirer celle consommées, pour obtenir $G^{(2)} = G.G \cup G$, $G^{(3)} = G^{(2)}.G \cup G \cdots$, et que vous répétez l'opération autant que la taille du plus long chemin, reste il des arêtes dont la fusion n'est pas déjà dans le graphe ? Que peut on en déduire de $G^{(n)}$?

Existe il un graphe transitif contenant G mais ne contenant pas $G^{(2)}$?
ne contenant pas $G^{(3)}$? $G^{(n)}$?
Qu'en déduis on pour $G^{(n)}$?

Utiliser union, reflexive et composition et la réponse aux question précédente pour écrire une fonction transcloture(R,union, intersection, reflexive) qui retourne la clôture réflexive et transitive de R.

La tester sur des graphes générés aléatoirement avec paramètres $10, 10, 1, 3$