NOM Prénom :

CC3 Informatique

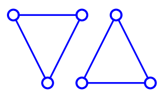
J1 MI 1003, groupe B3, Université Bordeaux

Attention recto et verso !

Exercice 1 (9 points)
Les proposition suivantes sont-elles vraies ou fausses ? Justifier.

1. Le graphe complet K_6 avec 6 sommets est Eulérien. Fait en cours ; l'erreur ici n'était pas excusable. Évidement c'est FAUX : K_6 ayant 6 sommets de degré 5, il ne peut être Eulérien. (Un graphe connexe est Eulérien ssi son nombre de sommets impairs est 0 ou 2.)

2. Un graphe dont le degré de chaque sommet est pair est forcément Eulérien. Pareil pour celui-ci, il a été fait en cours. C'est FAUX également, il ne faut pas oublier la connexité. Vous n'avez pas la totalité des points sans contre-exemple. En effet, d'un point de vue logique, si vous ne donnez pas de contre-exemple, rien ne dit qu'un graphe non connexe avec que des sommets de degré pair existe. Même si c'est évident.
   Exemple d'un graphe simple :

3. Un graphe Eulérien simple sans sommet de degré 1 a forcément des sommets de degré pair. (Question un peu dure)
   La question un peu dure de ce CC (mais ça va). C'était VRAI.
   En effet, considérons G un graphe Eulérien simple sans sommet de degré 1. Comme il est Eulérien, son nombre de sommets impairs est 0 ou 2. Si G a 0 sommet de degré impair, alors tous les autres sommets sont de degré pair (donc il en existe un).
   On peut donc supposer qu'il a 2 sommets impairs. Prenons un de ces sommets, il est forcément de degré supérieur à 3. Mais vu que le graphe est simple, cela veut dire que ce sommet a forcément 3 voisins distincts. Or deux sommets seulement sont impairs, donc un de ces sommets est de degré pair. Ce qui achève la preuve.

Exercice 2 (5 pts)
Ecrire une fonction Python `ExisteVoisinsMarques(G)` qui renvoie `True` s'il existe dans le graphe `G` deux sommets voisins marqués, `False` sinon.
Ce genre de programmes doit maintenant être maitrisé, ça fait depuis le début de l'année que je vous le répète !

```python
def ExisteVoisinsMarques(G):
    for s in listeSommets(G):
        for t in listeVoisins(s):
            if estMarqueSommet(s) and estMarqueSommet(t):
                return True
    return False
```

Tourner la page s.v.p.

Exercice 3 (6 pts)
Ecrire une fonction Python `admetUnCycleEulerien(G)` qui renvoie `True` si `G` admet un cycle Eulérien, `False` sinon. On supposera que `G` est connexe.
Alors là, grosse panique à bord, c'était un exercice pas si compliqué. En effet, l'unique théorème que vous deviez apprendre dit qu'un graphe connexe admet un cycle Eulérien ssi il n'a pas de sommet de degré impair. Du coup, la fonction était évidente :

```
def admetUnCycleEulerien(G):
    for s in listeSommets(G):
        if degre(s) % 2 == 1:
            return False
    return True
```

Le gros algorithme du cours qui cherche à calculer un cycle Eulérien ne marche que si on sait que le graphe admet un cycle Eulérien !! L'adaptation était donc quasiment impossible !

Annexe : voici un rappel des principales fonctions disponibles pour manipuler les graphes (à vous de voir celles qui sont utiles pour ce devoir).

L'argument G est un graphe	
`listeSommets(G)`	retourne la *liste* des *sommets* de G.
`nbSommets(G)`	retourne le *nombre* de *sommets* de G.
`sommetNom(G,etiquette)`	retourne le *sommet* de G désigné par son *nom* (étiquette). Exemple : `sommetNom (Europe, 'Italie')`.
`sommetNumero(G,i)`	retourne le *sommet* numéro i dans G ; la numérotation commence à 0.

L'argument s est un sommet	
`listeVoisins(s)`	retourne la *liste* des *voisins* de s.
`degre(s)`	retourne le *degré* de s.
`nomSommet(s)`	retourne le *nom* (étiquette) de s.
`colorierSommet(s,c)`	colorie s avec la couleur c. Exemples de couleurs : `'red'`, `'green'`, `'blue'`, `None`.
`couleurSommet(s)`	retourne la *couleur* de s.
`marquerSommet(s)` `demarquerSommet(s)`	marque ou démarque s.
`estMarqueSommet(s)`	retourne `True` si s est marqué, `False` sinon.