

Numéro d'anonymat :

Groupe :

Remplir l'en-tête de la copie, cacheter, et ne rien écrire d'autre sur la copie ; porter les réponses directement sur le sujet, et glisser celui-ci dans la copie en fin d'épreuve. *Ne pas oublier d'indiquer ci-dessus le numéro d'anonymat et le groupe.*

Le sujet comporte 5 exercices et 7 pages, plus une page d'annexe. Aucun document n'est autorisé : les principales fonctions de manipulation des graphes sont rappelées sur la page 9, que vous pouvez détacher pour plus de facilité.

Exercice 1 On considère la fonction suivante :

```
def f(a,x):  
    p = 1  
    n = 0  
    while p <= x:  
        p = p * a  
        n = n + 1  
    return n
```

1. Simuler l'exécution de $f(10,1234)$ en complétant le tableau suivant :

p	1
n	0

2. En général, que vaut $f(10,x)$?

3. On exécute le code suivant :

```
>>> for x in range(11):  
    print (x, f(2,x))
```

Donner le résultat ci-contre (écrire sur deux colonnes si nécessaire) :

4. À l'aide de la fonction f , écrire une fonction $g(n)$ qui calcule le nombre de chiffres de la représentation décimale de la factorielle $n!$

Exercice 2 Dans cet exercice $g5$ désigne le graphe à 5 sommets tel que :

```
>>> for s in listeSommets(g5):  
    listeAretesIncidentes(s)
```

```
[<arête: 'e4' D--A>, <arête: 'e1' A--B>]  
[<arête: 'e1' A--B>, <arête: 'e2' B--C>, <arête: 'e7' D--B>, <arête: 'e8' B--C>]  
[<arête: 'e8' B--C>, <arête: 'e5' C--E>, <arête: 'e2' B--C>, <arête: 'e3' C--D>]  
[<arête: 'e3' C--D>, <arête: 'e4' D--A>, <arête: 'e6' E--D>, <arête: 'e7' D--B>]  
[<arête: 'e5' C--E>, <arête: 'e6' E--D>]
```

1. Dessiner ce graphe. Est-ce un graphe planaire ?

2. Ce graphe est-il simple ? Justifier la réponse :

3. Ce graphe est-il Eulérien ? Justifier la réponse :

On dispose des fonctions suivantes :

```
def areteIncidenteNonMarquee(s):
    for a in listeAretesIncidentes(s):
        if not estMarqueeArete(a):
            return a
    return None

def cycleSansIssue(s):
    c = [s]
    a = areteIncidenteNonMarquee(s)
    while a != None:
        marquerArete(a)
        s = sommetVoisin(s, a)
        c = c + [a, s]
        a = areteIncidenteNonMarquee(s)
    return c
```

En réponse à chacune des questions qui suivent on décrira comme d'habitude un cycle par une liste constituée alternativement de sommets et d'arêtes, par exemple :

$[C, e_5, E, e_6, D, e_3, C]$

4. Aucune arête n'est marquée et on calcule :

$c_1 = \text{cycleSansIssue}(\text{sommetNom}(g5, 'A'))$

Donner la valeur de c_1 :

$c_1 = [\quad \quad \quad]$

Attention : il y a une seule réponse correcte, effectuer le calcul avec soin, en utilisant les informations données en début d'exercice.

5. Sans démarquer les arêtes marquées lors du calcul précédent, on calcule maintenant :

$c_2 = \text{cycleSansIssue}(\text{sommetNom}(g5, 'B'))$

Donner la valeur de c_2 :

$c_2 = [\quad \quad \quad]$

6. Donner le cycle Eulérien c_3 qui résulte des deux calculs précédents (attention, donner un cycle eulérien quelconque ne rapportera aucun point) :

$c_3 = [\quad \quad \quad \quad \quad \quad \quad \quad \quad]$

7. On démarque toutes les arêtes et on calcule à nouveau :

$c_4 = \text{cycleSansIssue}(\text{sommetNom}(g5, 'B'))$

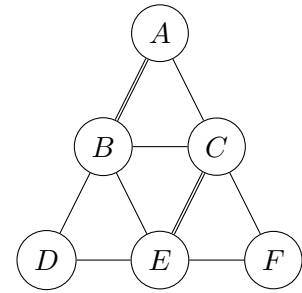
Donner la valeur de c_4 :

$c_4 = [\quad \quad \quad \quad \quad \quad \quad \quad \quad]$

Exercice 3

Les arêtes marquées d'un graphe G forment un *couplage* si elles sont deux à deux *disjointes*; autrement dit chaque sommet est l'extrémité d'au plus une arête marquée. Un sommet est dit *saturé* s'il possède une arête incidente marquée.

Par exemple dans le graphe T ci-contre les arêtes marquées AB et CE forment un couplage, et les sommets saturés sont A, B, C, E .

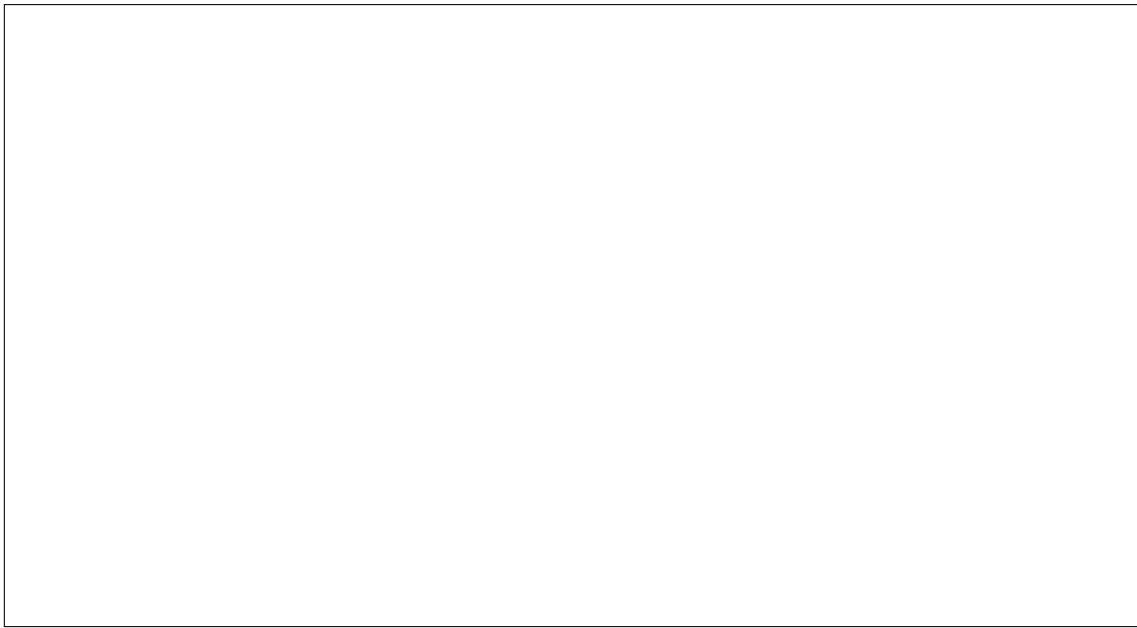


1. Dessiner un couplage du graphe T avec trois arêtes marquées :

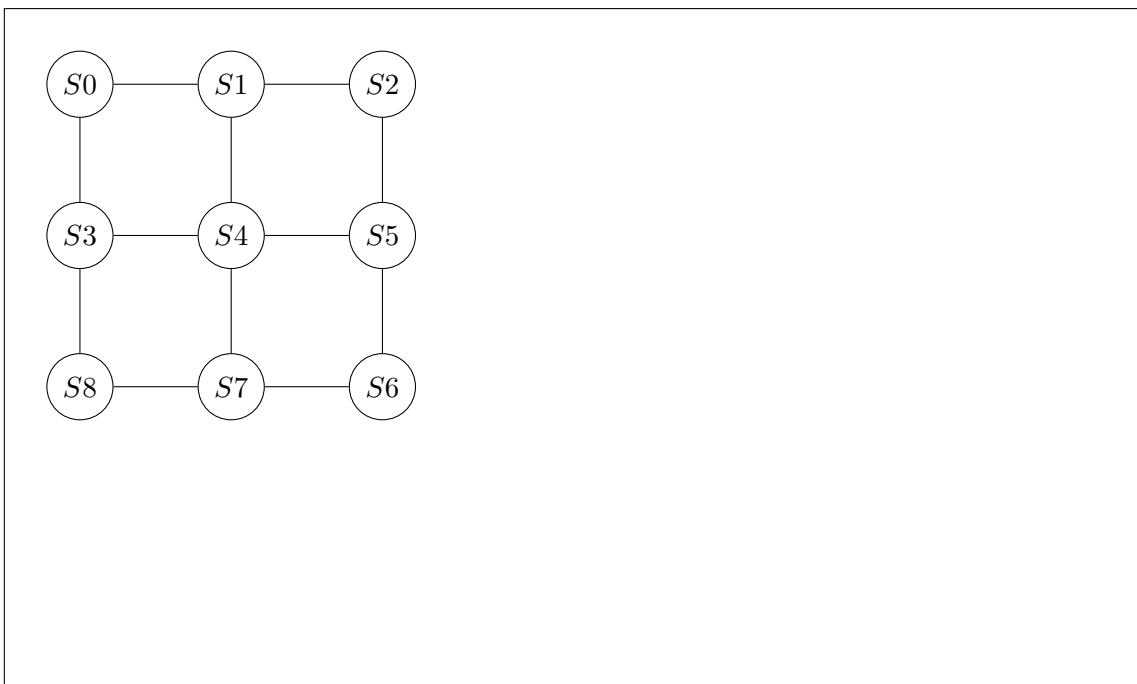
2. Écrire une fonction `sature(s)` qui retourne `True` ou `False` selon que le sommet s est saturé ou non :

3. Une arête est dite *libre* si aucune de ses extrémités n'est saturée. On reprend l'exemple donné en début d'exercice (les arêtes marquées sont AB et CE) : existe-t-il des arêtes libres dans T ? Justifier la réponse.

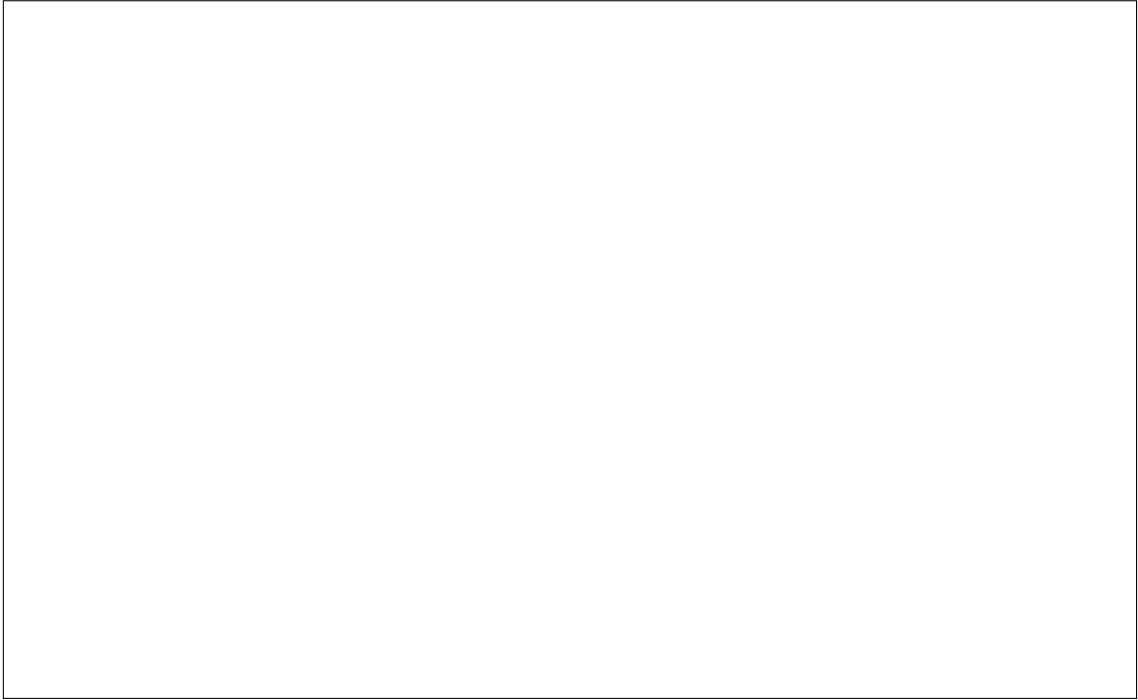
4. Écrire une fonction `areteLibre(s)` qui retourne une arête libre issue du sommet s s'il en existe une, et `None` sinon. *Note* : si s est saturé, la fonction retourne donc `None`.



5. On construit un couplage en appliquant l'algorithme suivant : pour chaque sommet s du graphe G on marque une arête libre issue de s s'il en existe une. Appliquer cet algorithme à la grille ci-dessous, en supposant qu'on parcourt les sommets dans l'ordre de leurs numéros, et qu'on choisit l'arête libre qui va vers le sommet de plus petit numéro. *Attention* : détailler à côté du dessin l'action effectuée pour chacun des neuf sommets :



6. Écrire une fonction `construireCouplage(G)` qui réalise l'algorithme précédent (on suppose qu'au départ aucune arête de G n'est marquée), et retourne la liste des arêtes marquées :

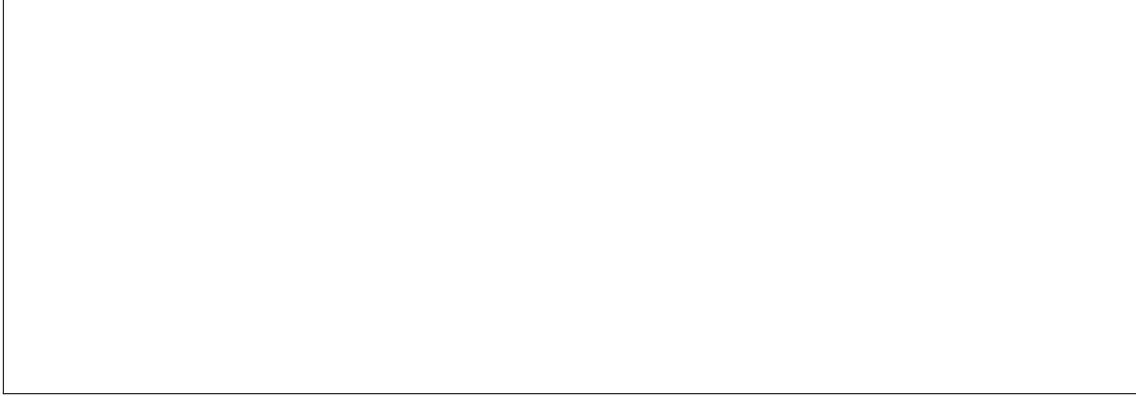


Exercice 4 Dans un graphe bien colorié tous les sommets sont coloriés, et les extrémités de n'importe quelle arête sont de couleurs différentes. Écrire une fonction `bienColorie(G)` qui retourne *True* si le graphe G est bien colorié, et *False* sinon.

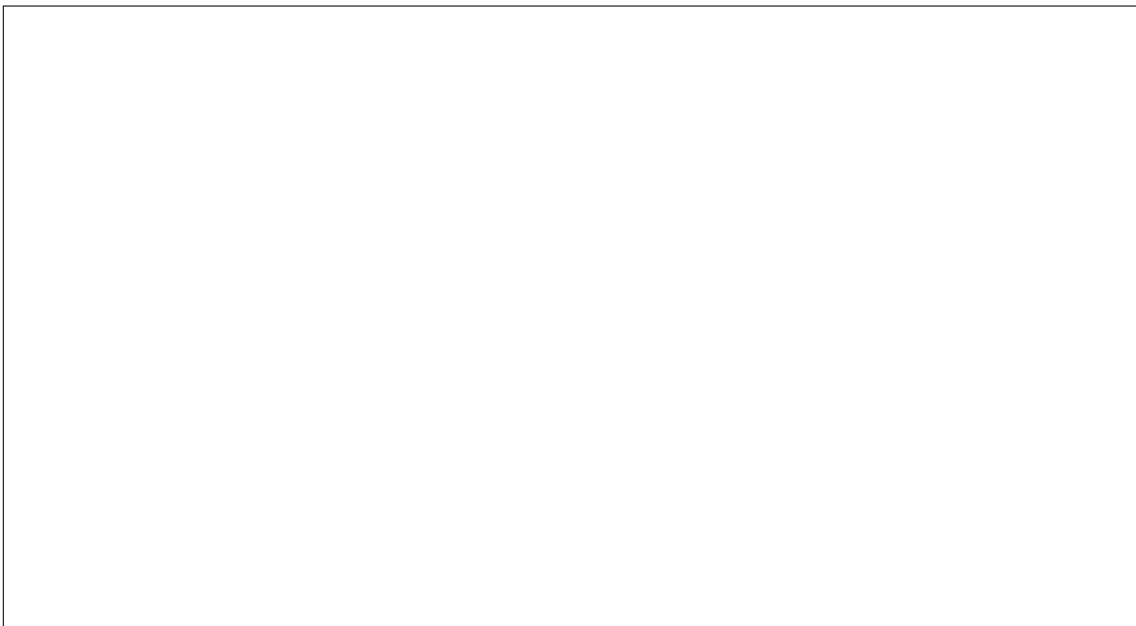


Exercice 5 Rappel : une composante connexe G' d'un graphe G est une partie connexe de G dont aucun sommet n'est relié au reste de G , autrement dit une partie isolée de G , que l'on peut donc aussi considérer comme un graphe. G est ainsi la réunion de ses composantes connexes (il n'y en a qu'une si G lui-même est déjà connexe).

1. Est-ce qu'une composante connexe peut contenir un seul sommet de degré impair ? Justifier.



2. Montrer que si un graphe contient exactement deux sommets x et y de degré impair, il existe nécessairement une chaîne reliant x à y (c'est-à-dire qu'ils sont dans la même composante connexe).



Annexe

Liste des fonctions disponibles pour manipuler les graphes ; cette feuille n'est pas à rendre avec le devoir, ne rien écrire dessus.

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

L'argument s est un sommet	
<code>listeVoisins(s)</code>	retourne la <i>liste</i> des <i>voisins</i> de s .
<code>degre(s)</code>	retourne le <i>degré</i> de s .
<code>nomSommet(s)</code>	retourne le <i>nom</i> (étiquette) de s .
<code>colorierSommet(s,c)</code>	colorie s avec la couleur c . Exemples de couleurs : 'red', 'green', 'blue', None.
<code>couleurSommet(s)</code>	retourne la <i>couleur</i> de s .
<code>marquerSommet(s)</code> <code>demarquerSommet(s)</code>	marque ou démarque s .
<code>estMarqueSommet(s)</code>	retourne True si s est marqué, False sinon.
<code>listeAretesIncidentes(s)</code>	retourne la <i>liste</i> des arêtes <i>incidentes</i> à s .
<code>sommetVoisin(s,a)</code>	retourne le voisin de s en suivant l'arête a .

L'argument a est une arête	
<code>nomArete(a)</code>	retourne le <i>nom</i> (étiquette) de a .
<code>marquerArete(a)</code> <code>demarquerArete(a)</code>	marque ou démarque a .
<code>estMarqueeArete(a)</code>	retourne True si a est marquée, False sinon.