CORRIGÉ DU DS 4

le 26/03/2026 – Durée : 2h

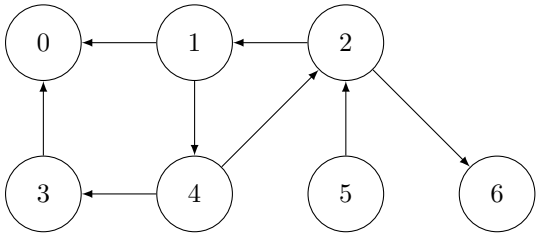
- L'usage de la calculatrice n'est pas autorisé.
- La clarté et la précision des raisonnements interviendront pour une grande part dans la notation. La présentation (en particulier les indentations) est également essentielle.
- Le résultat d'une question peut être admis afin de traiter une question suivante ; une fonction peut être utilisée dans la suite d'un exercice même si elle n'a pas été écrite.
- Le barème tiendra compte de la longueur du devoir.

Exercice 1

D'APRÈS CCINP MATHS 2 2025

1.
```
def degreMax(d):
    m = 0
    for sommet in d:
        deg_sortant = len(d[sommet] )
        if  deg_sortant > m:
            m = deg_sortant
    return m
```

2. Graphe inverse du graphe orienté donné en introduction :




```
def grapheInv(d):
    grInv = {}
    for s in d:
        grInv[s] = []
    for s1 in d:
        for s2 in d[s1]:
            grInv[s2].append(s1)
    return grInv
```

3.
```
def colorationValide(d,L):
    for sommet in d:
        for s in d[sommet]:
            if L[sommet] == L[s]:
                return False
    return True
```

4. Le pire des cas se produit par exemple lorsque le graphe est bien colorié et qu'il faut donc tout tester. On passe alors N fois dans la première boucle et pour chaque passage dans cette boucle, on passe au plus M fois dans la seconde boucle. Au total, on a donc au maximum NM tests d'égalité.

   Remarque : cette façon de voir n'est pas très intéressante en général. En effet, on peut majorer la longueur de la boucle interne par N et donc la complexité par N^2. Cela est meilleur que ce qui précède car en général on a $N < M$.
   Il est par contre intéressant de considérer le degré sortant maximal m. La complexité est en effet de l'ordre de Nm et dans de nombreux graphes m est beaucoup plus petit que N.

5.
```
SELECT MAX(duree)
FROM LOCATIONS
```

6.
```
SELECT F. codefilm , F.nomfilm , AVG(L.duree) AS D
FROM FILMS AS F JOIN LOCATIONS AS L
                  ON F.codefilm = L.codefilm
GROUP BY F.codefilm
HAVING D < 2
ORDER BY D
```

Exercice 2

D'APRÈS CCINP MP MATHS2 2024

1. L'instruction `s[1]` permet d'obtenir $\sigma(1)$

   La transposition demandée est représentée par `[0,1,3,2]`

2.
```
def comp(s1,s2):
    L=[]
    for k in range (len(s1)):
        L.append(s1[s2[k]])
    return L
```

3. On a $\sigma^{-1} \circ \sigma = id$, donc pour i parcourant la liste associée à σ, on a $\sigma^{-1} \circ \sigma(i) = i$.
   On construit donc une liste "vide" que l'on complète avec la relation précédente

```
def inv(s):
    L=[]
    for k in range (len(s)):
        L.append(0)
    for k in range (len(s)):
        L[s[k]]=k
    return L
```

4. On doit donc vérifier les 3 propriétés suivante :

   - le neutre `[0,1,2,3]` est un élément de `G`.
   - la composée de deux éléments de `G` appartient à `G`
   - l'inverse de tout élément de `G` appartient à `G`.

   On pose une variable `res` qui par défaut vaut `True` et deviendra `False` si l'une des 3 conditions précédente n'est pas vérifiée

```
    res = True
    if [0,1,2,3] not in G :
        res = False
    for k in range (len(G)):
        if inv(G[k]) not in G :
            res = False
        for j in range (k+1,len(G)):
            if comp(G[k],G[j]) not in G :
                res = False
    return res
```

5. Puisque le groupe est cyclique, on va construire à partir de σ, successivement : σ^2, σ^3...jusqu'à obtenir l'identité représentée par la liste `[0,1,2,3]`.

```
def cyclique(s):
    G=[s]
    t= s
    while t != [0,1,2,3] :
                t = comp(t,s)
        G.append(t)
        t = comp(t,s)
    return G
```

Exercice 3

D'APRÈS CCINP MP MATHS2 2019

1.
```
def estPremier(n):
    d = 2
    while d**2 <= n:
        if n%d == 0:
            return False
        d += 1
    return True
```

```
def liste_premiers(n):
    l = []
    for i in range(2,n+1):
        if estPremier(i):
            l.append(i)
    return l
```

3.
```
def valuation_p_adique(n,p):
    val = 0
    d = n
    while d%p == 0:
        val += 1
        d = d//p
    return val
```

4.
```
def val_p_adique(n,p):
    if n%p != 0:
        return 0
    else :
        return val_p_adique(n//p,p) +1
```

```
def decomposition_facteurs_premiers(n):
    l = []
    for i in range(2,n+1):
        if estPremier(i)  and n%i == 0:
            l.append([i,val_p_adique(n,i)])
    return l
```

Exercice 4

1.
```
def SimulU():
    x=rd.random()
    if x<0.5:
        return -1
    else:
        return 1
```

2.
```
def SimulX(n):
    X=0
    for i in range(1,n+1):
        X = X+SimulU()
    return X
```

3.
```
def SimulX_bis(n):
    X = 0
    Xmin = 0
    Xmax = 0
    T = 0
    for i in range(1,n+1):
        X = X+SimulU()
        if X > Xmax:
            Xmax = X
        if X < Xmin:
            Xmin = X
        if X == 0 and T == 0:
            T = i
    return [X,Xmin,Xmax,T]
```

4.
```
S=0
for k in range(N):
    l = SimulX_bis(n)
    if l[3] > 0:
        S = S+1

freq = float(S)/N
print(freq)
```