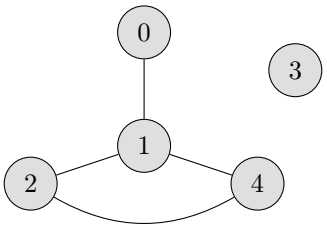
Planche 1

Un graphe est un ensemble constitué de n sommets numérotés de 0 à $n-1$, reliés par des arêtes.

Chaque arête est désignée par le couple des numéros de sommets qu'elle relie ; ainsi, l'arête reliant les sommets i et j est désignée par le couple (i, j), avec $0 \leqslant i < j \leqslant n-1$.

Pour décrire un graphe, il suffit de connaître n, le nombre de sommets, et L, la liste de ses arêtes données dans n'importe quel ordre.



Par exemple, le graphe représenté ci-dessus contient 5 sommets et la liste de ses arêtes est `[(2,4),(1,4),(1,2),(0,1)]`.

1. Écrire une fonction `non_isoles` d'argument la liste L des arêtes d'un graphe, renvoyant la liste de ses sommets non isolés, c'est-à-dire reliés par au moins une arête.

2. Écrire une fonction `degre` de deux arguments L et k qui renvoie le degré du sommet k dans le graphe d'arêtes L, c'est-à-dire le nombre d'arêtes dont ce sommet est l'une des extrémités.

   Par exemple, dans le graphe ci-dessus, les degrés respectifs des sommets 0, 1 et 3 sont 1, 3 et 0.

3. On appelle "liste d'adjacence" d'un graphe à n sommets la liste de longueur n dont l'élément d'indice k est lui-même une liste répertoriant l'ensemble des sommets reliés par une arête au sommet k.

   Écrire une fonction `adjacence` de deux arguments, le nombre n de sommets d'un graphe et la liste L de ses arêtes, renvoyant la liste d'adjacence du graphe.

   Pour l'exemple ci-dessus, la liste d'adjacence est `[[1],[0,2,4],[1,4],[],[1,2]]`.

4. Écrire une fonction `degmax` d'argument une liste d'adjacence A d'un graphe, renvoyant le sommet dont le numéro est le plus grand parmi tous ceux de degré maximum.

5. On dit qu'une liste de sommets $[n_0, \ldots, n_p]$ est un "chemin de longueur p" lorsque, pour tout i compris entre 0 et $p-1$, il existe une arête entre les sommets n_i et n_{i+1}.

   Écrire une fonction `chemin` de trois arguments, le nombre n de sommets d'un graphe, la liste L de ses arêtes et une liste C de sommets, qui renvoie un booléen indiquant si C est effectivement un chemin du graphe ou pas.

   Vérifier que `[0,1,2]` et `[0,1,4,2,1]` sont des chemins du graphe ci-dessus, mais pas `[0,2,1]`.

Corrigé — Planche 1

1.
```
def non_isoles(L):
    sommets = []
    for (i,j) in L:
        if i not in sommets:
            sommets.append(i)
        if j not in sommets:
            sommets.append(j)
    return sommets
```

2.
```
def degre(L,k):
    d = 0
    for (i,j) in L:
        if i==k or j==k:
            d += 1
    return d
```

3.
```
def adjacence(n,L):
    graphe = [[] for k in range(n)]
    for (i,j) in L:
        graphe[i].append(j)
        graphe[j].append(i)
    return graphe
```

4.
```
def degmax(A):
    i_max = 0
    for i in range(1,len(A)):
        if len(A[i]) >= len(A[i_max]):
            i_max = i
    return i_max
```

5.
```
def chemin(n,L,C):
    A = adjacence(n,L)
    for i in range(len(C)-1):
        if C[i+1] not in A[C[i]]:
            return False
    return True
```

Planche 2

On sait que l'écriture décimale d'un nombre rationnel strictement positif est périodique à partie d'un certain rang. Par exemple on a :

$$\frac{4741}{280} = 16,932\,142857\,142857\,142857\cdots \quad ,$$

que l'on récrit :

$$\frac{4741}{280} = 16,932\,\overline{142857} \quad .$$

La liste de chiffres $[1,4,2,8,5,7]$ est appelée « *partie périodique* » de l'écriture décimale, dans laquelle elle est précédée de la « *partie non périodique* », la liste de chiffres $[9,3,2]$.

On appelle donc « *écriture décimale périodique* » d'un nombre rationnel a/b la liste $[n,A,P]$, où n désigne la partie entière de a/b, A la partie non périodique de son écriture décimale, et P sa partie périodique.

1. Écrire une fonction `PE` de deux arguments a et b qui renvoie la partie entière de a/b.
2. Écrire une fonction `decimales` de trois arguments a, b et k qui renvoie la liste des k premières décimales de a/b, calculées par divisions euclidiennes successives.
3. Écrire une fonction `EDP` de deux arguments a et b qui renvoie l'écriture décimale périodique de a/b, sous forme d'une liste $[n,A,P]$. La partie périodique est identifiée dès qu'un reste de division euclidienne réapparaît. Il faut donc stocker les restes successifs.

Corrigé — Planche 2

1.
```
def PE(a,b):
    return a//b
```

2.
```
def decimales(a,b,k):
    L = []
    a = (a%b)*10
    for _ in range(k):
        L.append(a//b)
        a = (a%b)*10
    return L
```

3.
```
def EDP(a,b):
    n = a//b
    a = (a%b)*10
    decimales = []
    restes = []
    while True:
        decimales.append(a//b)
        for i in range(len(restes)):
            if restes[i] == a%b:
               return [n,decimales[:i+1],decimales[i+1:]]
        restes.append(a%b)
        a = (a%b)*10
```

Planche 3

1. Écrire une fonction `prod` de deux arguments, deux listes $L = [\ell_0, \cdots, \ell_{d-1}]$ et $C = [c_0, \cdots, c_{d-1}]$ de mêmes longueurs, renvoyant le produit scalaire $\sum_{i=0}^{d-1} \ell_i \, c_i$ des vecteurs représentés par L et C.

Une matrice carrée d'ordre d est représentée ici par la liste de ses lignes mises bout-à-bout. Par exemple, les listes `[0,1,2,3]` et `[0,1,2,3,4,5,6,7,8]` peuvent représenter les matrices :

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}.$$

2. Écrire une fonction `lig` de deux arguments, une liste M représentant une matrice carrée et un entier naturel i, qui renvoie la ligne i de la matrice représentée par M, sous forme d'une liste.

   Par exemple `lig([0,1,2,3],0)` et `lig([0,1,2,3],1)` donnent respectivement `[0,1]` et `[2,3]`.

3. Écrire une fonction `col` de deux arguments, une liste M représentant une matrice carrée et un entier naturel j, qui renvoie la colonne j de la matrice représentée par M, sous forme d'une liste.

   Par exemple `col([0,1,2,3],0)` et `col([0,1,2,3],1)` donnent respectivement `[0,2]` et `[1,3]`.

4. Écrire une fonction `mul` de deux arguments, deux listes représentant des matrices carrées M et N d'ordres identiques, qui renvoie la liste représentant le produit MN.

5. Écrire une fonction `puis` de deux arguments, une liste représentant un matrice carrée M et un entier naturel non nul p, qui renvoie la liste représentant M^p, en tenant compte de la remarque suivante : si $p = 2q$ est pair, $M^p = (M^q)^2$, et si $p = 2q + 1$ est impair, $M^p = M\,(M^q)^2$.

   Par exemple, `puis([0,1,1,1],2)`, `puis([0,1,1,1],3)` et `puis([0,1,1,1],5)` doivent donner respectivement les listes `[1,1,1,2]`, `[1,2,2,3]` et `[3,5,5,8]`.

Corrigé — Planche 3

1.
```
def prod(L,C):
    s = 0
    for k in range(len(C)):
        s += L[k]*C[k]
    return s
```

2.
```
def lig(M,i):
    n = int((len(M))**0.5)
    return M[i*n:(i+1)*n]
```

3.
```
def col(M,j):
    n = int((len(M))**0.5)
    return M[j::n]
```

4.
```
def mul(M,N):
    P = []
    n = int((len(M))**0.5)
    for i in range(n):
        for j in range(n):
            P.append(prod(lig(M,i),col(N,j)))
    return P
```

5.
```
def puis(M,p):
    if p == 1:
        return M
    elif p%2==0:
        q = p//2
        N = puis(M,q)
        return mul(N,N)
    else:
        q = p//2
        N = puis(M,q)
        return mul(M,mul(N,N))
```