



EPFL



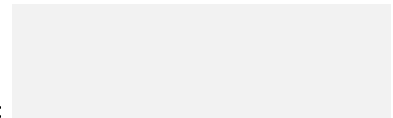
n/a

Ens. : G. Maatouk, C.D. Petrescu
Informatique et calcul scientifique (ICS) - (n/a)
12 janvier 2022
Durée : 105 minutes

n/a

SCIPER: 999999

Signature:



Attendez le début de l'épreuve avant de tourner la page.

Ce document est imprimé recto-verso, il contient 12 pages, les dernières pouvant être vides.
Ne pas dégrafer.

Pour les questions ouvertes (deuxième partie), répondez dans l'espace dédié (n'écrivez pas dans les marges) !

- Posez votre carte d'étudiant sur la table.
- Le seul matériel autorisé est une feuille A4 recto verso manuscrite, ainsi que le polycopié officiel de la première partie du cours (ou bien une version équivalente imprimée et reliée ou agrafée en un fascicule).
- L'utilisation d'une **calculatrice** et de tout outil électronique est interdite pendant l'épreuve.
- Pour les questions à **choix unique**, on comptera :
 - le nombre maximal de point(s) attribué(s) si la réponse est correcte,
 - 0 point si il n'y a aucune ou plus d'une réponse inscrite,
 - 0 point si la réponse est incorrecte.
- Utilisez un **stylo** à encre **noire ou bleu foncé** et effacez proprement avec du **correcteur blanc** si nécessaire.
- Si une question est erronée, l'enseignant se réserve le droit de l'annuler.

Respectez les consignes suivantes Read these guidelines Beachten Sie bitte die unten stehenden Richtlinien		
choisir une réponse select an answer Antwort auswählen	ne PAS choisir une réponse NOT select an answer NICHT Antwort auswählen	Corriger une réponse Correct an answer Antwort korrigieren
  		 
ce qu'il ne faut PAS faire what should NOT be done was man NICHT tun sollte		
     		



Première partie, questions à choix unique

Pour chaque question marquez la case correspondante à la réponse correcte sans faire de ratures. Il n'y a qu'une seule réponse correcte par question.

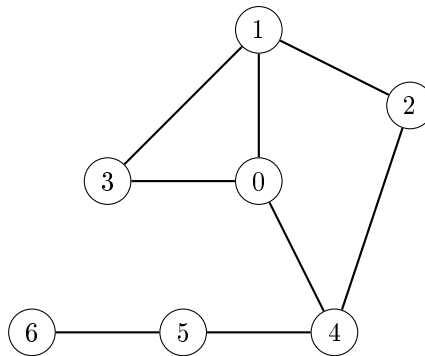
On reproduit ci-dessous le code des algorithmes de parcours de graphe BFS et DFS vus en cours.

```
from collections import deque

def BFS(G,s):
    n = len(G)
    a_parcourir = deque([s])
    vu = [0 for u in range(n)]
    vu[s] = 1
    while a_parcourir:
        sommet = a_parcourir.popleft()
        for u in G[sommet]:
            if not vu[u]:
                a_parcourir.append(u)
                vu[u] = 1
        print(sommet)
```

```
def DFS(G, s):
    print(s)
    vu[s] = 1 #vu: liste globale
                initialisee a
                len(G) zeros
    for u in G[s]:
        if not vu[u]:
            DFS(G,u)
```

On considère le graphe G suivant:



Question 1 : (6 points)

Lequel des parcours ci-dessous **ne peut pas** avoir été produit par l'appel DFS(G,0)?

- ☐ 0 4 2 1 3 5 6
- ☐ 0 4 5 6 2 1 3
- ☐ 0 1 3 2 4 5 6
- ☐ 0 1 3 4 2 5 6

Question 2 : (6 points)

Lequel des parcours ci-dessous **ne peut pas** avoir été produit par l'appel BFS(G,5)?

- ☐ 5 6 4 0 2 3 1
- ☐ 5 6 4 0 3 1 2
- ☐ 5 4 6 2 0 1 3
- ☐ 5 4 6 0 2 1 3

**Question 3 :** (6 points)

On reproduit ci-dessous le code de l'algorithme de recherche binaire vu en cours, où on a rajouté l'instruction `print(milieu, end=' ')` dans la boucle `while`.

```
def recherche_binaire(L, x):
    n = len(L)
    bas = 0
    haut = n-1

    while haut >= bas:
        milieu = (bas + haut)//2
        print(milieu, end=' ')
        if L[milieu] == x:
            return milieu
        elif L[milieu] > x:
            haut = milieu - 1
        else:
            bas = milieu + 1
```

On considère la liste

`L = [-8, -5, -5, -2, 0, 1, 2, 9, 12, 15, 18, 20, 26, 28, 32, 35]`

Un appel à `recherche_binaire(L,x)` produit l'affichage suivant:

7 11 9 10

Que peut-on en déduire de la valeur de x ?

- ☐ $18 \leq x < 20$
- ☐ $15 < x < 20$
- ☐ $15 < x \leq 18$
- ☐ $x = 18$

Question 4 : (3 points)

On considère les deux fonctions suivantes définies sur les entiers naturels:

$$f(n) = n(1 + \sqrt{n}) \text{ et } g(n) = 0.1n^2.$$

Laquelle des affirmations suivantes est vraie?

- ☐ $f(n) = \Theta(g(n))$
- ☐ $f(n) = \mathcal{O}(g(n))$ mais $g(n)$ n'est pas $\mathcal{O}(f(n))$
- ☐ $g(n) = \mathcal{O}(f(n))$ mais $f(n)$ n'est pas $\mathcal{O}(g(n))$
- ☐ On ne peut pas comparer l'ordre de croissance de f et de g

Question 5 : (3 points)

On considère les deux fonctions suivantes définies sur les entiers naturels:

$$f(n) = \log_2(n\sqrt{n}) \text{ et } g(n) = \log_2(\sqrt{n}).$$

Laquelle des affirmations suivantes est vraie?

- ☐ $f(n) = \Theta(g(n))$
- ☐ On ne peut pas comparer l'ordre de croissance de f et de g
- ☐ $f(n) = \mathcal{O}(g(n))$ mais $g(n)$ n'est pas $\mathcal{O}(f(n))$
- ☐ $g(n) = \mathcal{O}(f(n))$ mais $f(n)$ n'est pas $\mathcal{O}(g(n))$

**Question 6 :** (6 points)

On reproduit ci-dessous l'algorithme de tri par sélection vu en cours, où on a rajouté l'instruction `print(L)` à la fin de la boucle extérieure (boucle englobante):

```
def tri_par_selection(L):  
    n = len(L)  
    for i in range(n):  
        m = L[i]  
        m_index = i  
        for j in range(i+1,n):  
            if L[j] < m:  
                m = L[j]  
                m_index = j  
        L[i], L[m_index] = L[m_index], L[i]  
        print(L)
```

Quel affichage est produit par les deux instructions suivantes?

`L = [5, 4, 3, 2, 1]`

`tri_par_selection(L)`

- ☐ [4, 3, 2, 1, 5]
[3, 2, 1, 4, 5]
[2, 1, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
- ☐ [1, 4, 3, 2, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
- ☐ [1, 5, 4, 3, 2]
[1, 2, 5, 4, 3]
[1, 2, 3, 5, 4]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
- ☐ [5, 4, 3, 2, 1]
[4, 5, 3, 2, 1]
[3, 4, 5, 2, 1]
[2, 3, 4, 5, 1]
[1, 2, 3, 4, 5]

**Question 7 :** (6 points)

On reproduit ci-dessous l'algorithme de tri par insertion vu en cours, où on a rajouté l'instruction `print(L)` à la fin de la boucle extérieure (boucle englobante):

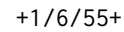
```
def tri_par_insertion(L):  
    n = len(L)  
    for i in range(n):  
        j = i  
        while j > 0 and L[j] < L[j-1]:  
            L[j], L[j-1] = L[j-1], L[j]  
            j -= 1  
        print(L)
```

Quel affichage est produit par les deux instructions suivantes?

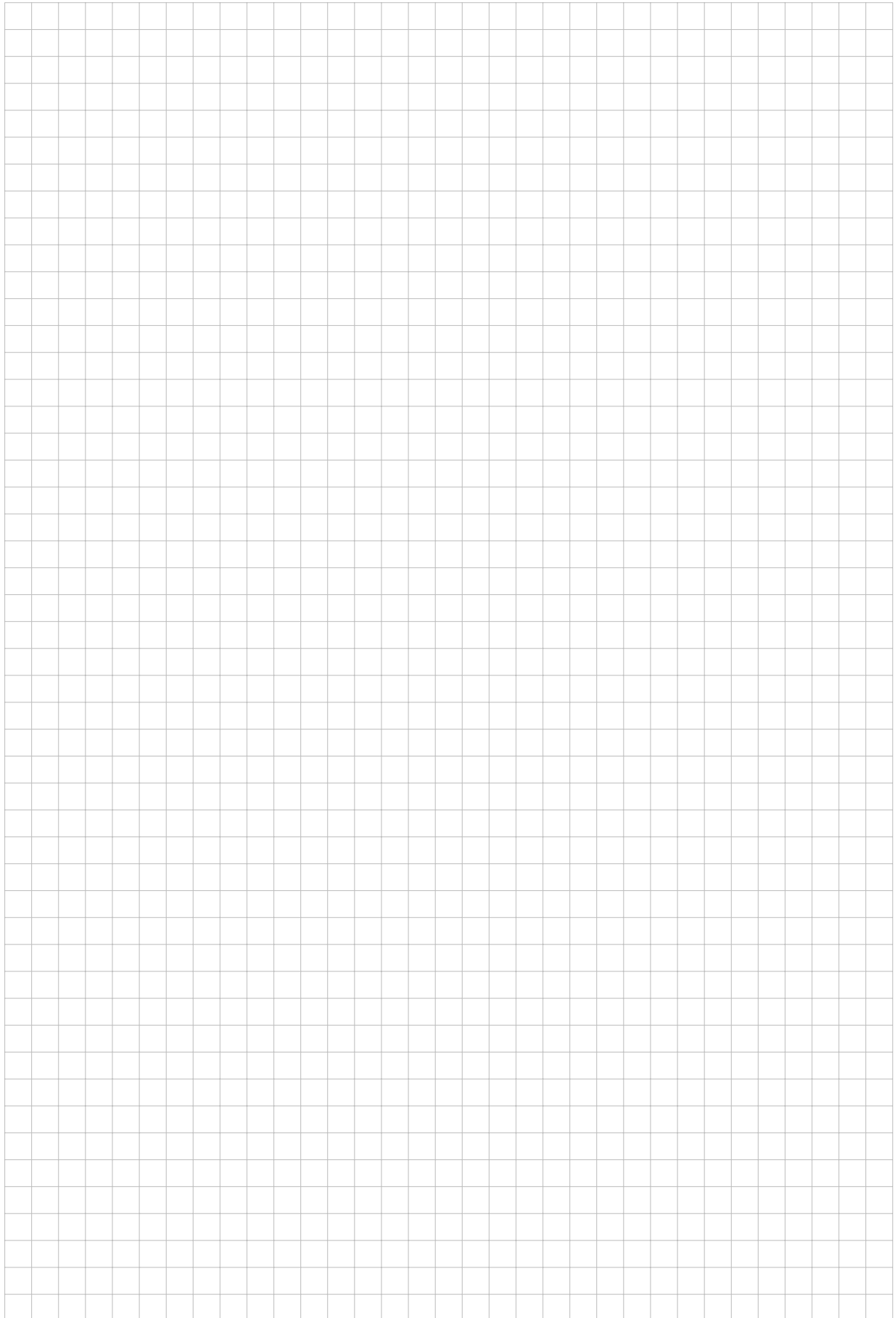
`L = [5, 4, 3, 2, 1]`

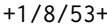
`tri_par_insertion(L)`

- ☐ [5, 4, 3, 2, 1]
[4, 5, 3, 2, 1]
[3, 4, 5, 2, 1]
[2, 3, 4, 5, 1]
[1, 2, 3, 4, 5]
- ☐ [1, 4, 3, 2, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
- ☐ [1, 5, 4, 3, 2]
[1, 2, 5, 4, 3]
[1, 2, 3, 5, 4]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
- ☐ [4, 3, 2, 1, 5]
[3, 2, 1, 4, 5]
[2, 1, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]



```
L = [4, 7, 5, 2, 6, 1, 3, 8]
tri_par_fusion(L, 0, len(L)-1)
```





0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32

On donne la représentation par matrice d'adjacence d'un graphe G , comme suit:

$$G = [[0,0,0,0,0,0], [1,0,0,0,0,1], [0,1,0,1,0,0], [0,0,1,0,1,0], [0,0,1,0,0,0], [0,0,0,1,0,0]]$$

- (a) Ce graphe est-il dirigé ou non dirigé? Justifiez brièvement.
- (b) Donnez la représentation par listes d'adjacence de ce graphe G . Nommez les sommets du graphe 0, 1, 2, 3, 4, 5.
- (c) Ecrivez un algorithme `matrice_vers_listes` qui prend en entrée la représentation par matrice d'adjacence d'un graphe (dirigé ou non) et retourne la représentation par listes d'adjacence de ce même graphe. Par exemple, l'instruction

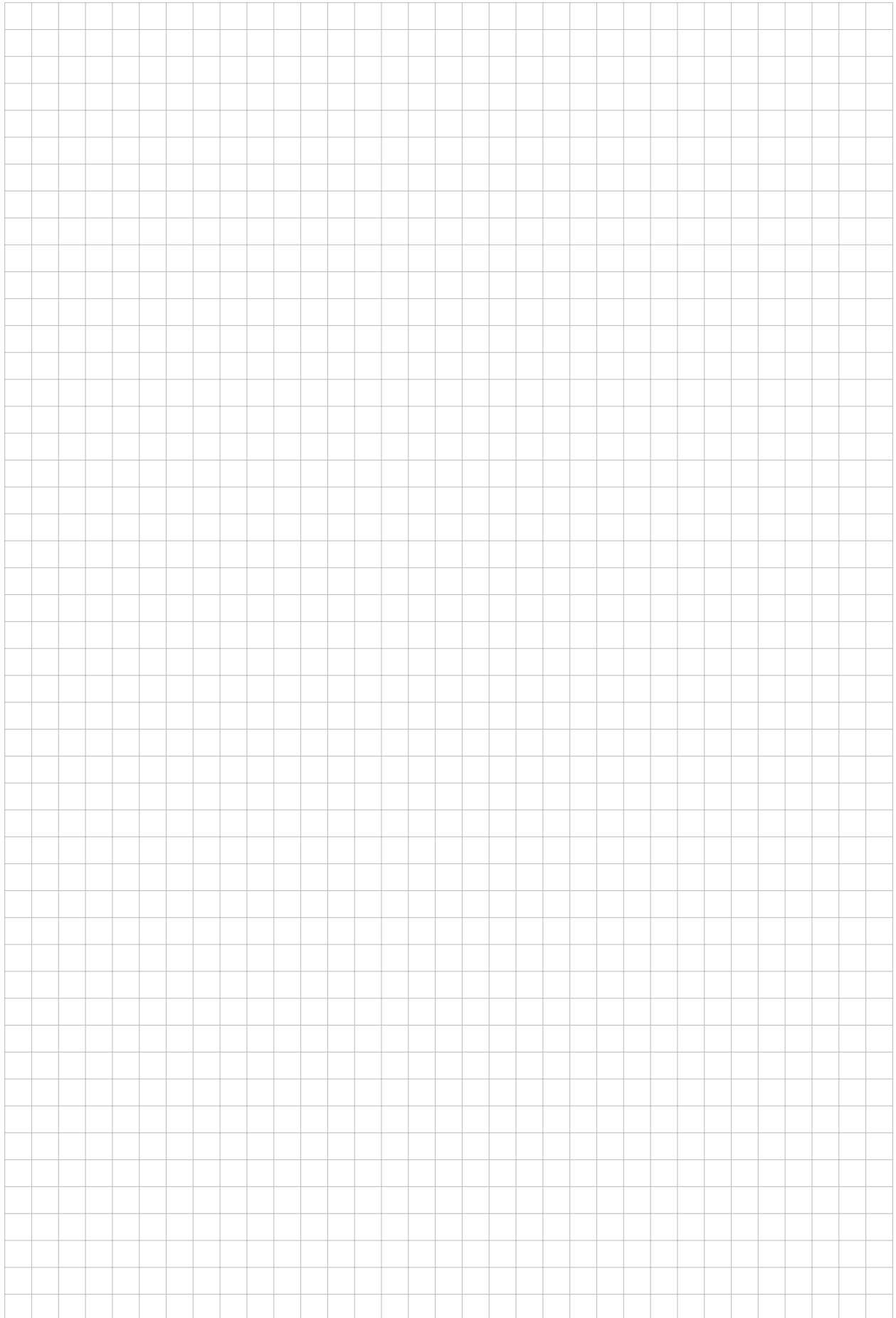
```
print(matrice_vers_listes([[0,1],[1,0]]))
```

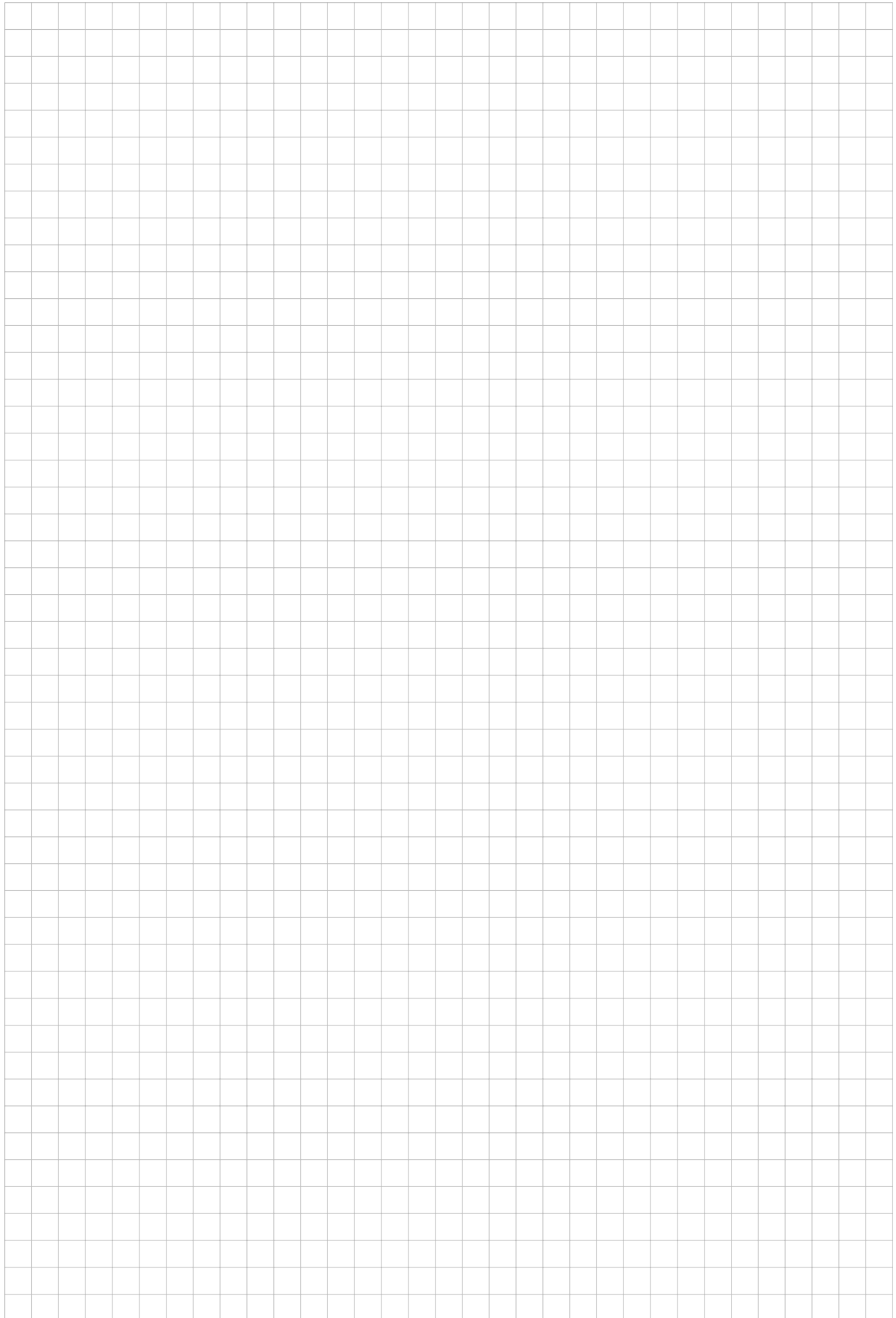
doit afficher

```
{0:[1], 1:[0]}
```
- (d) On dénote $T(n)$ le temps de parcours de l'algorithme `matrice_vers_listes` lorsqu'il prend en entrée la matrice d'adjacence d'un graphe à n sommets. Quel est l'ordre de croissance de $T(n)$ en notation $\Theta(\cdot)$? Justifiez brièvement.



+1/9/52+







Question 10: *Cette question est notée sur 20 points.*

	0		.		5		1		.		5		2		.		5		3		.		5		4		.		5		5		.		5		6		.		5
	7		.		5		8		.		5		9		.		5		10		.		5		11		.		5		12		.		5		13		.		5
	14		.		5		15		.		5		16		.		5		17		.		5		18		.		5		19		.		5		20				

Réservé au correcteur

Ecrivez un algorithme **récuratif** `c_count` qui prend en entrée une chaîne de caractères `s` et un caractère `c` (donc une chaîne de caractères de longueur 1) et retourne le nombre de fois où le caractère `c` apparaît dans `s`.

Par exemple, l'appel `c_count("ababa","a")` doit retourner 3; `c_count("ababa","c")` doit retourner 0; et `c_count("", "x")` doit retourner 0.

Votre algorithme ne doit contenir **aucune boucle** et ne peut pas faire appel aux méthodes `s.count()`, `s.split()` et `s.rsplit()`.



