

INFORMATIQUE

2022



Union des Professeurs
de classes préparatoires
Scientifiques

Sommaire

Option informatique	1
X – ENS A (XULCR) (4h) [i223m1ea]	
Différenciation algorithmique	1
X – ENS Info-maths (XULCR) (4h) [i22im1ea]	
Automates, points-fixes, et équivalence de langage	14
Mines-Ponts (3h) [i22mmoe]	
Lecture et correction d'un mot brouillé	27
Centrale-Supélec (4h) [i22cmoe]	
Automates, déterminisation et automate de Brzozowski	37
CCINP (4h) [i22pmoe]	
Partie I (option informatique) : automates - Partie II (informatique commune) : autour des tas - Partie III (option informatique) : autour de l'énumération des fractions positives	46
CAPES externe d'informatique - Épreuve 1 [i22c31e]	
Étude de protocoles de traçage de contacts numériques	54
CAPES externe d'informatique - Épreuve 2 [i22c32e]	
Gestion des données, algorithmes de tri	66
Agrégation externe d'informatique - Épreuve 1 [i22ag1e]	
Partie 1 : système - Partie 2 : nombres flottants - Partie 3 : logique, déduction naturelle - Partie 4 : réseaux	90
Agrégation externe d'informatique - Épreuve 2 [i22ag2e]	
De très grands entiers	104
Agrégation externe d'informatique - Épreuve 3 [i22ag3e]	
Option A : problème des 8 dames - Option B : compilation	117
Informatique commune	148
X – ENS B (XELCR) - MP, PC, PSI (2h) [i223m2e]	
Spéléo-logique	148
Mines-Ponts (2h) - MP-PC-PSI [i22mmce]	
Modélisation numérique d'un matériau magnétique	159
Centrale-Supélec - MP, PC, PSI, TSI (3h) [i22cice]	
Modélisations autour de la Formule 1	172

CCINP - PSI (3h) [i22psue]	
Assemblage et déformation de pièces automobiles	180
CCINP - TSI (3h) [i22piue]	
Représentation de données géolocalisées sur une carte numérique	208
Épreuves de modélisation	220
CCINP - PC (4h) [d22ppue]	
Étude d'une réaction à solide consommable	220
CCINP - PSI (4h) [d22psue]	
Modélisation de la prévention des tsunamis	236
CCINP - TSI (3h) [d22piue]	
Stabilisateur d'images Slick	252
CCINP - TPC (4h) [d22pcue]	
Étude d'une réaction à solide consommable	264
Banque PT (4h) [d22dtue]	
Suivi de la consommation d'eau d'une habitation	280

En guise d'introduction

Dans la vie, rien n'est à craindre, tout est à comprendre.

Marie Curie

Le recueil 2022 comporte trois parties. La première contient les épreuves de l'option informatique de nos classes auxquelles s'ajoutent les épreuves des CAPES et agrégation d'informatique. La deuxième contient les épreuves d'informatique commune. Enfin, la troisième contient des épreuves de modélisation qui comportent au moins une question d'informatique.

Le bulletin des concours Informatique n'est proposé aux adhérents que sous forme électronique. Dans ce recueil, le nom du fichier contenant chaque sujet figure dans le sommaire et en tête de chaque page. Les fichiers sont disponibles sur le site de l'UPS à l'adresse <https://ups-cpge.fr/ups.php?module=Maths&voir=recherche>. Ce bulletin et ses prédecesseurs sont à votre disposition sur le site de l'UPS à l'adresse <http://prepas.org/ups.php?rubrique=146>.

Amélie Stainer - Septembre 2022

amelie.stainer@prepas.org

**ÉCOLE POLYTECHNIQUE
ÉCOLES NORMALES SUPERIEURES
CONCOURS D'ADMISSION 2022**

**MARDI 26 AVRIL 2022
14h00 - 18h00
FILIÈRE MP - Épreuve n° 4
INFORMATIQUE A (XULSR)**

Durée : 4 heures

***L'utilisation des calculatrices n'est pas autorisée
pour cette épreuve***

*Cette composition ne concerne qu'une partie des candidats de la filière MP,
les autres candidats effectuant simultanément la composition de Physique et
Sciences de l'Ingénieur.*

Pour la filière MP, il y a donc deux enveloppes de Sujets pour cette séance.

Différentiation algorithmique

Le sujet comporte 12 pages, numérotées de 1 à 12.

Début de l'épreuve.

Dans ce sujet, on s'intéresse au problème de la différentiation algorithmique. Il s'agit de calculer efficacement des différentielles d'expressions mathématiques vues comme des composées de fonctions élémentaires. C'est l'une des pierres fondatrices du calcul formel, ainsi que de la descente de gradient dans les réseaux de neurones.

Ce sujet est constitué de cinq parties. La première partie est formée de préliminaires utiles dans le reste du sujet, mais les fonctions qui y sont définies peuvent être admises pour traiter les parties suivantes. La troisième partie est largement indépendante de la deuxième, à l'exception de la question III.7. La quatrième partie nécessite d'avoir compris les enjeux de la troisième partie. Enfin, la cinquième et dernière partie est nettement indépendante du reste du sujet, seule les questions V.1 et V.2 font référence aux sujets abordés avant.

Notations et définitions

- On note $(\mathbf{e}_1, \dots, \mathbf{e}_n)$ la base *canonique* de \mathbb{R}^n (pour n un entier strictement positif), de sorte que pour tout $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$, $\mathbf{x} = \sum_{i=1}^n x_i \mathbf{e}_i$.
- Pour une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (avec $n, m \in \mathbb{N}^*$), on note $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ (pour $1 \leq i \leq m$) la projection de f sur la coordonnée i : $f_i : \mathbf{x} \mapsto f(\mathbf{x}) \cdot \mathbf{e}_i$.
- Pour une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (avec $n, m \in \mathbb{N}^*$), on note $D_{\mathbf{x}}f(\mathbf{v})$ la *différentielle* de la fonction f en un point \mathbf{x} évaluée en un vecteur \mathbf{v} (cette notation présuppose que f est différentiable en \mathbf{x}). On rappelle que la fonction $D_{\mathbf{x}}f$ est linéaire en son argument \mathbf{v} .
- On rappelle également la *règle de la chaîne* : pour $n, m, p \in \mathbb{N}^*$, $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $g : \mathbb{R}^m \rightarrow \mathbb{R}^p$ et $\mathbf{x} \in \mathbb{R}^n$, on a :

$$D_{\mathbf{x}}(g \circ f) = D_{f(\mathbf{x})}g \circ D_{\mathbf{x}}f,$$

où \circ représente la composition de fonctions.

- On note également $\frac{\partial f}{\partial x_i} = D_{\mathbf{x}}f(\mathbf{e}_i)$ la dérivée de f en \mathbf{x} suivant le vecteur \mathbf{e}_i de la base canonique ; pour $\mathbf{v} = (v_1, \dots, v_n)$ et par linéarité de la différentielle, on a donc :

$$D_{\mathbf{x}}f(\mathbf{v}) = \sum_{i=1}^n v_i \frac{\partial f}{\partial x_i}$$

- On note $\mathcal{M}_{m,n}(\mathbb{R})$ l'ensemble des matrices à m lignes, n colonnes, et coefficients dans \mathbb{R} . On supposera toujours $m \geq 1, n \geq 1$.
- Pour $n, m \in \mathbb{N}^*$, la *matrice jacobienne* d'une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ en un point $\mathbf{x} \in \mathbb{R}^n$, notée $J_f(\mathbf{x})$, est la matrice de $\mathcal{M}_{m,n}(\mathbb{R})$ définie par :

$$J_f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \dots & \frac{\partial f}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

- En appliquant la règle de la chaîne, on observe que pour deux fonctions $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ et $g : \mathbb{R}^m \rightarrow \mathbb{R}^p$ et pour un point $\mathbf{x} \in \mathbb{R}^n$, on a : $J_{g \circ f}(\mathbf{x}) = J_g(f(\mathbf{x})) \times J_f(\mathbf{x})$.

Complexité

Par complexité en temps d'un algorithme \mathcal{A} , on entend le nombre d'opérations élémentaires (comparaison, addition, soustraction, multiplication, division, affectation, test, etc.) nécessaires à l'exécution de \mathcal{A} dans le cas le pire.

Par complexité en mémoire d'un algorithme \mathcal{A} , on entend l'espace mémoire total utilisé par l'exécution de \mathcal{A} dans le cas le pire, en supposant que l'espace mémoire occupé pour stocker une valeur d'un type de base (entier, booléen, flottant, caractère) est une constante.

Lorsque la complexité (en temps ou en espace) dépend d'un paramètre κ , on dit que \mathcal{A} a une complexité (en temps ou en espace) en $O(f(\kappa))$ s'il existe une constante $C > 0$ telle que, pour toutes les valeurs de κ suffisamment grandes (c'est-à-dire plus grandes qu'un certain seuil), pour toute instance du problème de paramètre κ , la complexité (en temps ou en espace) est au plus $C \cdot f(\kappa)$; on dit que \mathcal{A} a une complexité (en temps ou en espace) en $\Omega(f(\kappa))$) s'il existe une constante $C > 0$ telle que, pour toutes les valeurs de κ suffisamment grandes, pour toute instance du problème de paramètre κ , la complexité (en temps ou en espace) est au moins $C \cdot f(\kappa)$. Enfin, on a une complexité en $\Theta((f(\kappa)))$ quand elle est à la fois en $O((f(\kappa)))$ et en $\Omega(f(\kappa))$.

Quand l'on demande la complexité d'un algorithme, il s'agit de trouver une expression en $\Theta(\cdot)$ la plus simple possible. On dit qu'un algorithme est linéaire en κ si sa complexité en temps est en $\Theta(\kappa)$.

Rappels OCaml

- On rappelle les opérations de base sur les nombres en virgule flottante (ou flottants), que l'on utilisera pour représenter des nombres réels dans OCaml :
 - 1.0 et 0.0 (ou 1. et 0.) représentent les nombres 1 et 0 respectivement.
 - L'addition entre deux flottants **a** et **b** s'écrit **a +. b**.
 - La multiplication entre deux flottants **a** et **b** s'écrit **a *. b**.
- On rappelle quelques opérations de base sur les tableaux :
 - **Array.length** : `'a array -> int` donne la longueur d'un tableau.
 - **Array.make** : `int -> 'a -> 'a array` permet de créer un tableau : `Array.make m r` crée un tableau de taille m dont toutes les cases sont initialisées à r .
 - **Array.make_matrix** : `int -> int -> 'a -> 'a array array` permet de créer une matrice : `Array.make_matrix m n r` crée une matrice de taille $m \times n$ dont toutes les cases sont initialisées à r .
- Enfin, une opération de base sur les listes :
 - **List.rev** : `'a list -> 'a list` renvoie le miroir de la liste passée en argument (cette fonction a une complexité linéaire en la taille de la liste).

Dans l'ensemble du sujet, il sera fréquemment question de matrices de flottants. Pour simplifier l'écriture du type des fonctions manipulant de telles matrices, nous définissons le type **matrix** :

```
type matrix = float array array;;
```

Partie I

Question I.1. Donner une fonction OCaml

```
identite : int -> matrix
```

tenant en entrée un entier $n \in \mathbb{N}^*$ et renvoyant la matrice identité de $\mathcal{M}_{n,n}(\mathbb{R})$.

Question I.2. Donner une fonction OCaml

```
scalaire : float array -> float array -> float
```

tenant en entrée deux vecteurs u et v et renvoyant le produit scalaire $u \cdot v$.

Question I.3. Donner une fonction OCaml

```
mul_possible : matrix -> matrix -> bool
```

tenant en entrée deux matrices A et B et renvoyant `true` si et seulement si les dimensions de A et B sont telles que $A \times B$ est bien définie.

Question I.4. Donner une fonction OCaml

```
mul : matrix -> matrix -> matrix
```

qui calcule le produit $A \times B$ des deux matrices A et B de flottants passées en argument. Si A est de dimension $m \times n$ et B de dimension $n \times p$, on impose que la fonction `mul` effectue $m \times n \times p$ multiplications de flottants.

Dans l'ensemble du sujet, on utilisera la fonction `mul` à chaque fois qu'il s'agit de calculer le produit de deux matrices, sans chercher à rendre ce calcul plus efficace.

Partie II

Dans cette partie, on s'intéresse à la multiplication d'une suite finie de matrices réelles $(B^i)_{1 \leq i \leq n}$. Soit k_0, \dots, k_n la suite finie d'entiers tels que $B^i \in \mathcal{M}_{k_{i-1}, k_i}(\mathbb{R})$. On cherche à écrire un programme effectuant le parenthésage optimal sur les B^i pour un calcul efficace de leur multiplication $B^1 \times \dots \times B^n$. Pour chaque $1 \leq i \leq j \leq n$, on note $B^{i,j}$ le produit de matrices $B^i \times \dots \times B^j$, et $p^{i,j}$ le nombre minimum de multiplications de flottants nécessaires pour calculer $B^{i,j}$. On rappelle que l'on réutilisera la fonction `mul` de la partie précédente implémentant la multiplication de deux matrices, que l'on ne cherchera pas à améliorer.

Question II.1. On suppose (dans cette question uniquement) $n = 3$. Comparer le nombre de multiplications de flottants effectuées en évaluant $(B^1 \times B^2) \times B^3$ avec `mul` et en évaluant $B^1 \times (B^2 \times B^3)$ avec `mul`. Que faut-il en conclure sur l'ordre d'évaluation du produit $B^1 \times B^2 \times B^3$?

Question II.2. Donner une fonction OCaml

```
muld : matrix list -> matrix
```

qui prend en entrée une liste non vide de matrices (B^1, \dots, B^n) et renvoie leur multiplication (en utilisant `mul`) via un parenthésage à droite : $B^1 \times (B^2 \times (\dots))$.

Question II.3. Donner une fonction OCaml

```
mulg : matrix list -> matrix
```

qui prend en entrée une liste non vide de matrices (B^1, \dots, B^n) et renvoie leur multiplication (en utilisant `mul`) via un parenthésage à gauche : $((\dots) \times B^{n-1}) \times B^n$.

Question II.4. Donner une fonction OCaml

```
min_mul: int array -> int
```

qui prend en entrée le tableau $k = [k_0, \dots, k_n]$ des tailles de chacune des matrices, et renvoie $p^{1,n}$ en faisant des appels récursifs pour calculer les $p^{i,j}$, sans stockage d'information supplémentaire. Montrer que si, pour tout $i, k_i \geq 2$, la complexité en temps de `min_mul` est au moins exponentielle en n , c'est-à-dire en $\Omega(2^n)$.

Question II.5. Donner une fonction OCaml

```
min_mul_opt : int array -> int
```

qui calcule le même nombre, mais où une matrice auxiliaire de taille $n \times n$, initialement nulle, permet de stocker les valeurs de $p^{i,j}$ déjà calculées. Quelle est la complexité en temps de `min_mul_opt` en fonction de n ? En espace ?

Question II.6. Donner une fonction OCaml

```
mul_opt : matrix array -> matrix
```

qui prend en entrée le tableau des matrices $[B^1, \dots, B^n]$ et calcule $B^{1,n}$ en $p^{1,n}$ multiplications de flottants.

Partie III

On se propose d'étudier deux manières de calculer la différentiation de la composition d'une liste de fonctions. Soient f^1, f^2, \dots, f^n des fonctions de type $f^i : \mathbb{R}^{k_{i-1}} \rightarrow \mathbb{R}^{k_i}$ pour $k_0, \dots, k_n \in \mathbb{N}^*$. On note $f = f^n \circ \dots \circ f^1$. Il s'agit d'optimiser le calcul successif de la valeur de f^i et de celle de sa différentielle en un point. On note $J_x^i \in \mathcal{M}_{k_i, k_{i-1}}(\mathbb{R})$ la matrice jacobienne de f^i en $\mathbf{x} \in \mathbb{R}^{k_{i-1}}$. Pour tout i , on note J^i la fonction

$$J^i : \begin{cases} \mathbb{R}^{k_{i-1}} & \rightarrow \mathcal{M}_{k_i, k_{i-1}}(\mathbb{R}) \\ \mathbf{x} & \mapsto J_x^i \end{cases}$$

Question III.1. Pour une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ avec $n, m \in \mathbb{N}^*$, on note $\vec{D}f$ la fonction :

$$(\mathbf{x}, h) \mapsto (f(\mathbf{x}), (\mathbf{D}_{\mathbf{x}}f) \circ h)$$

Si on impose que le domaine de la fonction h est \mathbb{R}^p pour un certain $p \in \mathbb{N}^*$, quels sont l'ensemble de départ (domaine) et l'ensemble d'arrivée (codomaine) de la fonction $\vec{D}f$?

Montrer que l'opérateur \vec{D} vérifie :

$$\vec{D}(g \circ f) = \vec{D}g \circ \vec{D}f$$

Question III.2. Pour deux fonctions $f^1 : \mathbb{R}^{k_0} \rightarrow \mathbb{R}^{k_1}$ et $f^2 : \mathbb{R}^{k_1} \rightarrow \mathbb{R}^{k_2}$ et un point $\mathbf{x} \in \mathbb{R}^{k_0}$, montrer que $\vec{D}f^2(f^1(\mathbf{x}), \mathbf{D}_{\mathbf{x}}f^1) = ((f^2 \circ f^1)(\mathbf{x}), \mathbf{D}_{\mathbf{x}}(f^2 \circ f^1))$.

Question III.3. Donner une fonction OCaml `forward` de type :

```
float array ->
  (float array -> float array) list ->
    ((float array -> float array) -> float array -> matrix ->
      matrix)
```

qui prend en entrée un point $\mathbf{x} \in \mathbb{R}^{k_0}$, une liste de fonctions f^1, \dots, f^n de type $f^i : \mathbb{R}^{k_{i-1}} \rightarrow \mathbb{R}^{k_i}$, et une fonction

```
diff : (float array -> float array) -> float array -> matrix
```

renvoyant la jacobienne d'une fonction en un point passé en argument, et calcule la matrice représentant $\mathbf{D}_{\mathbf{x}}(f^n \circ \dots \circ f^1)$ via un parenthésage à droite de la forme $\mathbf{D}_{\mathbf{x}}(f^n \circ (f^{n-1} \circ \dots))$. On s'appuiera sur la définition de \vec{D} et sur la propriété établie à l'aide de la question précédente. On pourra s'aider d'une fonction récursive auxiliaire. Prouver la correction de cette fonction.

Question III.4. Quelle est la complexité en temps de cette fonction dans le cas où $k_i = 2^i$?

Dans le cas où pour tout i , $k_i = k$ avec $k \geq 2$ une constante quelconque ?

On supposera que les appels à la fonction `diff` se font en temps proportionnel à la taille de la matrice jacobienne renvoyée par cette fonction.

Question III.5. Donner une fonction `backward` de type

```
float array ->
  (float array -> float array) list ->
    ((float array -> float array) -> float array -> matrix ->
     matrix)
```

qui prend en entrée un point $\mathbf{x} \in \mathbb{R}^{k_0}$, une liste de fonctions f^1, \dots, f^n de type $f^i : \mathbb{R}^{k_{i-1}} \rightarrow \mathbb{R}^{k_i}$, et une fonction

```
diff : (float array -> float array) -> float array -> matrix
```

renvoyant la jacobienne d'une fonction en un point passé en argument, et calcule la matrice représentant $D_{\mathbf{x}}(f^n \circ \dots \circ f^1)$ via un parenthésage à gauche de la forme $D_{\mathbf{x}}((\dots \circ f^2) \circ f^1)$.

Question III.6. Pour une liste de fonctions de longueur n arbitraire, donner un exemple de cas où `backward` est plus efficace que `forward`, ainsi que de cas où `forward` est plus efficace que `backward`.

Question III.7. Sans écrire explicitement de fonction, expliquer comment adapter les techniques des questions de la partie II pour écrire une fonction `diff_opt` qui prend en entrée $\mathbf{x} \in \mathbb{R}^{k_0}$, une liste de n fonctions f^1, \dots, f^n de type $f^i : \mathbb{R}^{k_{i-1}} \rightarrow \mathbb{R}^{k_i}$, et un tableau de n fonctions $[J^1, \dots, J^n]$ et calcule avec un minimum de multiplications de flottants la matrice jacobienne de $f = f^n \circ \dots \circ f^1$ en \mathbf{x} .

Partie IV

La représentation d'une suite d'instructions mathématiques comme une composition de fonctions est assez peu concise. En particulier, elle ne tient pas compte des dépendances linéaires entre les instructions. Considérons par exemple la fonction :

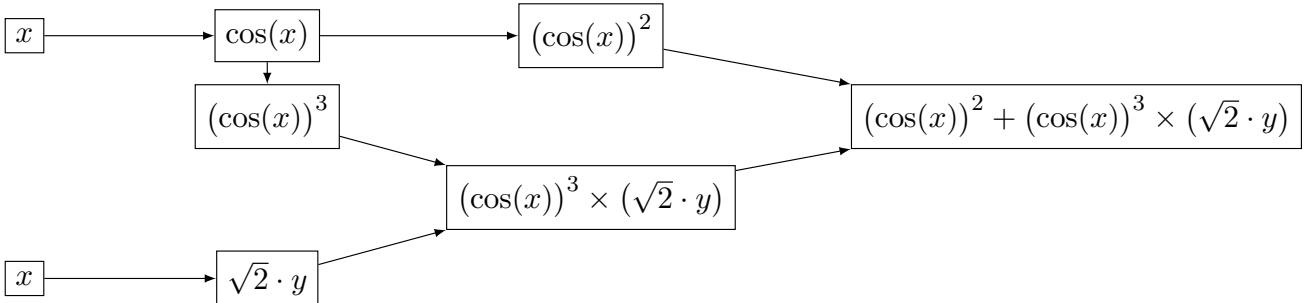
$$f : \begin{cases} \mathbb{R}^2 - \rightarrow \mathbb{R} \\ (x, y) \mapsto (\cos x)^2 + (\cos x)^3 \times (\sqrt{2} \cdot y) \end{cases}$$

On peut l'écrire comme une composition $f = f^4 \circ f^3 \circ f^2 \circ f^1$ avec :

$$\begin{aligned} f^1 &: (x, y) \mapsto (\cos x, \sqrt{2} \cdot y) \\ f^2 &: (x, y) \mapsto (x^2, x^3, y) \\ f^3 &: (x, y, z) \mapsto (x, y \times z) \\ f^4 &: (x, y) \mapsto x + y \end{aligned}$$

Propager la différentielle des fonctions f^i pour $1 \leq i \leq 4$ est une perte de temps. En effet, on manipule des fonctions qui ne font pas usage de certaines de leur variables, et on différencie des fonctions potentiellement linéaires, comme la fonction f^4 . On peut représenter plus finement les suites d'instructions mathématiques comme des graphes acycliques.

Par exemple, la fonction f ci-dessus se dessine comme suit :



On décrit un tel graphe comme un tableau T de sommets. Chaque sommet u contient deux informations : la fonction élémentaire qu'il représente et le tableau de ses prédécesseurs (les sommets v tels qu'il y a une arête de v à u). Comme le graphe est acyclique, on peut faire l'hypothèse que s'il y a une arête de v à u , l'indice de v dans T est strictement inférieur à l'indice de u dans T . On suppose par ailleurs que l'on ne manipule que des fonctions élémentaires à valeurs réelles $\mathbb{R}^n \rightarrow \mathbb{R}$.

Ces fonctions élémentaires sont alors représentées par une paire de type

```
type fonction_elementaire =
  (float array -> float) * (float array -> float array);;
```

où le deuxième élément représente la différentielle du premier élément (comme la fonction est à valeurs réelles, sa différentielle en un point est représentée directement par un vecteur et non par une matrice). On définit donc les types OCaml suivants :

```
type sommet_avant =
  {fct : fonction_elementaire; pred : int array};;

type graphe_avant = sommet_avant array;;
```

Lors du calcul par propagation avant de la différentielle d'une fonction, on parcourt le graphe de cette fonction *de la gauche vers la droite*, c'est-à-dire des sommets correspondant aux variables au sommet correspondant à la fonction complète, en traitant les prédecesseurs d'un sommet avant celui-ci. On stocke dans un *tableau de propagation* avant, au fur et à mesure, les valeurs de chaque fonction élémentaire et de sa différentielle en les points pertinents. Pour les sommets d'un graphe de fonction correspondant aux variables, le contenu du champ fct n'importe pas, car ils n'ont pas de prédecesseurs.

On peut par exemple coder le graphe de la fonction exemple f comme suit :

```

let mafonction =
  let id t = t.(0) in
  let did t = [|1.0|] in
  let n0 = {fct = (id, did); pred = [| |]} in
  let costab t = (cos (t.(0))) in
  let dcostab t = [| -1.0 *. sin (t.(0)) |] in
  let n2 = {fct = (costab, dcostab); pred = [|0|]} in
  let carre t = (t.(0)) ** 2.0 in
  let dcarre t = [| 2.0 *. t.(0) |] in
  let n3 = {fct = (carre, dcarre); pred = [|2|]} in
  let cube t = (t.(0)) ** 3.0 in
  let dcube t = [| 3.0 *. (t.(0)) **. 2.0 |] in
  let n4 = {fct = (cube, dcube); pred = [|2|]} in
  let scaler t = (sqrt 2.0) *. (t.(0)) in
  let dscaler t = [| sqrt 2.0 |] in
  let n5 = {fct = (scaler, dscaler); pred = [|1|]} in
  let mult t = (t.(0)) *. (t.(1)) in
  let dmult t = [| t.(1); t.(0) |] in
  let n6 = {fct = (mult, dmult); pred = [|4; 5|]} in
  let plus t = (t.(0)) +. (t.(1)) in
  let dplus t = [| 1.0; 1.0 |] in
  let n7 = {fct = (plus, dplus); pred = [|3; 6|]} in
  [| n0; n2; n3; n4; n5; n6; n7 |];

```

Question IV.1. On souhaite calculer la valeur de la différentielle de la fonction f en $(\frac{\pi}{4}, 1)$ évaluée en le point $(1, 1)$. Pour ce faire, calculer à la main l'ensemble des valeurs du tableau de propagation avant en détaillant le calcul. Les trois premières lignes du tableau sont :

Indice	Fonction	Valeur	Définition
0	x	$\frac{\pi}{4}$	1
1	y	1	1
2	$\cos(x)$	$\cos(\frac{\pi}{4}) = \frac{\sqrt{2}}{2}$	$(-\sin(\frac{\pi}{4})) \times 1 = -\frac{\sqrt{2}}{2}$

Question IV.2. Donner une fonction intermédiaire OCaml

```
collecte : (float * float) array -> int array
          -> float array * float array
```

tenant en entrée un tableau de propagation avant A (c'est-à-dire un tableau de couples (valeur fonction, valeur_déférentielle)), un tableau I de n indices de sommets et renvoyant un couple formé :

- d'un vecteur des n valeurs de fonctions dans A correspondant aux sommets de I ;
- d'un vecteur des n valeurs de déférentielles dans A correspondant aux sommets de I .

Pour l'exemple de la fonction f , les trois premiers éléments du tableau A sont de la forme :

```
[| (0.785398163397448279, 1 .);
  (1., 1.);
  (0.707106781186547573, -0.707106781186547573) |]
```

Question IV.3. En utilisant la fonction `collecte`, écrire une fonction Ocaml

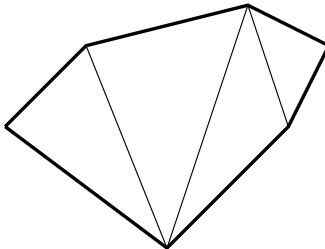
```
propagation_avant : graphe_avant -> float array
                     -> float array -> float
```

tenant en entrée un graphe représentant une fonction f , un tableau représentant un point \mathbf{x} , un tableau représentant un vecteur \mathbf{v} , et qui calcule par propagation avant le flottant $D_{\mathbf{x}}f(\mathbf{v})$.

Question IV.4. On voudrait traiter de manière particulière les fonctions linéaires, comme la dernière opération $(x, y) \mapsto x + y$ et la première opération $z \mapsto \sqrt{2} \cdot z$ dans l'exemple ci-dessus, car elles sont leur propres déférentielles et peuvent donc être traitées plus simplement qu'une fonction arbitraire. Comment peut-on modifier la structure du graphe de la fonction dans ce sens ?

Partie V

Une triangulation d'un polygone correspond à la partition du polygone en triangles dont les sommets sont des sommets du polygone.



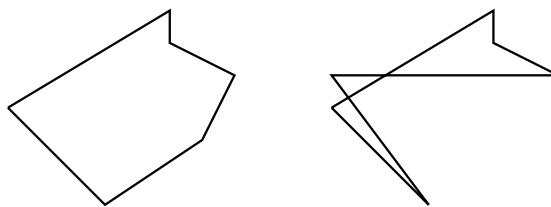
Question V.1. On considère des polygones convexes dont chaque sommet est étiqueté par un poids (entier naturel non nul). On définit le coût d'un triangle comme le produit des poids des trois sommets, et le coût d'une triangulation d'un polygone comme la somme des coûts de tous ses triangles. Montrer que le nombre minimum de multiplications de flottants nécessaires pour calculer la multiplication de $n \geq 2$ matrices (au sens de la partie **II**) est égal au coût minimum d'une triangulation d'un certain polygone convexe.

Question V.2. On reprend les notations de la partie II : soit une suite de matrices de flottants $(B^i)_{1 \leq i \leq n}$ et k_0, \dots, k_n la suite d'entiers tels que $B^i \in \mathcal{M}_{k_{i-1}, k_i}(\mathbb{R})$. On suppose de plus $k_0 = k_n$. Montrer que le nombre minimal de multiplications de flottants nécessaires au calcul de $B^1 \times \dots \times B^{n-1}$ est égal au nombre minimal de multiplications de flottants nécessaires au calcul de $B^n \times B^1 \times \dots \times B^{n-2}$.

Oublions désormais les poids sur les sommets de la triangulation d'un polygone. On représente un polygone comme un tableau de points dans $\mathbb{R} \times \mathbb{R}$. Il y a une arête entre deux sommets successifs du tableau, ainsi qu'entre la première et la dernière valeur.

```
type polygone = (float * float) array;;
```

Un polygone est dit simple si deux arêtes non consécutives ne se croisent pas, et deux arêtes consécutives n'ont en commun que l'un de leurs sommets. On ne considéra que des polygones simples et on ne cherchera pas à vérifier leur simplicité.



Polygone simple Polygone non-simple

On formalise la triangulation d'un polygone comme une liste d'arêtes à ajouter au polygone. Les sommets du polygone sont désignés par leur place dans le tableau représentant le polygone : le premier sommet est celui en position 0 dans le tableau, etc. Une arête entre le i -ème sommet du polygone et le j -ème sommet est décrite comme un couple d'entiers (i, j) avec $i < j$.

Question V.3. Donner une fonction OCaml testant la convexité d'un polygone simple :

```
est_convexe : polygone -> bool
```

Indication : On pourra utiliser l'orientation de produits vectoriels de vecteurs bien choisis.

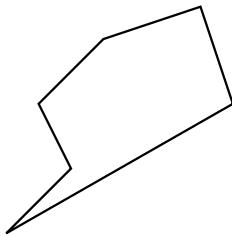
Question V.4. Combien d'arêtes la triangulation d'un polygone convexe à n cotés introduit-elle ? Justifier.

Question V.5. Donner une fonction OCaml

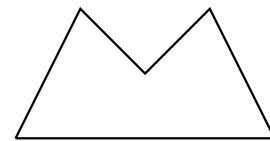
```
triangule_convexe : polygone -> (int * int) list
```

qui prend en entrée un polygone supposé convexe et renvoie une triangulation de ce polygone sous la forme d'une liste d'arêtes en un temps linéaire en le nombre de sommets du polygone.

On considère désormais des polygones non-convexes. Un polygone simple est dit strictement monotone par rapport à l'axe des ordonnées si toute ligne horizontale ne coupe le polygone qu'en au plus deux points.



Polygone simple strictement monotone



Polygone simple non strictement monotone

Ainsi, si on imagine un point se déplaçant le long d'un polygone simple, de son sommet le plus haut vers son sommet le plus bas, il ne fera que descendre ou se déplacer horizontalement, jamais il ne remontera.

Question V.6. Donner une fonction OCaml

```
separe_polygone : polygone -> int list * int list
```

qui prend en entrée un polygone simple strictement monotone et sépare ses sommets (représentés par leurs indices) en deux listes, l'une contenant les sommets du plus haut (inclus) vers le plus bas (exclus) en suivant l'un des deux chemins à partir du sommet le plus haut, l'autre les sommets du plus haut (exclus) vers le plus bas (inclus) en suivant l'autre chemin. Ainsi, tout sommet du polygone doit être classé une et une seule fois dans l'une des listes en sortie, et chaque liste est triée par ordre décroissant de deuxième coordonnée. La fonction donnée doit être linéaire en le nombre de sommets du polygone.

Question V.7. Proposer un algorithme qui prend en entrée un polygone supposé simple strictement monotone et renvoie une triangulation de ce polygone en un temps linéaire en le nombre de sommets du polygone. Cet algorithme pourra commencer par utiliser l'algorithme implémenté par la fonction `separe_polygone`.

Fin du sujet.

ECOLES NORMALES SUPERIEURES**CONCOURS D'ADMISSION 2022**

Le concours d'admission à l'ENS est un concours d'admission à l'ENS et à l'ENSAE.

La date limite de dépôt des candidatures est le 15 mai 2022.

Le concours d'admission à l'ENS est un concours d'admission à l'ENS et à l'ENSAE.

VENDREDI 29 AVRIL 2022
14h00 - 18h00

FILIÈRE MP - Epreuve n° 10
INFO-MATHEMATIQUES (ULSR)

On fixe un ensemble fini Σ appelé alphabet dans lesquels sont apposées lettres. Un mot est une suite finie de lettres. Un langage est un ensemble de mots. L'ensemble de tous les mots est noté Λ^* . On introduit variables et fonctions pour décrire ces langages. Les variables w, u, v, \dots peuvent décrire les mots. La case vide est notée λ , la concaténation de deux mots est notée wv ; une autre sera parfois confondue avec le nom d'une lettre enregistrée. Par exemple, un mot w dont constitué de la lettre a s'appelle un mot a .

Un automate fini (AFN) est un triplet (Σ, δ, q_0) .

δ est un ensemble d'états.

$\rightarrow \delta : \Sigma \times \{0, 1\} \rightarrow \{0, 1\}$ est une fonction valable pour chaque état, s'il est acceptant (1) ou pas (0).
 $\rightarrow \delta : \Sigma \times \{0, 1\} \rightarrow \Sigma$ est la fonction de transition, toutes unités étant aussi état et dépendent logiquement de la lecture d'une lettre dans un état donné.

Nous qui nous faisons le droit de ne pas donner un état initial. Ainsi donné un automate, nous nous intéressons aux langages reconnaissables à partir de chacun des états de cet automate. On appelle une automate probabiliste, toutes les transitions sur la gauche, en utilisant des probabilités pour décrire les états, correspondant à des étapes égales pour

Durée : 4 heures

L'utilisation des calculatrices n'est pas autorisée pour cette épreuve

Automates, points-fixes, et équivalence de langage

Le sujet comprend douze pages, numérotées de 1 à 12.

Début de l'épreuve.

La première partie introduit les automates et permet de se familiariser avec la notion d'équivalence de langage entre états d'un automate. Un algorithme pour vérifier l'équivalence de deux états y est proposé.

La seconde partie met en place des outils (théorèmes de point fixe, clôtures) qui permettent dans la troisième partie de prouver la correction de cet algorithme, puis d'en proposer des optimisations.

La dernière partie est plus courte, on y considère la question du calcul de la relation d'équivalence entre états dans sa globalité, et on y utilise les outils de la seconde partie pour proposer un algorithme.

Les parties 1 et 2 sont indépendantes ; il est conseillé de les traiter en premier car les parties 3 et 4 en dépendent. Il est permis d'admettre les réponses à certaines questions pour répondre aux suivantes.

1 Automates

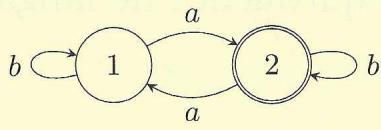
On fixe un ensemble fini A appelé *alphabet*, dont les éléments sont appelés *lettres*. Un *mot* est une suite finie de lettres. Un *language* est un ensemble de mots. L'ensemble de tous les mots est noté A^* . On utilise les variables $a, b, c \dots$ pour dénoter les lettres et les variables $u, v, w \dots$ pour dénoter les mots. Le mot vide est noté ϵ ; la concaténation de deux mots u, v est notée uv ; une lettre sera parfois confondue avec le mot d'une lettre correspondant. Par exemple, au est le mot constitué de la lettre a suivie du mot u .

Un *automate (déterministe)* est un triplet $\langle X, o, \delta \rangle$ où :

- X est un ensemble d'*états* ;
- $o : X \rightarrow \{0, 1\}$ est une fonction indiquant pour chaque état, s'il est acceptant (1) ou non (0) ;
- $\delta : X \times A \rightarrow X$ est la fonction de transition, totale, indiquant vers quel état se déplacer lors de la lecture d'une lettre dans un état donné.

Notons que nous faisons le choix de ne pas désigner un état initial ; étant donné un automate, nous nous intéresserons aux langages reconnus à partir de chacun des états de cet automate. On représente un automate graphiquement comme ci-dessous sur la gauche, en utilisant des cercles pour dénoter les états, à bordures doubles quand ils sont acceptants, et des flèches étiquetées par

les lettres de l'alphabet pour dénoter les transitions. Ici, il s'agit d'un automate dont l'ensemble d'états est $\{1, 2\}$, sur l'alphabet $\{a, b\}$; les fonctions o et δ correspondantes sont données sur la droite.



x	$\delta(x, a)$	$\delta(x, b)$	$o(x)$
1	2	1	0
2	1	2	1

On étend la fonction de transition aux mots comme suit, par récurrence sur les mots :

$$\begin{aligned} \delta^* : X \times A^* &\rightarrow X \\ \left\{ \begin{array}{l} \delta^*(x, \epsilon) \stackrel{\text{def}}{=} x \\ \delta^*(x, au) \stackrel{\text{def}}{=} \delta^*(\delta(x, a), u) \end{array} \right. \end{aligned}$$

(Dans ce sujet, on note $\stackrel{\text{def}}{=}$ l'égalité par définition.) Intuitivement, $\delta^*(x, u)$ est l'état atteint en lisant le mot u à partir de l'état x . On en déduit la fonction associant à chaque état x de l'automate le langage $L(x)$ reconnu par cet état.

$$L(x) \stackrel{\text{def}}{=} \{u \mid o(\delta^*(x, u)) = 1\}$$

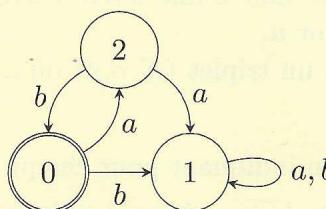
Notons que par définition, on a pour tout état x , toute lettre a , et tout mot u :

$$\begin{aligned} \epsilon \in L(x) &\Leftrightarrow o(x) = 1 \\ au \in L(x) &\Leftrightarrow u \in L(\delta(x, a)) \end{aligned}$$

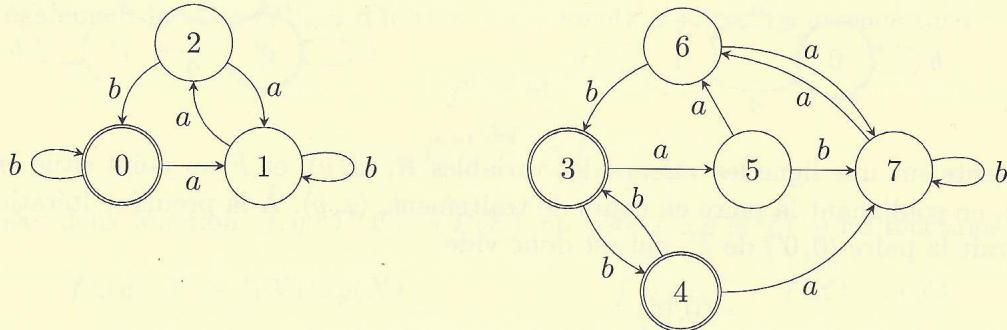
Pour les questions de cette partie (1.1 à 1.5), il n'est pas demandé de justifier les réponses.

Question 1.1 Pour l'automate à deux états dessiné en haut de la page, donner une description intuitive des langages reconnus par les états 1 et 2 (c'est à dire, $L(1)$ et $L(2)$).

Question 1.2 Donner une description précise des langages reconnus par les trois états de l'automate suivant.



Question 1.3 Considérons l'automate suivant. Bien que cet automate contienne deux parties disjointes, on le considère comme un automate à huit états. Donner une description intuitive des langages reconnus par les états 0, 1, et 2 de l'automate. Les états 3 à 7 reconnaissent aussi ces trois langages ; quelle est la correspondance ?



Deux états x, y sont équivalents, noté $x \simeq y$, lorsqu'ils reconnaissent le même langage ($L(x) = L(y)$).

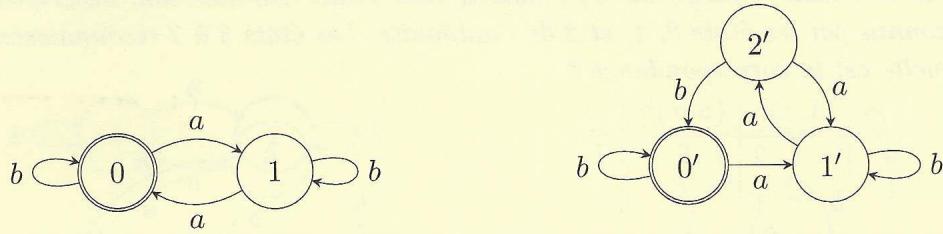
Question 1.4 Décrire la relation \simeq sur l'ensemble d'états $[0; 7]$ pour l'automate de la question 1.3.

Nous allons étudier l'algorithme 1 ci-dessous, permettant de tester l'équivalence de deux états dans un automate donné. On ignore dans un premier temps les invariants I1 et I2, et I3 qui sera défini à la question 3.7. À la ligne 2.2, la sémantique de l'instruction `continuer` consiste à revenir à l'entrée de la boucle (ligne 2), sans exécuter les lignes 2.3 à 2.5. On suppose que la variable F est implémentée par une file “premier entré premier sorti”, et que pour la boucle interne (ligne 2.4), les éléments sont considérés selon l'ordre alphabétique.

Algorithme 1 Un algorithme pour tester l'équivalence de deux états x_0 et y_0 d'un automate $\langle X, o, \delta \rangle$ donné.

- (1) $R := \emptyset$; $F := \{(x_0, y_0)\}$;
 - (2) tant que F n'est pas vide,
 - // I1: $\langle x_0, y_0 \rangle \in R \cup F$
 - // I2: $R \subseteq p(R \cup F)$
 - // I3:
 - (2.1) extraire une paire $\langle x, y \rangle$ de F ;
 - (2.2) si $\langle x, y \rangle \in R$, continuer;
 - (2.3) si $o(x) \neq o(y)$, renvoyer faux;
 - (2.4) pour tout $a \in A$, ajouter $\langle \delta(x, a), \delta(y, a) \rangle$ à F ;
 - (2.5) ajouter $\langle x, y \rangle$ à R ;
 - (3) renvoyer vrai;
-

Nous allons exécuter cet algorithme en partant de la paire d'états $\langle 0, 0' \rangle$ de l'automate suivant :



On représente sur une ligne les valeurs des variables R , $\langle x, y \rangle$, et F au point situé entre les lignes 2.1 et 2.2, en soulignant la paire en cours de traitement, $\langle x, y \rangle$. À la première itération, R est vide et on extrait la paire $\langle 0, 0' \rangle$ de F , qui est donc vide :

$$\underline{\langle 0, 0' \rangle}$$

- On vérifie que $o(0) = o(0')$, et on insère les successeurs de la paire $\langle 0, 0' \rangle$ à la fin de F : $\langle 1, 1' \rangle$ selon la lettre a , puis $\langle 0, 0' \rangle$ selon la lettre b . La paire soulignée passe dans R et on souligne la nouvelle paire à traiter :

$$\langle 0, 0' \rangle \ \underline{\langle 1, 1' \rangle} \ \langle 0, 0' \rangle$$

on vérifie que $o(1) = o(1')$, on insère les successeurs de la paire $\langle 1, 1' \rangle$ ($\langle 0, 2' \rangle$ puis $\langle 1, 1' \rangle$) ; la paire soulignée passe dans R et on souligne la suivante :

$$\langle 0, 0' \rangle, \langle 1, 1' \rangle \ \underline{\langle 0, 0' \rangle} \ \langle 0, 2' \rangle, \langle 1, 1' \rangle$$

cette paire est déjà dans R , on la barre et on passe à la suivante :

$$\langle 0, 0' \rangle, \langle 1, 1' \rangle, \cancel{\langle 0, 0' \rangle}, \underline{\langle 0, 2' \rangle} \ \langle 1, 1' \rangle$$

on a $o(0) \neq o(2')$, l'algorithme renvoie faux.

Question 1.5 Exécuter l'algorithme 1 pour la paire d'états $\langle 0, 3 \rangle$ de l'automate de la question 1.3. Utiliser la représentation précédente, et ne donner que la dernière ligne. Faire de même en partant de la paire $\langle 2, 5 \rangle$.

Nous démontrerons la correction de cet algorithme dans la partie 3.

2 Fonctions d'ensembles, de relations

Soit E un ensemble, potentiellement infini. On note $\mathcal{P}(E)$ l'ensemble des parties de E . Notons que pour toutes parties $X, Y \subseteq E$, on a $X \subseteq Y$ si et seulement si $X \cup Y = Y$.

Une suite $(X_i)_{i \in \mathbb{N}}$ de parties de E est dite *croissante* (resp. *décroissante*) si pour tout indice i , $X_i \subseteq X_{i+1}$ (resp. $X_{i+1} \subseteq X_i$).

On s'intéresse aux fonctions de $\mathcal{P}(E)$ dans lui-même. On note id la fonction identité, et $f \circ g$ la composition de deux fonctions :

$$\text{id} : X \mapsto X \quad f \circ g : X \mapsto f(g(X))$$

On définit également la suite $(f^i)_{i \in \mathbb{N}}$ d'itérées d'une fonction f , par récurrence sur i :

$$\begin{aligned} f^0 &\stackrel{\text{def}}{=} \text{id} \\ f^{i+1} &\stackrel{\text{def}}{=} f \circ f^i \end{aligned}$$

Étant données deux fonctions $f, g : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$, on note $f \cup g$ et $f \cap g$ les fonctions

$$f \cup g : X \mapsto f(X) \cup g(X) \quad f \cap g : X \mapsto f(X) \cap g(X)$$

Plus généralement, étant donnée une famille $(f_i)_{i \in I}$ de fonctions, on note $\bigcup_{i \in I} f_i$ et $\bigcap_{i \in I} f_i$ les fonctions

$$\bigcup_{i \in I} f_i : X \mapsto \bigcup_{i \in I} f_i(X) \quad \bigcap_{i \in I} f_i : X \mapsto \bigcap_{i \in I} f_i(X)$$

Une fonction $f : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ est dite

- *croissante* si pour tous $X, Y \subseteq E$ tels que $X \subseteq Y$, on a $f(X) \subseteq f(Y)$.
- *continue* si pour toute suite croissante $(X_i)_{i \in \mathbb{N}}$ de parties de E , on a $f(\bigcup_{i \in \mathbb{N}} X_i) = \bigcup_{i \in \mathbb{N}} f(X_i)$.
- *co-continue* si pour toute suite décroissante $(X_i)_{i \in \mathbb{N}}$ de parties de E , on a $f(\bigcap_{i \in \mathbb{N}} X_i) = \bigcap_{i \in \mathbb{N}} f(X_i)$.

La fonction identité est trivialement croissante, continue, et co-continue. De même, la composée de deux fonctions croissantes (resp. continues, co-continues) est croissante (resp. continue, co-continue), et l'union d'une famille de fonctions croissantes (resp. continues, co-continues) est croissante (resp. continue, co-continue).

À partir de la question suivante, toutes les réponses doivent être soigneusement justifiées.

Question 2.1 Montrer que toute fonction continue est croissante.

Question 2.2 Montrer que toute fonction co-continue est croissante.

Question 2.3 Une fonction croissante est-elle toujours continue ? co-continue ? Justifier vos réponses.

Etant donnée une fonction f , une partie $X \subseteq E$ est :

- un *post-point fixe* (de f) si $X \subseteq f(X)$,
- un *pré-point fixe* (de f) si $f(X) \subseteq X$,
- un *point fixe* (de f) si $X = f(X)$.

Question 2.4 (i) Montrer que toute fonction admet un post-point fixe et un pré-point fixe.

(ii) Donner une fonction qui n'admette pas de point fixe.

2.1 Plus grand point fixe

On va démontrer que toute fonction croissante admet un plus grand point fixe. On fixe pour cela une fonction croissante $f : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$.

Question 2.5 Montrer que l'union d'une famille de post-points fixes est un post-point fixe.

Soit νf l'union de tous les post-points fixes de f :

$$\nu f \stackrel{\text{def}}{=} \bigcup_{X \subseteq f(X)} X$$

Par la question précédente, νf est un post-point fixe : $\nu f \subseteq f(\nu f)$.

Question 2.6 (i) Montrer que νf est aussi un pré-point fixe.

(ii) En déduire que c'est le plus grand point fixe (au sens de l'inclusion).

Question 2.7 Montrer que f admet aussi un plus petit point fixe.

Si l'on suppose de plus que la fonction f est co-continue, on peut caractériser son plus grand point fixe autrement. Remarquons tout d'abord que $(f^i(E))_{i \in \mathbb{N}}$ est une suite de parties de E .

Question 2.8 (i) Montrer que la suite $(f^i(E))_{i \in \mathbb{N}}$ est décroissante.

(ii) Montrer que si f est co-continue, alors $\nu f = \bigcap_{i \in \mathbb{N}} f^i(E)$.

2.2 Plus petite clôture

On ordonne les fonctions point à point : f est contenue dans g , noté $f \subseteq g$, si pour tout $X \subseteq E$, $f(X) \subseteq g(X)$.

Une fonction f est :

- *extensive* si $\text{id} \subseteq f$ (c'est à dire, $\forall X, X \subseteq f(X)$);
- *saturante* si $f \circ f \subseteq f$ (c'est à dire, $\forall X, f(f(X)) \subseteq f(X)$);
- une *clôture* si elle est croissante, extensive, et saturante.

Etant donnée une fonction f , on pose $f^\omega \stackrel{\text{def}}{=} \bigcup_{i \in \mathbb{N}} (f \cup \text{id})^i$.

Question 2.9 (i) Montrer que f^ω est extensive et contient f .

(ii) Montrer que si f est continue, alors f^ω est une clôture.

Question 2.10 Montrer que si f est continue, alors pour tout X , $f^\omega(X)$ est le plus petit pré-point fixe de f contenant X .

Question 2.11 Déduire des questions précédentes que lorsque f est continue, f^ω est la plus petite clôture contenant f .

2.3 Relations et fonctions de relations

On fixe dans cette partie un ensemble I . Une *relation* est une partie de $I \times I$. On s'intéresse aux fonctions des relations dans les relations. Autrement dit, on spécialise la partie précédente au cas $E = I \times I$. On note 1 la relation identité, $R \cdot S$ la composition de deux relations R, S , et R^\top la transposée d'une relation R :

$$\begin{aligned} 1 &\stackrel{\text{def}}{=} \{\langle i, i \rangle \mid i \in I\} \\ R \cdot S &\stackrel{\text{def}}{=} \{\langle i, k \rangle \mid \exists j \in I, \langle i, j \rangle \in R \wedge \langle j, k \rangle \in S\} \\ R^\top &\stackrel{\text{def}}{=} \{\langle j, i \rangle \mid \langle i, j \rangle \in R\} \end{aligned}$$

Etant donnée une relation R , on notera parfois $x R y$ pour $\langle x, y \rangle \in R$.

Question 2.12 Comment qualifie-t-on usuellement une relation R telle que $1 \subseteq R$? telle que $R^\top \subseteq R$? telle que $R \cdot R \subseteq R$?

Soient $r, s, t : \mathcal{P}(I \times I) \rightarrow \mathcal{P}(I \times I)$ les trois fonctions suivantes :

$$r : R \mapsto 1 \qquad s : R \mapsto R^\top \qquad t : R \mapsto R \cdot R$$

Question 2.13 Qu'est-ce qu'un pré-point fixe pour r ? pour s ? pour t ? pour $r \cup t$? pour $r \cup s \cup t$?

Question 2.14 Montrer que les trois fonctions r , s et t sont continues.

Au vu des questions 2.10 et 2.11, on appelle usuellement la fonction t^ω clôture transitive : c'est une clôture, et $t(R)$ est la plus petite relation transitive contenant R . De manière similaire, les fonctions $(r \cup t)^\omega$ et $(r \cup s \cup t)^\omega$ sont respectivement les fonctions de clôture réflexive-transitive et réflexive-symétrique-transitive.

3 Équivalence de deux états

On cherche désormais des algorithmes permettant de tester l'équivalence de deux états dans un automate fini donné. Nous allons tout d'abord montrer que l'algorithme proposé en partie 1 est correct, et de complexité quadratique. On fixe dans toute cette partie un automate $\langle X, o, \delta \rangle$, dont l'ensemble d'états X est fini, de taille n .

3.1 Un algorithme quadratique

Soit $p : \mathcal{P}(X \times X) \rightarrow \mathcal{P}(X \times X)$ la fonction suivante entre relations sur l'ensemble X des états :

$$p : R \mapsto \{\langle x, y \rangle \mid o(x) = o(y) \wedge \forall a \in A, \delta(x, a) R \delta(y, a)\}$$

Pour toutes relations R, S , en déroulant cette définition, on a

$$S \subseteq p(R) \text{ si et seulement si } \forall x, y \in X, x S y \Rightarrow o(x) = o(y) \wedge \forall a \in A, \delta(x, a) R \delta(y, a)$$

On appelle *bisimulation* tout post-point fixe de p : toute relation R telle que $R \subseteq p(R)$.

Question 3.1 Donner la plus petite bisimulation contenant la paire d'états $\langle 0, 3 \rangle$ de l'automate de la question 1.3. Montrer qu'il n'existe pas de bisimulation contenant la paire d'états $\langle 2, 5 \rangle$.

Question 3.2 Montrer que la fonction p est croissante.

D'après la question 2.5, l'union de toutes les bisimulations est une bisimulation : c'est le plus grand point fixe νp de la fonction p .

Question 3.3 (i) Montrer que la relation d'équivalence entre états (\simeq) est une bisimulation.

(ii) Montrer que pour toute bisimulation R , $R \subseteq \simeq$.

(iii) En déduire que la plus grande bisimulation est l'équivalence de langage : $\nu p = \simeq$.

Afin de prouver que deux états sont équivalents, il suffit donc de trouver une bisimulation contenant ces deux états. Inversement, pour prouver que deux états ne sont pas équivalents, il suffit de trouver un mot accepté par l'un mais pas l'autre. C'est exactement ce que fait l'algorithme de la partie 1.

Question 3.4 Montrer que le premier invariant ($I1$) est satisfait à l'entrée de la boucle, et préservé à chaque itération.

Question 3.5 Faire de même pour le second invariant ($I2$).

Question 3.6 Que peut-on en déduire à la sortie de la boucle (ligne 3) ?

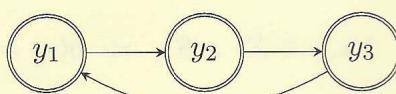
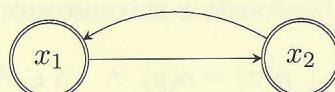
Question 3.7 Donner un troisième invariant ($I3$) impliquant que les états de départ ne sont pas équivalents lorsque l'on sort de la boucle prématûrément (ligne 2.3). Démontrer sa préservation. (Un tel invariant ne s'exprime pas de manière aussi concise que $I2$, mais reste relativement simple.)

Question 3.8 Donner une quantité décroissant strictement à chaque itération non-triviale (lorsque l'instruction `continuer` n'est pas exécutée). En déduire la terminaison de l'algorithme.

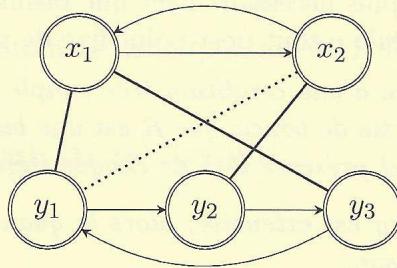
Question 3.9 En déduire que l'algorithme est correct et a une complexité dans le pire cas en $O(kn^2)$ où n est le nombre d'états de l'automate et k le nombre de lettres de l'alphabet. (On ne comptera que le nombre d'appels aux fonctions o et δ .)

Question 3.10 La variable F peut être implementée par n'importe quelle structure de données de file (premier entrant premier sorti) ou de pile (premier entrant dernier sorti). Ces choix correspondent à deux stratégies d'exploration de l'automate ; lesquelles ? Dans quel cas ce choix a une importance sur le temps d'exécution de l'algorithme ? Donner un exemple.

Considérons l'automate suivant, sur un alphabet réduit à une lettre de telle sorte que l'on peut omettre l'étiquetage des transitions, et dont tous les états sont acceptants.



Si l'on commence l'exécution de l'algorithme précédent à partir de la paire $\langle x_1, y_1 \rangle$, on insère successivement les paires $\langle x_1, y_1 \rangle$, $\langle x_2, y_2 \rangle$, $\langle x_1, y_3 \rangle$, ... ce que l'on peut représenter comme ci-dessous, la paire représentée en pointillée étant la prochaine devant être traitée :



Question 3.11 Représenter comme ci-dessus la relation R obtenue lorsque l'algorithme termine. Combien de paires contient cette relation ?

Question 3.12 En généralisant l'exemple précédent, donner une séquence d'automates et de paires d'états dans ces automates démontrant que la borne quadratique en le nombre d'états peut effectivement être atteinte, asymptotiquement. On se contentera de traiter le cas d'un alphabet à une seule lettre, et on considérera comme précédemment des automates dont tous les états sont acceptants.

3.2 Un algorithme linéaire ?

Soit $g : \mathcal{P}(X \times X) \rightarrow \mathcal{P}(X \times X)$ une fonction. L'algorithme 2 est une variante de l'algorithme 1, où la seule modification apparaît en ligne 2.2 : au lieu de tester $\langle x, y \rangle \in R$, on teste $\langle x, y \rangle \in g(R)$. L'algorithme 1 s'obtient donc en prenant la fonction identité pour g .

Algorithme 2 Optimisation générique de l'algorithme 1, paramétrée par la fonction g utilisée à la ligne 2.2.

```

(1)  $R := \emptyset$ ;  $F := \{\langle x_0, y_0 \rangle\}$ ;
(2) tant que  $F$  n'est pas vide,
    // I1:  $\langle x_0, y_0 \rangle \in R \cup F$ 
    // I2':
    // I3:
    (2.1) extraire une paire  $\langle x, y \rangle$  de  $F$ ;
    (2.2) si  $\langle x, y \rangle \in g(R)$ , continuer;
    (2.3) si  $o(x) \neq o(y)$ , renvoyer faux;
    (2.4) pour tout  $a \in A$ , ajouter  $\langle \delta(x, a), \delta(y, a) \rangle$  à  $F$ ;
    (2.5) ajouter  $\langle x, y \rangle$  à  $R$ ;
(3) renvoyer vrai;
  
```

Si l'on choisit une fonction extensive pour g , ce nouvel algorithme peut terminer plus rapidement : certaines paires ne sont plus insérées dans la file F . Cela peut bien entendu fausser le résultat : par exemple, si g est la fonction constante valant $X \times X$ partout, l'algorithme renvoie toujours vrai.

Nous allons proposer des conditions sur la fonction g permettant d'assurer la correction d'une telle variante, puis nous chercherons des fonctions g satisfaisant ces conditions.

L'invariant I2 précédemment utilisé n'est plus valide pour cet algorithme. De même, à la sortie de la boucle principale, R n'est plus nécessairement une bisimulation.

On appelle *bisimulation modulo g* tout post-point fixe de $p \circ g$.

A C Aoy(R)

Question 3.13 Sous l'hypothèse d'une condition très simple sur la fonction g , donner un nouvel invariant I2' garantissant en sortie de boucle que R est une bisimulation modulo g . Démontrer que I2' est satisfait avant la boucle et préservé lors de chaque itération.

Question 3.14 Montrer que si g est extensive, alors la quantité donnée à la question 3.8 permet toujours de garantir la terminaison.

Il nous faut maintenant trouver une condition garantissant que toute bisimulation modulo g est bien contenue dans l'équivalence de langage.

Nous nous plaçons pour cela à nouveau dans le cadre de la partie 2 : on fixe une fonction croissante $f : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$, dont le plus grand point fixe nous intéresse.

Une fonction $h : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ est dite *compatible* (avec f) si h est croissante et $h \circ f \subseteq f \circ h$. L'identité est trivialement compatible.

Question 3.15 Montrer que la composition de deux fonctions compatibles est compatible.

Question 3.16 Montrer que l'union d'une famille de fonctions compatibles est compatible.

Question 3.17 Montrer que si h est compatible, alors il en est de même de h^ω .

Question 3.18 Soit h une clôture compatible avec f .

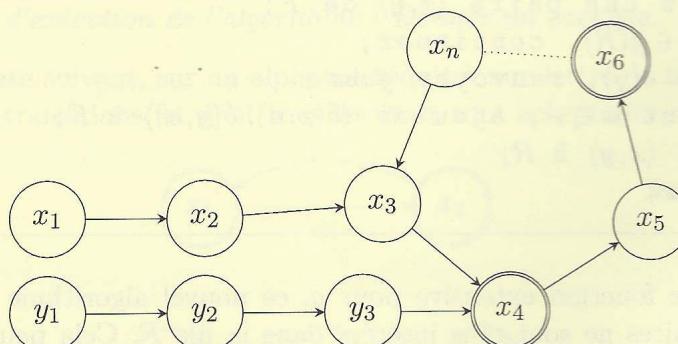
- (i) Montrer que pour tout post-point fixe X de $f \circ h$, $h(X)$ est un post-point fixe de f .
- (ii) En déduire que $\nu(f \circ h) \subseteq \nu f$.
- (iii) L'inclusion réciproque est-elle vérifiée ?

On revient maintenant dans le cadre des automates déterministes.

Question 3.19 En utilisant les questions précédentes, donner une condition simple sur la fonction g garantissant la correction de l'algorithme optimisé.

Nous allons maintenant exhiber des fonctions spécifiques pouvant être employées pour g .

Considérons un automate de la forme suivante :



L'alphabet n'a qu'une seule lettre, c'est pourquoi les transitions ne sont pas étiquetées. Il y a $n+3$ états dont seulement deux acceptants.

Question 3.20 Combien d'itérations nécessite l'algorithme 1 pour prouver que x_1 et y_1 sont équivalents ?

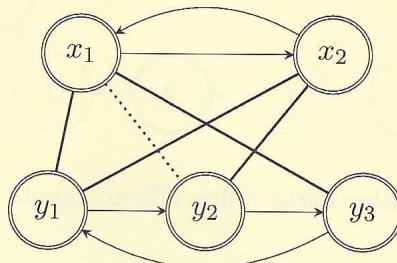
Question 3.21 Proposer une optimisation permettant de terminer en temps constant sur cette famille d'entrées, et prouver sa correction via le cadre précédemment mis en place.

Reconsidérons maintenant l'automate de la question 3.11. Au début de la cinquième itération, on a :

$$R = \{\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \langle x_1, y_3 \rangle, \langle x_2, y_1 \rangle\}$$

$$F = \{\langle x_1, y_2 \rangle\}$$

ce qui peut être représenté ainsi :



Question 3.22 Proposer une optimisation permettant de terminer lors de cette cinquième itération et prouver sa correction via le cadre précédemment mis en place.

Question 3.23 Toujours pour cette optimisation, donner une nouvelle quantité permettant de prouver que l'on effectue au plus n itérations non-triviales (n étant comme auparavant le nombre d'états de l'automate).

Il reste toutefois à implémenter de façon efficace le test en ligne 2.2. Nous ne nous en chargeons pas ici ; cela peut-être fait en complexité amortie "presque constante". L'algorithme ainsi obtenu pour tester l'équivalence de deux états d'un automate est du à John Hopcroft et Richard Karp, sa complexité est "presque linéaire" : $O(kn\alpha(n))$ où k est le nombre de lettres et n le nombre d'états, et $\alpha(\cdot)$ est une fonction à croissance extrêmement faible (en fait, une inverse de la fonction d'Ackermann).

4 Relation globale d'équivalence

On s'intéresse maintenant au calcul, dans un automate $\langle X, o, \delta \rangle$ donné, de l'ensemble des paires d'états équivalents, c'est à dire de la relation \simeq dans sa globalité.

Question 4.1 En utilisant l'algorithme de John Hopcroft et Richard Karp, donner un algorithme simple pour calculer la relation \simeq , de complexité $O(kn^3\alpha(n))$. *+ test de toute la paire*

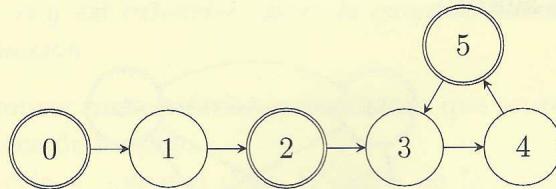
Cette approche n'est cependant pas optimale. Pour faire mieux, on va utiliser la caractérisation alternative du plus grand point fixe d'une fonction.

Question 4.2 En utilisant les questions 2.8 et 3.3, et sans supposer que l'automate est fini, montrer que l'on a $\simeq = \bigcap_{i \in \mathbb{N}} p^i(X \times X)$.

Question 4.3 Montrer que pour tout $i \in \mathbb{N}$, $p^i(X \times X)$ est une relation d'équivalence.

Question 4.4 Montrer que l'on a $\simeq = p^{n-1}(X \times X)$, où n est comme précédemment le nombre d'états de l'automate.

Question 4.5 Calculer les termes de la suite $(p^i(X \times X))_{i \in \mathbb{N}}$ pour l'automate ci-dessous. (On rappelle qu'une relation d'équivalence peut être vue comme une partition en classes d'équivalence, on pourra donc représenter ainsi de telles relations.) En déduire un automate à quatre états reconnaissant les langages reconnus par les six états de l'automate de départ.



Il s'agit maintenant de calculer cette suite efficacement. Soit P_0 la relation et d la fonction suivantes :

$$P_0 \stackrel{\text{def}}{=} \{(x, y) \mid o(x) = o(y)\} \quad d(R) \stackrel{\text{def}}{=} R \cap \{(x, y) \mid \forall a \in A, \delta(x, a) R \delta(y, a)\}$$

Question 4.6 Montrer que pour tout entier i , $d^i(P_0) = p^{i+1}(X \times X)$.

Question 4.7 Proposer un algorithme prenant en entrée un automate fini $\langle X, o, \delta \rangle$ et renvoyant la relation \simeq , de complexité $O(kn^2 \log n)$ dans le pire cas. On détaillera les structures de données utilisées.

Question 4.8 En déduire un algorithme prenant en entrée un automate et renvoyant un automate de taille minimale dont les états reconnaissent les différents langages reconnus par ceux de l'automate de départ.

On appelle un tel algorithme un algorithme de *minimisation*.

Question 4.9 Raffiner la routine calculant d pour obtenir un algorithme de minimisation de complexité $O(kn^2)$ dans le pire cas.

L'algorithme ainsi obtenu est dû à Edward Moore, il a ensuite été amélioré par John Hopcroft pour obtenir une complexité en $(O(kn \log n))$.

Fin du sujet.

A2022 – INFO MP

**ÉCOLE DES PONTS PARISTECH,
ISAE-SUPAERO, ENSTA PARIS,
TÉLÉCOM PARIS, MINES PARIS,
MINES SAINT-ÉTIENNE, MINES NANCY,
IMT ATLANTIQUE, ENSAE PARIS,
CHIMIE PARISTECH - PSL.**

Concours Mines-Télécom,
Concours Centrale-Supélec (Cycle International).

CONCOURS 2022**ÉPREUVE D'INFORMATIQUE MP**

Durée de l'épreuve : 3 heures

L'usage de la calculatrice et de tout dispositif électronique est interdit.

Cette épreuve concerne uniquement les candidats de la filière MP.

*Les candidats sont priés de mentionner de façon apparente
sur la première page de la copie :*

INFORMATIQUE - MP

L'énoncé de cette épreuve comporte 9 pages de texte.

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Préliminaires

Présentation du sujet

L'épreuve est composée d'un problème unique, comportant 27 questions. Dans ce problème, nous étudions un mécanisme pour lire un mot qui a été brouillé (par exemple `corekte`) et proposer un mot ciblé comme correction de ce mot (par exemple `correct`). Ce type de traitement est couramment utilisé dans le contexte de la vérification orthographique, de la reconnaissance vocale, de la lecture optique ou encore de la conception de moteurs de recherche tolérant aux requêtes mal formulées.

Après cette section de préliminaires, le problème est divisé en trois sections. Dans la première section (page 1), nous étudions comment mesurer des erreurs commises à la saisie d'un mot par une méthode de programmation dynamique. Dans la deuxième section (page 4), nous étudions comment stocker un corpus de mots par une méthode arborescente et comment fouiller ce corpus à l'aide d'un automate déjà construit. Dans la troisième section (page 7), nous étudions comment construire un filtre de fouille par une méthode inspirée de la théorie des automates.

Dans tout l'énoncé, un même identificateur écrit dans deux polices de caractère différentes désignera la même entité, mais du point de vue mathématique pour la police en italique (par exemple n ou n') et du point de vue informatique pour celle en romain avec espacement fixe (par exemple `n` ou `nprime`).

Travail attendu

Pour répondre à une question, il sera permis de réutiliser le résultat d'une question antérieure, même sans avoir réussi à établir ce résultat.

Il faudra coder des fonctions à l'aide du langage de programmation OCaml, en reprenant l'en-tête de fonction fourni par le sujet, sans nécessairement recopier la déclaration des types. Quand l'énoncé demande de coder une fonction, sauf demande explicite de l'énoncé, il n'est pas nécessaire de justifier que celle-ci est correcte ou que des préconditions sont satisfaites.

Le barème tient compte de la clarté des programmes : nous recommandons de choisir des noms de variables intelligibles ou encore de structurer de longs codes par des blocs ou par des fonctions auxiliaires dont on décrit le rôle.

1 Une mesure des erreurs de saisie

1.1 Une fonction mystère

La constante entière `max_int` désigne le plus grand entier représentable par OCaml. Nous nous donnons la fonction `mystere` suivante.

```
let rec mystere z = match z with
  (* La fonction mystere calcule ... *)
  | [] -> max_int
  | [a] -> a
  | a::b::y -> mystere ((if a <=b then a else b)::y);;
```

- 1 – Donner la signature de la fonction `mystere`. Justifier brièvement.

 Épreuve d'option informatique MP 2022

2 – Dire si, quelle que soit l'entrée z respectant le typage, le calcul de `mystere z` se termine et le démontrer. Préciser, le cas échéant, le nombre d'appels à la fonction `mystere`.

3 – Compléter sommairement le commentaire de la ligne 2. Énoncer une propriété qui caractérise exactement la valeur de retour de la fonction `mystere`. Démontrer cette propriété.

Dans la question suivante, le terme *complexité en espace* désigne un ordre de grandeur asymptotique de l'espace utilisé en mémoire lors de l'exécution d'un algorithme pour stocker tant l'entrée que des résultats intermédiaires et la valeur de retour.

4 – Quelle est la complexité en espace de l'appel `mystere z`? Est-elle optimale ?

1.2 Distance d'édition de Levenshtein

Nous fixons un alphabet de 27 *symboles* $\Sigma = \{a, b, \dots, y, z, \$\}$ qui se représentent par le type `char`. L'ensemble des *mots* sur l'alphabet Σ se note Σ^* ; la longueur d'un mot $w \in \Sigma^*$ se note $|w|$. Dans toute cette sous-section, nous fixons deux mots : le mot $b = b_1 \dots b_m$, dit *brouillé*, de longueur m et le mot $c = c_1 \dots c_n$, dit *ciblé*, de longueur n .

Définition : Nous appelons *distance* entre un mot brouillé $b \in \Sigma^*$ et un mot ciblé $c \in \Sigma^*$, et nous notons $\text{dist}(b, c)$, le nombre minimum de symboles qu'il faut supprimer, insérer ou substituer à un autre symbole pour transformer le mot brouillé b en le mot ciblé c . On remarque que $\text{dist}(b, c) = \text{dist}(c, b)$.

Par exemple, la distance entre le mot brouillé $b = \text{corekte}$ et le mot ciblé $c = \text{correct}$ vaut 3 : en effet, on peut insérer un `r`, substituer un `c` au `k` et supprimer le `e` final pour passer du mot b au mot c ; par ailleurs, on peut vérifier que cette transformation est impossible en effectuant deux opérations ou moins.

Nous notons $\text{préf}_i(b)$ le *préfixe de longueur i* du mot b , c'est-à-dire le mot des i premiers symboles de b . Pour $i = 0$, il s'agit du mot vide ε . Pour tous les indices i compris entre 0 et m et pour j compris entre 0 et n , nous notons $d_{i,j} = \text{dist}(\text{préf}_i(b), \text{préf}_j(c))$ la distance entre les préfixes $\text{préf}_i(b)$ et $\text{préf}_j(c)$.

5 – Pour tous les entiers i compris entre 0 et m et j compris entre 0 et n , déterminer les distances $d_{i,0}$ et $d_{0,j}$.

6 – Pour tous les entiers i compris entre 0 et $m - 1$ et j compris entre 0 et $n - 1$, exprimer la distance $d_{i+1,j+1}$ en fonction des distances $(d_{i',j'})_{\substack{0 \leq i' \leq i \\ 0 \leq j' \leq j}}$.

Épreuve d'option informatique MP 2022

Indication Ocaml : Un élément de type char se déclare entre deux apostrophes : par exemple, on code 'a' pour définir le symbole a. Les mots de Σ^* se représentent par le type char list. Nous déclarons

```
type mot = char list;;
```

Nous rappelons la syntaxe des fonctions suivantes :

- List.length, de type 'a list -> int : la longueur d'une liste ℓ est List.length ℓ ;
- Array.make, de type int -> 'a -> 'a array : un tableau de longueur n dont chaque case est initialisée avec la valeur v s'obtient par Array.make n v . Dans un tableau, les indices sont numérotés à partir de 0. On accède au coefficient en position i du tableau a par l'expression $a.(i)$, on le modifie par l'instruction $a.(i) \leftarrow v$.

- 7 – Écrire une fonction array_of_mot (w:mot) : char array dont la valeur de retour est un tableau formé des symboles du mot w écrits dans le même ordre.

Indication Ocaml : Nous rappelons la syntaxe de la fonction suivante :

- Array.make_matrix, de type int -> int -> 'a -> 'a array array : une matrice de taille $s \times t$ dont toutes les cases sont initialisées avec la valeur v s'obtient par Array.make_matrix s t v . Dans une matrice, les indices sont numérotés à partir de 0. On accède au coefficient en position (i, j) de la matrice a par l'expression $a.(i).(j)$, on le modifie par l'instruction $a.(i).(j) \leftarrow v$.

- 8 – Écrire une fonction distance (b:mot) (c:mot) : int qui calcule par mémoisation la distance $\text{dist}(b, c)$.

Indication Ocaml : Nous rappelons la syntaxe de la fonction suivante :

- List.filter, de type ('a -> bool) -> 'a list -> 'a list : la sous-liste des éléments x de la liste ℓ tels que le prédicat $p(x)$ vaut true s'obtient par filter p ℓ .

- 9 – Soit n_{\max} un entier. Exprimer la complexité en temps de la fonction distance b c en fonction des longueurs m et n des mots b et c . En déduire la complexité de l'instruction List.filter (fun c -> (distance b c) $\leq k$) ℓ_c ; ; où ℓ_c est une liste de mots ciblés, chacun de longueur inférieure ou égale à n_{\max} , et où k est un entier naturel.

- 10 – Soit k la distance $\text{dist}(b, c)$. Montrer qu'il existe un entier r , des mots b_0, b_1, \dots, b_r appartenant chacun à Σ^* , des mots c_0, c_1, \dots, c_r appartenant chacun à Σ^* et des symboles x_0, x_1, \dots, x_{r-1} appartenant chacun à Σ tels que

$$b = b_0 x_0 b_1 \dots x_{r-1} b_r \quad \text{et} \quad c = c_0 x_0 c_1 \dots x_{r-1} c_r$$

et

$$k = \sum_{i=0}^r \max(|b_i|, |c_i|).$$

2 Fouille dans un trie

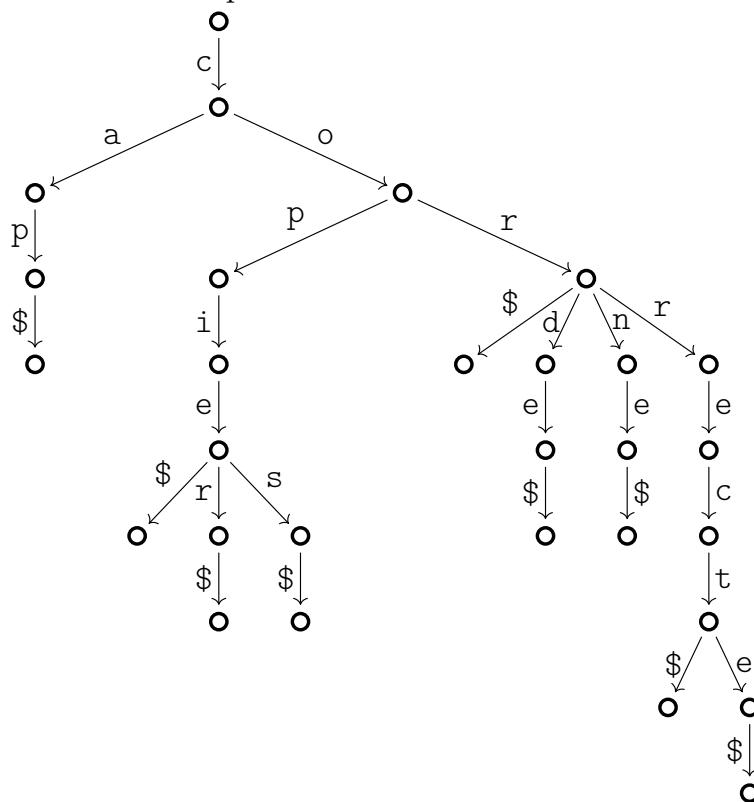
2.1 Représentation d'un corpus de mots par un trie

Définition : Un *trie* est un type particulier d'arbre dont chaque arête est orientée de la racine vers les feuilles et est étiquetée par un symbole de Σ . Le trie vide est formé d'un seul sommet et de zéro arête. Lorsqu'une arête est étiquetée par le symbole $\$$, son extrémité finale doit être le trie vide. La taille d'un trie t , notée $|t|$, est le nombre de ses arêtes.

On dit qu'un mot $c = c_1 \dots c_n \in (\Sigma \setminus \{\$\})^n$ appartient à un trie s'il existe un chemin, constitué de $n + 1$ arêtes, issu de la racine et dont les arêtes successives ont pour étiquettes respectives les symboles c_1, c_2, \dots, c_n et $\$$. Le symbole $\$$ indique la présence d'un mot et joue le rôle de symbole terminateur.

Dans cette partie, les tries sont utilisés afin de représenter un corpus de mots ciblés.

FIGURE 1 – Exemple de trie de taille 28 contenant 9 mots.



- 11 – Donner l'ensemble des mots du trie représenté à la figure 1.

Dans ce sujet, le terme *dictionnaire* désigne en général un type de données abstrait qui contient des associations entre une clé et une valeur.

- 12 – Nommer deux structures de données concrètes, qui réalisent le type abstrait *dictionnaire*. Pour chacune d'entre elles, rappeler sans justifier la complexité en temps de l'opération *insertion*.

Épreuve d'option informatique MP 2022

Indication Ocaml : Nous définissons un type polymorphe `'a char_map` qui réalise une structure de données persistante de dictionnaire associant des clés de type `char` à des valeurs de type quelconque `'a`. Nous disposons de la constante et de la fonction suivante :

- `CharMap.empty`, de type `'a char_map`, qui désigne le dictionnaire vide.
- `CharMap.add`, de type `char -> 'a -> 'a char_map -> 'a char_map`, telle que la valeur de retour de `CharMap.add x t d` est un dictionnaire contenant les associations du dictionnaire `d` ainsi qu'une association supplémentaire entre la clé `x` et la valeur `t`. Si la clé `x` était déjà associée dans le dictionnaire `d`, l'ancienne association de `x` disparaît.

Pour représenter les tries, nous définissons le type

```
type trie = Node of trie char_map ;;
```

13 – Définir deux constantes `trie_vide` et `trie_motvide`, de type `trie`, qui réalisent respectivement le trie vide et le trie contenant le mot vide ε .

14 – Écrire une fonction `trie_singleton (x:char)` : `trie` qui construit un trie contenant le mot formé d'un seul symbole $x \in \Sigma \setminus \{\$\}$.

Indication Ocaml : Nous utilisons un type polymorphe `'a option` défini par

```
type 'a option =
| None
| Some of 'a;;
```

qui permet de représenter des valeurs de type `'a` parfois non définies. Nous complétons le type `'a char_map` par la fonction suivante :

- `CharMap.find`, de type `char -> 'a char_map -> 'a option`, telle que l'appel de `CharMap.find x d` renvoie, en temps constant, `Some t` si la clé `x` est associée à la valeur `t` dans le dictionnaire `d` et renvoie `None` s'il n'existe pas d'association de clé `x` dans le dictionnaire `d`.

15 – Écrire une fonction `trie_mem (c:mot) (Node tcm:trie)` : `bool` qui teste si le mot $c \in (\Sigma \setminus \{\$\})^*$ appartient au trie $t = \text{Node } tcm$.

16 – Écrire une fonction `trie_add (c:mot) (Node tcm:trie)` : `trie` dont la valeur de retour est un trie contenant les mêmes mots que le trie $t = \text{Node } tcm$ ainsi que le mot $c \in (\Sigma \setminus \{\$\})^*$.

17 – Nous construisons le trie représenté à la figure 1 en déclarant d'abord la constante `trie_vide` (question 13), puis en appliquant neuf fois la fonction `trie_add` (question 16). Compte tenu du caractère persistant du type `'a char_map`, combien d'exemplaires de `trie_vide` coexiste-t-il une fois la construction terminée ? Expliquer.

Épreuve d'option informatique MP 2022

Définition : Un trie est dit *élagué* si toute feuille est précédée d'une arête d'étiquette \$.

Indication Ocaml : Nous complétons le type 'a char_map par les fonctions suivantes :

- CharMap.is_empty, de type 'a char_map -> bool, qui teste si un dictionnaire est le dictionnaire vide.
- CharMap.filter_map, de type (char -> 'a -> 'a option) -> 'a char_map -> 'a char_map, telle que CharMap.filter_map f d renvoie le dictionnaire d' restreint aux associations entre la clé x et la valeur t' où, d'une part, il existe dans le dictionnaire d une association entre la clé x et la valeur t et, d'autre part, $f(t)$ vaut Some tprime. Si le dictionnaire d contient un couple clé et valeur (x, t) mais que $f(t)$ vaut None, alors le dictionnaire d' ne contient pas d'association de clé x . En voici une illustration :

$\begin{array}{lll} d & x_1 & \mapsto & t_1 \\ & x_2 & \mapsto & t_2 \\ & x_3 & \mapsto & t_3 \\ & \vdots & & \vdots \end{array}$	$\begin{array}{llll} d' & x_1 & \mapsto & t'_1 & f(t_1) = \text{Some } t1\text{prime} \\ & & & & f(t_2) = \text{None} \\ & x_3 & \mapsto & t'_3 & f(t_3) = \text{Some } t3\text{prime} \\ & \vdots & & \vdots & \end{array}$
---	--

- 18 – Compléter le code suivant afin que la valeur de retour de trie_trim t soit un trie élagué contenant les mêmes mots que le trie $t = \text{Node tcm}$.

```
let rec trie_trim (Node tcm:trie) : trie =
  let filtre (x:char) (y:trie) : trie option =
    (* à compléter *)
    in
    Node(CharMap.filter_map filtre tcm);;
```

2.2 Filtrage dans un trie

- 19 – Soient $\mathbf{b} \in (\Sigma \setminus \{\$\})^*$ un mot brouillé, t un trie contenant un ensemble de mots ciblés et k un entier naturel. On suppose avoir engendré la liste $\ell_{\mathbf{b}}$ des mots de $(\Sigma \setminus \{\$\})^*$ à distance inférieure ou égale à k du mot \mathbf{b} . Montrer que la liste $\ell_{\mathbf{b}}$ compte $O(|\mathbf{b}|^k)$ éléments. En déduire la complexité en temps de l'instruction List.filter (fun bb -> trie_mem bb t) lbb;;.

Définition : Nous appelons *système de transitions* sur l'alphabet Σ la donnée d'un triplet (Q, \hat{q}, Δ) où

- Q est un ensemble (fini ou non), dit *ensemble des états*,
- $\hat{q} \in Q$ est un état, dit *état initial*,
- $\Delta \subseteq Q \times \Sigma \times Q$ est une relation, dite *relation de transition*.

Un mot $\mathbf{c} = c_1 \dots c_n \in \Sigma^n$ est *accepté* par le système de transitions (Q, \hat{q}, Δ) s'il existe une suite d'états $(q_j)_{0 \leq j \leq n}$ telle que l'état q_0 égale l'état initial \hat{q} et, pour tout j compris entre 0 et $n - 1$, le triplet (q_j, c_{j+1}, q_{j+1}) appartient à la relation de transition Δ .

 Épreuve d'option informatique MP 2022

On peut voir un système de transitions comme un automate éventuellement infini dont tous les états sont finals.

- 20 – Dessiner, sans justifier, un système de transitions fini qui accepte les mots $w \in \Sigma^*$ n'ayant pas le mot ccmp comme facteur (c'est-à-dire que le mot w n'est pas accepté si et seulement s'il contient quatre symboles consécutifs valant c, c, m et p).

Nous disons qu'un système de transitions (Q, \hat{q}, Δ) est *déterministe* s'il existe une fonction partiellement définie $\delta : Q \times \Sigma \rightarrow Q$ telle que l'on ait

$$\Delta = \{(q, x, \delta(q, x)) ; (q, x) \in Q \times \Sigma \text{ et } \delta(q, x) \text{ est défini}\}.$$

Nous notons alors (Q, \hat{q}, δ) ce système.

Indication Ocaml : Afin de représenter des systèmes de transitions déterministes, nous convenons de nous appuyer sur un type `etat` pour représenter l'ensemble des états Q . Ce type sera explicité ultérieurement. Nous déclarons

```
type syst_trans = etat -> char -> etat option;;
```

pour représenter la fonction de transition δ . Pour tout couple $(q, x) \in Q \times \Sigma$, si `delta` est de type `syst_trans`, on accède à l'état image $q' = \delta(q, x)$ par l'expression `delta q x` qui vaut alors `Some qprime` si la transition est bien définie ou bien `None` si la transition n'est pas définie.

- 21 – Écrire une fonction `trie_filter` (`qchapeau:etat`) (`delta:syst_trans`) (`Node tcm:trie`) : `trie` qui renvoie un `trie` contenant les mots du `trie` $t = \text{Node } tcm$ acceptés par le système de transitions déterministe (Q, \hat{q}, δ) . Il n'est pas demandé de renvoyer un `trie` élagué.

- 22 – Un système de transitions déterministe (Q, \hat{q}, δ) étant fixé, quelle est la complexité en temps du calcul `trie_filter` `qchapeau delta t` en fonction du `trie` t ? On suppose que l'exécution de la fonction de transition δ s'effectue en temps constant.

3 Système de transitions des voisins d'un mot brouillé

Dans toutes les questions restantes, les lettres b et c désignent systématiquement deux mots de longueur m et n de $(\Sigma \setminus \{\$\})^*$. La lettre k désigne un entier naturel. L'écriture $c\$$ désigne la concaténation du mot c et du symbole $\$$; nous parlons alors de *mot prolongé*.

L'objectif de cette section est de construire un système de transitions non déterministe qui, à partir d'un entier naturel k et d'un mot brouillé b , accepte n'importe quel préfixe du mot prolongé $c' = c\$$ où c est un mot de $(\Sigma \setminus \{\$\})^*$ tel que $\text{dist}(b, c) \leq k$ et aucun autre mot n'est accepté.

 Épreuve d'option informatique MP 2022

Définition : Nous appelons *transformations élémentaires* les $(k+3)$ appellations **suppr**, **subs** et **h -ins-puis-id**, avec $0 \leq h \leq k$; nous notons \mathcal{T} leur ensemble.

Soient $\mathbf{c}' = c'_1 \dots c'_n \in \Sigma^*$ un mot de longueur n et $\mathbf{b}' \in \Sigma^*$ un mot de longueur m . Nous disons qu'une suite $\boldsymbol{\tau} = (\tau_1, \tau_2, \dots, \tau_n)$ de n transformations élémentaires est un *script de transformation* du mot \mathbf{c}' en le mot \mathbf{b}' s'il existe une factorisation $\beta_1 \beta_2 \dots \beta_n$ du mot \mathbf{b}' telle que pour tout entier j compris entre 1 et n ,

- (1) β_j est un mot de Σ^* ,
- (2) lorsque $\tau_j = \text{suppr}$, alors le mot β_j est le mot vide ϵ ,
- (3) lorsque $\tau_j = \text{subs}$, alors le mot β_j est de longueur 1 et, en l'identifiant à un symbole, il est distinct du symbole c'_j ,
- (4) lorsque $\tau_j = h\text{-ins-puis-id}$, alors le mot β_j est de longueur $h+1$ et le dernier symbole de β_j vaut le symbole c'_j .

Nous observons que la factorisation du mot \mathbf{b}' en $\beta_1 \beta_2 \dots \beta_n$ est unique et ne dépend que du script $\boldsymbol{\tau}$.

Voici un exemple de script de transformation entre le mot $\mathbf{c}' = \text{correct\$}$ et le mot $\mathbf{b}' = \text{incorekte\$}$.

$(c'_j)_{1 \leq j \leq 8}$	c	o	r	r	e	c	t	\$
$(\beta_j)_{1 \leq j \leq 8}$	inc	o	r	ϵ	e	k	t	$\epsilon \$$
$(\tau_j)_{1 \leq j \leq 8}$	2-ins-puis-id	0-ins-puis-id	0-ins-puis-id	suppr	0-ins-puis-id	subs	0-ins-puis-id	1-ins-puis-id

Définition : Le *coût* d'une transformation élémentaire est précisé par le tableau suivant :

Transformation élémentaire	suppr	subs	h -ins-puis-id
Coût	1	1	h

Le coût d'un script de transformation est la somme des coûts des transformations élémentaires constituant le script.

Dans l'exemple ci-dessus, le coût du script vaut 5 (ou encore $2 + 0 + 0 + 1 + 0 + 1 + 0 + 1$).

Définition : Nous utilisons le terme *k-script* pour raccourcir l'expression « script de transformation de coût inférieur ou égal à k ».

23 – Montrer que la distance **dist** (\mathbf{b}, \mathbf{c}) est inférieure ou égale à k si et seulement s'il existe un *k*-script du mot prolongé $\mathbf{c}\$$ vers le mot prolongé $\mathbf{b}\$$.

Dans les questions 24 et 25, la lettre j désigne un entier avec $0 \leq j \leq n$ et $\boldsymbol{\tau} = (\tau_1, \tau_2, \dots, \tau_j) \in \mathcal{T}^j$ un *k*-script depuis le préfixe $\text{préf}_j(\mathbf{c}\$)$ du mot prolongé $\mathbf{c}\$$ vers un certain préfixe \mathbf{p} du mot prolongé $\mathbf{b}\$$. On appelle \mathbf{s} le suffixe de $\mathbf{b}\$$ tel que $\mathbf{b}\$$ se factorise en $\mathbf{b}\$ = \mathbf{p}\mathbf{s}$.

24 – Si l'on a $0 \leq j < n$, par quelles appellations $\tau_{j+1} \in \mathcal{T}$ est-il possible de compléter le *k*-script $\boldsymbol{\tau}$ pour que $(\tau_1, \tau_2, \dots, \tau_{j+1})$ soit un *k*-script depuis le préfixe $\text{préf}_{j+1}(\mathbf{c}\$)$ vers un certain préfixe du mot prolongé $\mathbf{b}\$$? On exprimera sa réponse en fonction du $(j+1)^{\text{e}}$ symbole c_{j+1} de $\mathbf{c}\$$, du suffixe \mathbf{s} et du coût κ du script $\boldsymbol{\tau}$.

25 – Si l'on a $j = n$, par quelles appellations $\tau_{n+1} \in \mathcal{T}$ et sous quelles conditions est-il possible de compléter le *k*-script $\boldsymbol{\tau}$ pour que $(\tau_1, \tau_2, \dots, \tau_{n+1})$ soit un *k*-script depuis le mot prolongé $\mathbf{c}\$$ vers le mot prolongé $\mathbf{b}\$$?

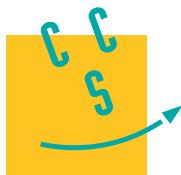
Épreuve d'option informatique MP 2022

Indication Ocaml : Nous précisons le type etat en déclarant

```
type etat = mot * int;;
```

- 26 – Le mot brouillé $\mathbf{b} \in (\Sigma \setminus \{\$\})^*$ étant toujours fixé, décrire en OCaml un système de transitions (Q, \hat{q}, Δ) qui accepte tout préfixe du mot $\mathbf{c}\$$, avec $\mathbf{c} \in (\Sigma \setminus \{\$\})^*$, tel que $\text{dist}(\mathbf{b}, \mathbf{c}) \leq k$ et qui n'accepte aucun autre mot. On définira une fonction `etat_initial (b:mot) (k:int) : etat` qui construit l'état initial \hat{q} et une fonction `delta (k:int) (q:etat) (x:char) : etat list` qui renvoie la liste des états q' tels que l'on ait $(q, x, q') \in \Delta$.
- 27 – Comment adapter la fonction `trie_filter`, écrite à la question 21, pour qu'elle fonctionne avec le système de transitions de la question 26 ? On ne demande pas de code. Préciser la complexité en temps. Pourquoi peut-on souhaiter adapter la fonction `trie_filter` plutôt que de déterminiser le système de transitions ?

FIN DE L'ÉPREUVE



Option informatique

MP

2022

CONCOURS CENTRALE-SUPÉLEC

4 heures

Calculatrice autorisée

Ce sujet aborde différents problèmes autour des automates et des expressions rationnelles. On explore dans une première partie des propriétés sur le miroir d'un langage, sur les palindromes et sur les automates correspondants. On implémente ensuite l'algorithme de déterminisation d'un automate, afin de construire de manière effective l'automate de Brzozowski qui a la propriété d'être minimal. Dans une deuxième partie, on travaille sur la syntaxe des expressions rationnelles, puis sur une construction de l'expression rationnelle associée à un automate donné, par un algorithme divisor pour régner, dû à Conway, en exploitant une représentation matricielle d'un automate et la construction de l'étoile d'une matrice d'expressions rationnelles. Enfin, dans une troisième partie, on introduit les dérivées d'Antimirov, qui permettent d'obtenir un automate fini non déterministe avec peu d'états qui reconnaît le langage spécifié par une expression rationnelle. Les trois parties sont indépendantes, de difficulté progressive.

Langages et mots

On appelle *alphabet* tout ensemble fini de lettres. On note généralement l'alphabet Σ .

On note Σ^* l'ensemble de tous les mots formés sur l'alphabet Σ .

La *longueur* (ou la *taille*) d'un mot $w \in \Sigma^*$ est son nombre de lettres et se note $|w|$. Le *mot vide*, noté ε , est le seul mot de longueur nulle.

Si un mot $w \in \Sigma^*$ est de longueur $|w| = n$, on le note $w = a_0a_1\dots a_{n-1}$, où les a_i sont des lettres de Σ .

Un *langage* sur l'alphabet Σ est un ensemble $L \subset \Sigma^*$.

L'*étoile de Kleene* d'un langage L , notée L^* , est le plus petit langage qui inclut L , qui contient ε et qui est stable par concaténation.

La concaténation de deux langages L et L' est notée $L \cdot L'$, souvent abrégé en LL' lorsqu'il n'y a pas d'ambiguïté.

Automates finis

Un *automate fini non déterministe* sur un alphabet Σ est un quadruplet $A = (Q, I, F, T)$, où Q est un ensemble fini d'états, $I \subset Q$ est le sous-ensemble des états initiaux, $F \subset Q$ est le sous-ensemble des états finaux et l'ensemble $T \subset Q \times \Sigma \times Q$ est l'ensemble des transitions, étiquetées par les lettres de l'alphabet Σ .

Si $(q, a, q') \in T$, on note $q \xrightarrow{a} q'$ cette transition.

Pour représenter graphiquement un automate, on utilise une flèche entrante pour désigner un état initial et une flèche sortante pour désigner un état final, comme l'illustre l'exemple de la figure 1.

Un mot $w = a_0\dots a_{n-1}$ est reconnu par l'automate A s'il existe une succession de transitions :

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots q_{n-1} \xrightarrow{a_{n-1}} q_n \quad \text{avec} \quad q_0 \in I \quad \text{et} \quad q_n \in F.$$

On dira que le mot w *étiquette un chemin* dans l'automate A allant de q_0 à q_n .

Le langage d'un automate A , noté L_A , est exactement l'ensemble des mots reconnus par l'automate A . On dit alors que A *reconnait* L_A . Un langage est dit *reconnaissable* s'il est le langage d'un automate fini.

Un *automate fini déterministe* sur un alphabet Σ est un quadruplet $A = (Q, \{q_0\}, F, \delta)$, où l'ensemble des états initiaux est un singleton (un unique état initial) et où l'ensemble des transitions T est remplacé par une fonction de transition δ définie sur un sous-ensemble de $Q \times \Sigma$ et à valeurs dans Q . Pour chaque couple $(q, a) \in Q \times \Sigma$, il existe au plus une transition (q, a, q') qui, si elle existe, est telle que $q' = \delta(q, a)$.

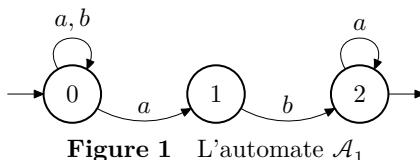
L'automate est déterministe *complet* si la fonction de transition δ est définie sur $Q \times \Sigma$. Dans ce cas, on définit la *fonction de transition étendue* δ^* sur $Q \times \Sigma^*$ par

$$\forall q \in Q, \quad \begin{cases} \delta^*(q, \varepsilon) = q \\ \delta^*(q, wa) = \delta(\delta^*(q, w), a) \end{cases} \quad \forall w \in \Sigma^*, \forall a \in \Sigma$$

Les automates seront représentés par le type Caml suivant

```
type automate = { nb : int; (* nombre d'états *)
                  init : int list; (* états initiaux *)
                  final : int list; (* états finaux *)
                  trans : (int * char * int) list };; (* transitions *)
```

l'ensemble d'états Q d'un automate implémenté étant toujours supposé être un intervalle d'entiers $\llbracket 0, n - 1 \rrbracket$.

**Figure 1** L'automate \mathcal{A}_1

Par exemple, l'automate \mathcal{A}_1 de la figure 1 est codé par

```

let a1 = { nb = 3 ;
            init = [0];
            final = [2];
            trans = [(0, 'a', 0); (0, 'a', 1); (0, 'b', 0); (1, 'b', 2); (2, 'a', 2)] } ;;
  
```

On accède au nombre d'états par `a1.nb`, à la liste des états initiaux par `a1.init`, à la liste des états finaux par `a1.final` et à la liste des transitions par `a1.trans`.

Expressions rationnelles

Soit Σ un alphabet. On définit la *syntaxe des expressions rationnelles* par :

- \emptyset, ε et a sont des expressions rationnelles, pour toute lettre $a \in \Sigma$;

- si E et F sont deux expressions rationnelles, alors $(E + F)$, $(E \cdot F)$ et E^* sont des expressions rationnelles.

La *sémantique des expressions rationnelles* est définie par l'application \mathcal{L} qui associe à toute expression rationnelle un langage rationnel sur Σ par :

$$\begin{cases} \mathcal{L}(\emptyset) = \emptyset & (\text{langage vide}) \\ \mathcal{L}(\varepsilon) = \{\varepsilon\} & (\text{langage contenant le mot vide}) \\ \mathcal{L}(a) = \{a\} & \forall a \in \Sigma \end{cases}$$

et, si E et F sont deux expressions rationnelles,

$$\begin{cases} \mathcal{L}(E + F) = \mathcal{L}(E) \cup \mathcal{L}(F) \\ \mathcal{L}(E \cdot F) = \mathcal{L}(E) \cdot \mathcal{L}(F) \\ \mathcal{L}(E^*) = \mathcal{L}(E)^* \end{cases}$$

où $*$ représente l'étoile de Kleene d'un langage et \cdot représente la concaténation de deux langages.

Programmation

Le seul langage de programmation autorisé dans cette épreuve est Caml. Toutes les fonctions des modules `Array` et `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max` ou `incr` ainsi que les opérateurs comme `/` ou `mod`) peuvent être librement utilisés.

Généralement, les objets mathématiques dans le texte seront notés A, m, i, n, ℓ , alors qu'ils seront représentés en Caml par `a, m, i, n, 1`.

Les complexités demandées sont des complexités temporelles dans le pire des cas et seront exprimées sous la forme $O(f(n, m))$, où f est une fonction usuelle simple et où n et m sont des paramètres correspondant aux tailles des objets en entrée de l'algorithme.

I Mots et automates

I.A – Miroir d'un mot et automate transposé

Pour tout mot $w = a_0a_1\dots a_{n-1}$ de longueur $n \in \mathbb{N}^*$, on définit son *mot miroir* \tilde{w} par $\tilde{w} = a_{n-1}\dots a_1a_0$. Par convention, le mot vide ε est son propre miroir.

Pour tout langage $L \subset \Sigma^*$, on définit son *langage miroir* \tilde{L} constitué de l'ensemble des mots miroirs du langage L :

$$\tilde{L} = \{\tilde{w} \mid w \in L\}.$$

Q 1. Décrire le langage L_1 de l'automate \mathcal{A}_1 de l'exemple de la figure 1 et décrire son langage miroir \tilde{L}_1 .

Q 2. Dessiner un automate $\tilde{\mathcal{A}}_1$, reconnaissant le langage miroir \tilde{L}_1 .

Soit $A = (Q, I, F, T)$ un automate non déterministe et $L = L_A$ le langage qu'il reconnaît.

Q 3. Donner, en justifiant, la construction de l'automate miroir $\tilde{A} = (Q, I', F', T')$ qui reconnaît le langage \tilde{L} .

Q 4. Écrire une fonction `transpose` de signature `automate -> automate` qui étant donné un automate A non déterministe en entrée, renvoie un automate non déterministe qui reconnaît le miroir de L_A .

Q 5. Quelle est la complexité de cette fonction ?

I.B – Palindromes et rationalité

Soit $w \in \Sigma^*$. On dit que le mot w est un *palindrome* si $\tilde{w} = w$.

Q 6. Écrire une fonction `palindrome` de signature `string -> bool` qui teste, en temps linéaire, si un mot est un palindrome.

On rappelle que pour tout $0 \leq i < (\text{String.length } s)$, le i -ième caractère de la chaîne de caractères s est obtenu par l'expression `s.[i]`.

Pour un alphabet Σ , on note $\text{Pal}(\Sigma)$ l'ensemble des palindromes de Σ^* .

Q 7. Montrer que si Σ est un alphabet à une lettre, alors $\text{Pal}(\Sigma)$ est rationnel.

Q 8. Montrer que si Σ contient au moins deux lettres, alors $\text{Pal}(\Sigma)$ n'est pas rationnel.

On pourra utiliser un automate et un mot de $\text{Pal}(\Sigma) \cap a^*ba^*$.

Soit $L \subset \Sigma^*$ un langage reconnu par l'automate $A = (Q, I, F, T)$.

Pour $(q, q') \in Q^2$, on note $L_{q,q'}$ le langage de tous les mots w qui étiquettent un chemin dans A partant de q et arrivant en q' .

Q 9. Montrer que $L_{q,q'}$ est reconnaissable et exprimer le langage L_A en fonction de langages $L_{q,q'}$.

Q 10. Montrer que $\text{Pal}(\Sigma) \cap (\Sigma^2)^* = \{u\tilde{u} \mid u \in \Sigma^*\}$.

Soit L un langage rationnel reconnu par un automate $A = (Q, I, F, T)$.

On définit les langages $D(L) = \{w\tilde{w} \mid w \in L\}$ et $R(L) = \{w \in \Sigma^* \mid w\tilde{w} \in L\}$.

Q 11. Décrire simplement les langages $D(a^*b)$ et $R(a^*b^*a^*)$.

Q 12. Les langages $D(L)$ et $R(L)$ sont-ils reconnaissables ?

On pourra faire intervenir les langages $L_{q,q'}$, définis ci-dessus.

I.C – Déterminisation

On rappelle que pour tout automate $A = (Q, I, F, T)$ non déterministe, on peut définir l'automate déterminisé accessible $A_{\text{det}} = (Y, \{I\}, F', \delta)$ où $Y \subset \mathcal{P}(Q)$ est l'ensemble des états accessibles depuis l'état initial $\{I\}$ dans l'automate des parties. Cet automate déterminisé accessible reconnaît le même langage que l'automate A .

Q 13. Écrire un automate \mathcal{A}_2 non déterministe à 4 états qui reconnaît le langage $L_2 = (b + ab)^*ba$. Cet automate devra avoir un unique état initial et un unique état final.

Q 14. Appliquer l'algorithme de déterminisation sur l'automate miroir $\widetilde{\mathcal{A}}_2$ afin d'obtenir l'automate $\mathcal{A}_3 = (\widetilde{\mathcal{A}}_2)_{\text{det}}$. Les états de \mathcal{A}_3 seront renommés e_0, e_1, \dots

Q 15. Appliquer l'algorithme de déterminisation sur l'automate miroir $\widetilde{\mathcal{A}}_3$ afin d'obtenir l'automate $\mathcal{A}_4 = (\widetilde{\mathcal{A}}_3)_{\text{det}}$. Ses états seront renommés q_0, q_1, \dots

Q 16. Quel doit être le langage reconnu par l'automate \mathcal{A}_4 ?

On cherche à généraliser cette construction de façon effective. Pour cela, on va implémenter l'algorithme de déterminisation.

Il faut d'abord choisir une représentation pour les parties de Q (c'est-à-dire des ensembles d'états). Une solution naïve consisterait à utiliser des listes d'états. Lors du déroulement de l'algorithme de déterminisation, on peut être amené à effectuer des réunions d'ensembles. Une concaténation simple des listes génère des doublons qu'il faut ensuite supprimer afin que les listes codent bien des ensembles d'états.

Q 17. Écrire une fonction `supprimer` de signature `'a list -> 'a list` qui prend une liste en entrée et supprime toutes les occurrences multiples de ses éléments.

Q 18. Donner la complexité de votre algorithme en fonction de la taille de la liste d'entrée.

On choisit plutôt de coder les ensembles d'états par des entiers.

Pour un automate $A = (Q, I, F, T)$ tel que $Q = \llbracket 0, n-1 \rrbracket$, toute partie de Q va être représentée par un entier entre 0 et $2^n - 1$. Dans la suite, on supposera $n \leq 20$. Soit X une partie de $\llbracket 0, n-1 \rrbracket$. On définit le numéro de X par la fonction suivante

$$\text{numero}(X) = \sum_{i \in X} 2^i.$$

On se donne `pow` un tableau des puissances de 2, qui contient toutes les puissances 2^k , pour $0 \leq k \leq 20$.

```
let pow = Array.make 21 1 ;
for i = 1 to 20 do
    pow.(i) <- pow.(i-1) * 2
done ;;
```

Soit $q \in \llbracket 0, n-1 \rrbracket$ un état et $k \in \llbracket 0, 2^n - 1 \rrbracket$ le numéro d'un ensemble d'états X , c'est-à-dire $\text{numero}(X) = k$.

Q 19. Écrire une fonction `est_dans` de signature `int -> int -> bool` qui teste, à l'aide d'opérations arithmétiques, si l'état q est dans l'ensemble d'états représenté par le numéro k en $O(1)$ opérations.

Soit ℓ une liste d'états contenant éventuellement plusieurs fois le même état, représentant l'ensemble X .

Q 20. Écrire une fonction `numero` de signature `int list -> int` qui calcule le numéro de l'ensemble X .

Par exemple $\ell = [1; 5; 2; 5; 2; 5; 2; 1; 2; 1]$ représente l'ensemble $X = \{1, 2, 5\}$, de numéro $38 = 2^1 + 2^2 + 2^5$.

Soit ℓ une liste d'états et X un ensemble d'états représenté par son numéro k .

Q 21. Écrire une fonction `intersecte` de signature `int list -> int -> bool` qui vérifie si un élément de ℓ est contenu dans l'ensemble X représenté par k .

On prépare désormais la fonction de transition de l'automate déterminisé accessible.

Soit X un ensemble d'états de Q . On suppose désormais que l'automate est sur l'alphabet à deux lettres $\Sigma = \{a, b\}$.

On cherche à calculer la fonction de transition $\delta : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$ de l'automate déterminisé. On rappelle que, pour $c \in \{a, b\}$ et $X \in \mathcal{P}(Q)$,

$$\delta(X, c) = \bigcup_{q \in X} \{q' \in Q \mid (q, c, q') \in T\}.$$

La transition $(X, c, \delta(X, c))$ sera alors dans l'automate déterminisé.

En parcourant l'ensemble des transitions T de l'automate, on va simultanément calculer les états $(\delta(X, a), \delta(X, b))$, ce qui correspond à la table de transition depuis l'état X .

Q 22. Écrire une fonction `etat_suivant` de signature `int -> (int*char*int) list -> (int*int)` qui, étant donné en entrée un entier k tel que $k = \text{numero}(X)$ et la liste des transitions T , calcule le couple d'entiers (k_a, k_b) tels que $k_a = \text{numero}(\delta(X, a))$ et $k_b = \text{numero}(\delta(X, b))$.

Au moment de construire l'automate déterminisé accessible A_{det} , on va être amené à renommer (c'est-à-dire ici renuméroter) les états de A_{det} pour avoir au final un ensemble d'états Y de la forme $\llbracket 0, N - 1 \rrbracket$ où N sera le nombre de parties de Q accessibles dans l'automate des parties. Pour cela, on va simplement utiliser une liste contenant des couples (k, v) où k est le numéro d'un ensemble d'états X et v le numéro final par lequel k sera remplacé. Par exemple, si à un moment donné de l'algorithme, la liste contient $(6, 2)$, "l'ensemble d'états 6" (qui correspond dans $\mathcal{P}(Q)$ à $\{1, 2\}$) est renuméroté 2.

Q 23. Écrire une fonction `cherche` de signature `int -> (int*int) list -> int` qui renvoie le nouveau numéro d'un ensemble d'états représenté par son numéro k dans une liste comme ci-dessus (-1 si k n'est pas présent).

Q 24. Écrire une fonction `determinise` de signature `automate -> automate` qui calcule le déterminisé accessible de l'automate d'entrée. On expliquera brièvement la démarche utilisée.

Q 25. Quelle est la complexité de votre fonction `determinise` en fonction du nombre d'états n de A et du nombre d'états N de A_{det} ?

I.D – Algorithme de Brzozowski

L'algorithme de Brzozowski permet d'obtenir un automate déterministe ayant un nombre minimal d'états, reconnaissant le même langage que l'automate initial.

On se donne un automate $A = (Q, I, \{f\}, T)$ qui reconnaît le langage L et tel que l'automate miroir \tilde{A} est déterministe et accessible.

On note $A_{\text{det}} = (Y, \{I\}, F, \delta)$ le déterminisé accessible de A .

Si u est un mot et L un langage, on note $u^{-1}L = \{w \in \Sigma^* \mid uw \in L\}$.

Q 26. Soit $q \in Q$ un état et $u \in \Sigma^*$ un mot. Montrer que si $q \in \delta^*(\{I\}, u)$, alors il existe un mot $w \in \Sigma^*$ tel que $uw \in L$.

Q 27. Montrer la propriété $(*)$: si l'on prend deux mots u et v dans Σ^* tels que $u^{-1}L = v^{-1}L$, alors dans l'automate A_{det} déterminisé, $\delta^*(\{I\}, u) = \delta^*(\{I\}, v)$.

Q 28. En déduire que si A est un automate quelconque reconnaissant L , alors en posant $B = (\tilde{A})_{\text{det}}$, montrer que $(\tilde{B})_{\text{det}}$ reconnaît L et vérifie la propriété $(*)$.

Q 29. Écrire une fonction `minimal` de signature `automate -> automate` appliquant la construction de Brzozowski sur l'automate d'entrée. On fera abstraction de la taille des automates générés, possiblement problématique.

II Expression rationnelle associée à un automate

Dans cette partie, on introduit un algorithme, dû à Conway, pour le calcul de l'expression rationnelle associée au langage d'un automate, via l'utilisation de matrices dont les coefficients sont des expressions rationnelles.

II.A – Simplification d'expressions rationnelles équivalentes

On se donne en Caml le type `exprat` des expressions rationnelles

```
type exprat = Vide
| Epsilon
| Lettre of char
| Union of exprat * exprat ;;
| Concat of exprat * exprat
| Etoile of exprat ;;
```

II.A.1)

Q 30. Écrire une fonction `lettre` de signature `exprat -> int` qui renvoie le nombre de lettres présentes dans l'expression rationnelle en argument. Par exemple, si $E = (a^*b) + ab(a + \varepsilon)^* + \emptyset$, (`lettre e`) doit renvoyer 7.

Q 31. Écrire une fonction `est_vide` de signature `exprat -> bool` qui teste si le langage rationnel représenté par l'expression rationnelle en argument est vide.

II.A.2) Dans cette section, on travaille formellement sur la syntaxe des expressions rationnelles.

On utilise les équivalences évidentes suivantes

$$\emptyset + E \equiv E + \emptyset \equiv E \quad E \cdot \varepsilon \equiv \varepsilon \cdot E \equiv E \quad E \cdot \emptyset \equiv \emptyset \cdot E \equiv \emptyset \quad \emptyset^* \equiv \varepsilon \quad \varepsilon^* \equiv \varepsilon \quad (E^*)^* \equiv E^*$$

où la notation $E \equiv E'$ signifie que les langages représentés sont égaux : $\mathcal{L}(E) = \mathcal{L}(E')$.

La fonction suivante réalise une simplification à la racine sur une expression du type `Union` en suivant la règle donnée.

```
let su expr = match expr with
| Union( Vide, e ) -> e
| Union( e, Vide ) -> e
| _ -> expr;;
```

De même, on peut écrire une fonction `sc` : `exprat -> exprat` qui simplifie à la racine une expression de type `Concat`. On suppose codées ces fonctions.

Q 32. Écrire une fonction `se` : `exprat -> exprat` qui simplifie à la racine une expression de type `Etoile` avec les règles données.

Prenons par exemple $E_n = (a + (b.(b.(b.(b....\emptyset)...)))$, où n lettres b concaténées se succèdent.

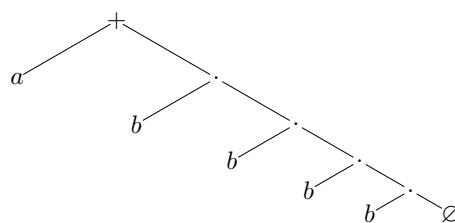


Figure 2 Exemple de l'arbre syntaxique de l'expression E_4

Q 33. Combien d'applications de règles décrites ci-dessus sont-elles nécessaires pour obtenir à partir de E_n l'expression équivalente a ?

Q 34. Écrire une fonction `simplifie` : `exprat -> exprat` qui simplifie une expression rationnelle selon les règles données.

II.B – Matrices d'expressions rationnelles

Dans la suite, on considère des matrices d'expressions rationnelles

```
type mat = exprat array array ;;
```

La matrice nulle de taille n est la matrice de taille n où chaque coefficient vaut \emptyset .

La matrice identité de taille n est la matrice de taille n où chaque coefficient vaut ε sur la diagonale et \emptyset en dehors de la diagonale.

On définit la somme de deux matrices A et B de taille $(n \times m)$ par la matrice $A + B$ de taille $(n \times m)$ où $[A + B]_{i,j} = A_{i,j} + B_{i,j}$ où le $+$ représente l'opération rationnelle d'union et $0 \leq i \leq n-1$, $0 \leq j \leq m-1$.

On définit le produit de deux matrices A et B de taille $(n \times p)$ et $(p \times q)$ à la manière du produit matriciel usuel AB , de taille $(n \times q)$, où la somme de coefficients est remplacée par l'union et où le produit de coefficients est remplacé par la concaténation des expressions rationnelles.

II.B.1)

Q 35. Écrire une fonction `somme` de signature `mat -> mat -> mat` effectuant la somme de deux matrices d'expressions rationnelles de même taille $(n \times p)$. Quelle est sa complexité ?

Q 36. Écrire une fonction `produit` de signature `mat -> mat -> mat` effectuant le produit de deux matrices d'expressions rationnelles, en supposant que les tailles sont bien compatibles (la première de taille $(n \times p)$ et la seconde de taille $(p \times q)$.) On ne vérifiera pas la compatibilité des tailles. Quelle est sa complexité ?

On cherche désormais à définir l'étoile de Kleene d'une matrice carrée d'expressions rationnelles.

II.B.2) Étude de l'étoile d'une matrice de taille 2

Plaçons-nous dans le cas d'une matrice carrée de taille 2, $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, où a, b, c et d sont quatre lettres.

On associe à cette matrice M le graphe étiqueté à deux sommets de la figure 3.

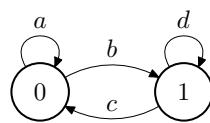


Figure 3

On note $L_{i,j}$ le langage de l'automate $\mathcal{A}_{i,j} = (\{0, 1\}, \{i\}, \{j\}, T)$, où $T = \{(i, M_{i,j}, j) \mid (i, j) \in \{0, 1\}^2\}$.

Q 37. Donner une expression rationnelle sur l'alphabet $\{a, b, c, d\}$ pour décrire chaque langage $L_{i,j}$.

II.B.3) Étoile d'une matrice carrée de taille quelconque

Pour la définition de l'étoile d'une matrice carrée d'expressions rationnelles, on procède récursivement, sur la taille n de la matrice carrée M :

- si la taille vaut 1, $M = (e)$, donc $M^* = (e^*)$;
- sinon, pour M de taille $n \geq 2$, on découpe M par blocs

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

où A et D sont carrées de taille ≥ 1 , et on définit

$$M^* = \begin{pmatrix} A' & B' \\ C' & D' \end{pmatrix}$$

où

$$A' = (A + BD^*C)^* \quad B' = A^*B(D + CA^*B)^* \quad C' = D^*C(A + BD^*C)^* \quad D' = (D + CA^*B)^*$$

On suppose déjà codées les deux fonctions suivantes sur les matrices d'expressions rationnelles :

- (`decouper m n1 n2`) renvoie quatre matrices blocs A, B, C et D telles que A est carrée de taille n_1 , D est carrée de taille n_2 et $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, où la matrice d'entrée M est carrée de taille $n = n_1 + n_2$.

`decouper : mat -> int -> int -> (mat*mat*mat*mat)`

- (`recoller a b c d`) renvoie la matrice $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ à partir des quatre blocs A, B, C et D codés par a, b, c, d dont les tailles sont compatibles.

`recoller : mat -> mat -> mat -> mat -> mat`

On propose la décomposition récursive suivante pour une matrice M carrée de taille $n \geq 2$,

$$M = \begin{pmatrix} a & B \\ C & D \end{pmatrix}$$

où a est une lettre et D est carrée de taille $n - 1$. Ainsi,

$$M^* = \begin{pmatrix} A' & B' \\ C' & D' \end{pmatrix}$$

où

$$A' = (a + BD^*C)^* \quad B' = a^*B(D + Ca^*B)^* \quad C' = D^*C(a + BD^*C)^* \quad D' = (D + Ca^*B)^*$$

Q 38. Évaluer les différentes complexités des sommes et produits effectués, et en déduire que si $C(n)$ est la complexité du calcul de l'étoile pour une matrice de taille n , alors $C(n) = 2C(n - 1) + O(n^2)$. En déduire la complexité de cet algorithme.

On se place dans le cas où la taille n de la matrice M est une puissance de 2, avec $n \geq 2$.

On propose désormais la décomposition récursive suivante

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

où les matrices A et D sont carrées de taille $n/2$ chacune.

Q 39. Évaluer les différentes complexités des sommes et produits effectués, et en déduire que si $C(n)$ est la complexité du calcul de l'étoile pour une matrice de taille n , alors $C(n) = 4C(n/2) + O(n^3)$. En déduire la complexité de cet algorithme.

Q 40. Comment gérer le cas des matrices M de taille n quelconque ? Quelle complexité peut-on obtenir pour le calcul de M^* ?

Q 41. Écrire la fonction `etoile` de signature `mat -> mat` qui renvoie l'étoile d'une matrice en utilisant l'algorithme récursif le plus adéquat.

II.C – Algorithme de Conway

Soit $A = (Q, I, F, T)$ un automate, où l'ensemble d'états est $Q = \llbracket 0, n - 1 \rrbracket$.

On définit M_A la matrice de transition de l'automate A par la matrice d'expressions rationnelles de taille $(n \times n)$ telle que pour $0 \leq i, j \leq n - 1$, $[M_A]_{i,j} = \sum_{c \in \Sigma | (i, c, j) \in T} c$ s'il existe au moins une telle lettre c , \emptyset sinon.

On admet la propriété suivante : pour tout état $(i, j) \in Q^2$, $\mathcal{L}([M_A^*]_{i,j}) = L_{i,j}$, où $L_{i,j}$ est le langage défini en question 9.

Q 42. Montrer que

$$L_A = \mathcal{L}([X M_A^* Y]_{0,0})$$

où X est une matrice ligne d'expressions rationnelles de la forme $X = (x_0 \dots x_{n-1})$ où chaque $x_i \in \{\emptyset, \varepsilon\}$,

et Y est une matrice colonne d'expressions rationnelles de la forme $Y = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix}$ où chaque $y_j \in \{\emptyset, \varepsilon\}$. On précisera les valeurs de X et Y en fonction de l'automate A .

Q 43. Écrire la fonction `langage` de signature `automate -> exprat` prenant en entrée un automate et renvoyant une expression rationnelle représentant le langage de cet automate. Quelle est la complexité de cette fonction ?

III Automate des dérivées d'Antimirov

On propose pour conclure une méthode permettant de calculer un automate non déterministe ayant peu d'états à partir d'une expression rationnelle.

Si S et S' sont deux ensembles d'expressions rationnelles, on convient que

$$S \cdot S' = \{E \cdot E' \mid (E, E') \in S \times S'\}$$

En particulier, on a $\emptyset \cdot S = \emptyset$ et $\{\varepsilon\} \cdot S = S$.

Soit E une expression rationnelle sur un alphabet Σ et soit $a \in \Sigma$ une lettre. On définit la dérivée partielle de E par a , notée $\partial_a(E)$, comme un *ensemble d'expressions rationnelles* défini inductivement par

$$\begin{aligned} \partial_a(\emptyset) &= \emptyset && \text{(où } \emptyset \text{ est l'ensemble vide)} \\ \partial_a(\varepsilon) &= \emptyset \\ \partial_a(b) &= \begin{cases} \{\varepsilon\} & \text{si } a = b \\ \emptyset & \text{sinon} \end{cases} && \text{pour toute lettre } b \in \Sigma \\ \partial_a(E + F) &= \partial_a(E) \cup \partial_a(F) \\ \partial_a(E^*) &= \partial_a(E) \cdot \{E^*\} \\ \partial_a(EF) &= \begin{cases} \partial_a(E) \cdot \{F\} & \text{si } \varepsilon \notin \mathcal{L}(E) \\ \partial_a(E) \cdot \{F\} \cup \partial_a(F) & \text{sinon} \end{cases} \end{aligned}$$

Notons bien que dans une dérivée partielle, on a une expression rationnelle, et que son résultat est un ensemble d'expressions rationnelles.

Par exemple, pour $E = a^*(a + b) = (a^*) \cdot (a + b)$, on calcule $\partial_a(E)$ et $\partial_b(E)$ ainsi : comme $\varepsilon \in \mathcal{L}(a^*)$,

$$\begin{aligned} \partial_a(E) &= (\partial_a(a^*)) \cdot \{a + b\} \cup \partial_a(a + b) \\ &= (\partial_a a) \cdot \{a^*\} \cdot \{a + b\} \cup \partial_a(a) \cup \partial_a(b) \\ &= \{\varepsilon\} \cdot \{a^*(a + b)\} \cup \{\varepsilon\} \cup \emptyset \\ &= \{a^*(a + b); \varepsilon\} \\ \partial_b(E) &= (\partial_b(a^*)) \cdot \{a + b\} \cup \partial_b(a + b) \\ &= (\partial_b a) \cdot \{a^*(a + b)\} \cup \emptyset \cup \{\varepsilon\} \\ &= \emptyset \cup \{\varepsilon\} \\ &= \{\varepsilon\} \end{aligned}$$

Q 44. Pour $E = (ab + b)^*ba$, calculer $\partial_a(E)$ et $\partial_b(E)$.

Cette définition de dérivée partielle est étendue à tout mot $w \in \Sigma^*$ et à des ensembles d'expressions rationnelles par : pour $a \in \Sigma$, $w \in \Sigma^*$ et S un ensemble d'expressions rationnelles,

$$\partial_\varepsilon(E) = \{E\} \quad \partial_{wa}(E) = \partial_a(\partial_w(E)) \quad \partial_w(S) = \bigcup_{E \in S} \partial_w(E)$$

On construit alors l'automate d'Antimirov à partir des dérivées partielles d'une expression rationnelle.

Partons de E une expression rationnelle. L'automate d'Antimirov de l'expression E est $A = (Q, I, F, T)$ défini par

$$\begin{cases} Q = \{E_1 \mid \exists w \in \Sigma^*, E_1 \in \partial_w(E)\} \\ I = \{E\} \\ F = \{E_1 \in Q \mid \varepsilon \in \mathcal{L}(E_1)\} \\ T = \{(E_1, c, E_2) \in Q \times \Sigma \times Q \mid E_2 \in \partial_c(E_1)\} \end{cases}$$

On rappelle la notation $w^{-1}L$ de la partie I : pour tout mot $w \in \Sigma^*$ et tout langage $L \subset \Sigma^*$,

$$w^{-1}L = \{u \in \Sigma^* \mid wu \in L\}$$

Q 45. Dessiner l'automate obtenu à partir de l'expression rationnelle $E = (ab + b)^*ba$. On indiquera précisément l'ensemble d'états Q .

Q 46. Montrer que pour tous mots u, v et tout langage L , $v^{-1}u^{-1}L = (uv)^{-1}L$.

Pour S ensemble d'expressions rationnelles, on note $\mathcal{L}(S)$ la réunion des langages des expressions de S . On admet que, si E est une expression rationnelle et x une lettre,

$$\mathcal{L}(\partial_x(E)) = x^{-1}\mathcal{L}(E)$$

Q 47. Soit S un ensemble d'expressions rationnelles sur Σ et w un mot de Σ^* . Montrer que

$$\mathcal{L}(\partial_w(S)) = w^{-1}\mathcal{L}(S).$$

Q 48. Montrer que pour tout mot $w \in \Sigma^*$, l'ensemble $\partial_w(E)$ est l'ensemble des états accessibles depuis l'état E en lisant le mot w .

Q 49. En déduire que l'automate d'Antimirov reconnaît bien le langage de l'expression rationnelle E .

Pour tout mot $w \in \Sigma^*$ et $w \neq \varepsilon$, et pour toutes expressions rationnelles E et F sur Σ , on vérifie que

$$\partial_w(E + F) = \partial_w(E) \cup \partial_w(F) \tag{III.1}$$

$$\partial_w(EF) \subset \partial_w(E) \cdot F \cup \bigcup_{v \in S^+(w)} \partial_v(F) \tag{III.2}$$

$$\partial_w(E^*) \subset \bigcup_{v \in S^+(w)} \partial_v(E) \cdot E^* \tag{III.3}$$

où $S^+(w)$ est l'ensemble des suffixes non vides d'un mot w .

Pour une expression rationnelle E , on note

$$Q(E) = \bigcup_{w \in \Sigma^*, w \neq \varepsilon} \partial_w(E).$$

Q 50. Montrer que pour toute expression rationnelle E , le cardinal de $Q(E)$ est majoré par le nombre de lettres présentes dans l'écriture syntaxique de E (qu'on notera $\|E\|$). Qu'en déduit-on sur l'automate d'Antimirov ?

• • • FIN • • •

SESSION 2022

MP7IN



ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH

ÉPREUVE SPÉCIFIQUE - FILIÈRE MP

INFORMATIQUE

Durée : 4 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

Le sujet est composé de trois parties, toutes indépendantes.

Partie I - Automates

L'objectif de cette partie est de proposer quelques éléments autour d'un automate, dit augmenté, construit autour d'un automate fini non déterministe et d'un sous-ensemble d'états.

Définition 1 (Automate fini non déterministe)

Un *automate fini non déterministe* est un quintuplet $\mathcal{A} = (Q, \Sigma, I, \delta, F)$ avec :

- Q un ensemble fini non vide d'états, de cardinal $|Q|$;
- Σ un alphabet ;
- $I \subset Q$ l'ensemble des états initiaux ;
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ une fonction de transition : si $q \in Q$ et $a \in \Sigma$, $\delta(q, a)$ désigne l'ensemble des états q' de Q tels qu'il existe une transition étiquetée par a de q vers q' ;
- $F \subset Q$ l'ensemble des états finaux.

Définition 2 (Automate augmenté)

Soient $\mathcal{A} = (Q, \Sigma, I, \delta, F)$ un automate fini non déterministe et $O \subset Q$ un sous-ensemble d'états de Q . On appelle *automate augmenté* par O la paire (\mathcal{A}, O) , que l'on notera par la suite \mathcal{A}_O .

La notion de calcul est la même entre \mathcal{A} et \mathcal{A}_O : un calcul dans \mathcal{A} est une séquence d'états $q_1 \cdots q_n$ de Q telle qu'il existe toujours au moins une transition entre deux états successifs de cette séquence. Le mot construit par concaténation des étiquettes de chacune des transitions est alors reconnu par l'automate.

L'ensemble O introduit la notion de *calcul réussi au seuil s*.

Définition 3 (Calcul réussi au seuil s)

Soit \mathcal{A}_O un automate augmenté. Un calcul passant par les états $q_i \in Q, i \in \llbracket 0, n \rrbracket$ est dit *réussi au seuil s* si :

- i) $q_0 \in I$;
- ii) $q_n \in F$;
- iii) pour tout $i \geq 0$ tel que $i + s \leq n$, $\{q_i \cdots q_{i+s}\} \cap O \neq \emptyset$.

Définition 4 (Langage reconnu par un automate augmenté au seuil s)

Soit \mathcal{A}_O un automate augmenté et $s \in \mathbb{N}$. Le *langage reconnu par \mathcal{A}_O au seuil s*, noté $L_s(\mathcal{A}_O)$, est l'ensemble des calculs réussis par \mathcal{A}_O au seuil s.

On considère, pour les **questions 1 à 3**, l'automate de la **figure 1**, où $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $I = \{q_0\}$ et $F = \{q_2\}$.

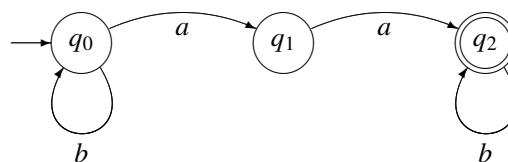


Figure 1 - Automate exemple

- Q1.** Donner sans justification $L_2(\mathcal{A}_{\{q_0\}})$.
- Q2.** Donner, en justifiant votre réponse, $L_2(\mathcal{A}_{\{q_1\}})$.
- Q3.** Soit $s \in \mathbb{N}$. Donner, en justifiant votre réponse, le cardinal de $L_s(\mathcal{A}_\emptyset)$ en fonction de s , où \emptyset est l'ensemble vide.
- Q4.** Donner, sans justification, un majorant des calculs réussis au seuil s ne passant par aucun état de O .

- Q5.** Soit maintenant un calcul réussi au seuil s qui passe au moins par un état de O . On note $\tilde{q} = q_0 \cdots q_l$ la suite des états correspondants et $i_0 \cdots i_k \in \llbracket 0, l \rrbracket$ la suite des indices dans \tilde{q} correspondant aux états de O . Donner, sans justification, en fonction de s :
- i) une majoration de i_0 ;
 - ii) un encadrement de i_k ;
 - iii) une majoration de $i_{j+1} - i_j$ pour $j \in \llbracket 0, k-1 \rrbracket$.
- Q6.** Soit \mathcal{A}_O un automate augmenté, avec $|Q| = p$, où Q est l'ensemble des états de l'automate. Pour $s \in \mathbb{N}$, construire, en justifiant votre construction, un automate fini à $(s+1)p$ états reconnaissant $L_s(\mathcal{A}_O)$.

Partie II - Autour des tas

Cette partie comporte des questions de programmation qui seront abordées en utilisant **exclusivement le langage Python** (Informatique Pour Tous).

L'objectif est ici d'étudier et d'implémenter quelques outils autour d'une structure de données appelée *tas binomial*. Un tas binomial est une structure assez proche du tas binaire (utilisé par exemple pour réaliser une file de priorité), pour lequel la procédure de fusion de deux tas est efficace et peu complexe.

II.1 - Arbre binomial

Définition 5 (Arbre enraciné)

Un *arbre enraciné* est un graphe acyclique orienté possédant une unique racine et tel que tous les nœuds sauf la racine ont un unique parent.

La **figure 2** présente un exemple d'arbre enraciné dans lequel r est la racine.

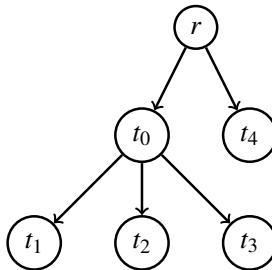


Figure 2 - Exemple d'arbre enraciné

On définit un arbre enraciné (non vide) par un couple $(r, [t_0, \dots, t_{n-1}])$, où r est la valeur de la racine et $[t_0, \dots, t_{n-1}]$ est la liste de ses fils, chaque t_i , $i \in \llbracket 0, n-1 \rrbracket$ étant un arbre. Un arbre vide est défini par `None`. Par abus de notation, on confond dans la suite la racine de l'arbre et sa valeur, ainsi que les fils d'un arbre avec leur racine.

Q7. Écrire les fonctions Python :

- `Vide(a)` qui renvoie `True` si l'arbre `a` est vide, `False` sinon ;
- `Racine(a)` qui renvoie la racine de `a` si `a` est non vide ;
- `Fils(a)` qui renvoie la liste des arbres, fils de la racine de `a`.

Définition 6 (Arbre binomial)

Un *arbre binomial* a_k d'ordre $k \geq 0$ est un arbre enraciné dans lequel les fils de chaque nœud sont ordonnés. Il est défini récursivement comme suit :

- i) $a_0 = (r, [])$ est constitué d'un nœud unique, la racine ;
- ii) pour $k \in \mathbb{N}$, soit $a_k = (r, [t_0, \dots, t_{n-1}])$ un arbre non vide. a_k est un arbre binomial d'ordre k si :
 - t_{n-1} est un arbre binomial d'ordre $(k - 1)$;
 - $(r, [t_0, \dots, t_{n-2}])$ est un arbre binomial d'ordre $(k - 1)$;
 - la racine de t_{n-1} a une valeur supérieure ou égale à r .

La **figure 3** donne un exemple d'arbre binomial d'ordre 3. Les valeurs dans l'arbre sont des entiers.

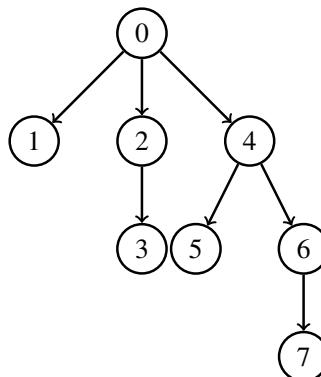


Figure 3 - Exemple d'arbre binomial d'ordre 3

- Q8.** Écrire une fonction Python `ArbreBinomial(a1, a2)` qui construit, à partir de deux arbres binomiaux a_1 et a_2 d'ordre $k - 1$, un arbre binomial a_k d'ordre k contenant les mêmes valeurs que celles des arbres a_1 et a_2 . Dans la suite, on note $a_1 \oplus a_2$ cette opération.
- Q9.** Montrer par récurrence que la racine d'un arbre binomial d'ordre k a exactement k fils.
- Q10.** En déduire une fonction Python `Ordre(a)` qui renvoie l'ordre de l'arbre binomial a .
- Q11.** Montrer qu'un arbre binomial a d'ordre k possède 2^k nœuds.
- Q12.** Écrire une fonction récursive Python `EstUnArbreBinomial(a)` qui renvoie `True` si a est un arbre binomial, `False` sinon.

II.2 - Tas binomial

Un *tas* est une structure de données de type arbre qui permet en particulier de retrouver directement un élément qui doit être traité en priorité.

Définition 7 (Tas binomial)

Soient $k \geq 0$ et $T = \{a_0, \dots, a_k\}$ un ensemble d'arbres. T est un *tas binomial* de longueur $k + 1$ si, pour tout $i \in \llbracket 0, k \rrbracket$, a_i est soit un arbre vide, soit un arbre binomial d'ordre i . Si $i = k$, a_i ne peut, de plus, pas être vide et a_k est donc un arbre binomial d'ordre k .

On note dans la suite $|T|$ le nombre de nœuds d'un tas T .

- Q13.** Quelle structure Python adopter pour coder un tas ?

Définition 8 (Signature d'un tas)

Soit $T = \{a_0, \dots, a_k\}$ un tas binomial de longueur $k + 1$. On appelle *signature* de T la suite $s_0 \dots s_k$ telle que pour tout $i \in \llbracket 0, k \rrbracket$, $s_i = 0$ (respectivement $s_i = 1$) si l'arbre a_i est vide (respectivement n'est pas vide).

Q14. Soit T un tas binomial de longueur $k + 1$. En utilisant sa signature, calculer $|T|$ et montrer que $2^k \leq |T| < 2^{k+1}$. En déduire k en fonction de $|T|$.

Q15. Écrire une fonction Python `MinimumTas(T)` qui retourne la valeur minimum du tas T . En donner la complexité en fonction de $|T|$.

Les tas se construisent itérativement à partir de données. On est donc amené, pour un tas T , à ajouter un à un des éléments.

Soit p un élément que l'on souhaite ajouter à un tas binomial T non vide et déjà construit. L'insertion de la valeur p dans le tas T se fait alors selon l'algorithme 1.

Algorithme 1 - Insertion de p dans T .

Entrées : un tas $T = \{a_0 \dots a_k\}$, une valeur p

Sorties : un tas T augmenté de la valeur p

début

```

 $i \leftarrow 0$ 
Coder  $p$  dans un arbre binomial  $a$  d'ordre 0
tant que  $i < k + 1$  et  $a$  non vide faire
    si  $a_i$  est vide alors
         $a_i \leftarrow a$ 
        Vider  $a$ 
    sinon
         $a \leftarrow a \oplus a_i$  (***)  

        Vider  $a_i$ 
     $i \leftarrow i + 1$ 
si  $a$  n'est pas vide alors
    Ajouter  $a$  au tas  $T$ .

```

Q16. Coder l'algorithme 1 sous la forme d'une fonction Python `Insertion(p, T)`.

Q17. Évaluer la complexité de cet algorithme en fonction de $|T|$. On suppose que l'étape marquée (***) s'effectue en temps constant.

Q18. Donner la signature du tas résultant de l'insertion de p dans T en fonction de la signature de T .

Q19. Donner, sans justification, un invariant de boucle pour la boucle de l'algorithme 1 permettant de prouver la correction de ce dernier.

Partie III - Autour de l'énumération des fractions positives

Cette partie comporte des questions nécessitant un **code OCaml**. Pour ces questions, **les réponses ne feront pas appel aux fonctionnalités impératives du langage** (en particulier pas de boucles, pas de références).

Notations

Dans la suite, on notera :

- $n \wedge d$ le PGCD (Plus Grand Commun Diviseur) de n et d ;
- $\lceil x \rceil$ la partie entière supérieure de x . Ainsi $\lceil 3.45 \rceil = 4$;
- $\lfloor x \rfloor$ la partie entière inférieure de x . Ainsi $\lfloor 3.45 \rfloor = 3$;
- \log_2 la fonction logarithme de base 2. C'est la fonction réciproque de la fonction $i \mapsto 2^i$.

Une *fraction*, ou *nombre rationnel*, est le quotient de deux nombres entiers, le dénominateur étant par définition non nul. On notera \mathbb{Q}^+ l'ensemble des fractions positives.

L'objectif de cette partie est d'étudier une structure de données permettant d'énumérer l'ensemble des fractions positives.

Plusieurs travaux se sont intéressés à l'énumération des nombres rationnels, le plus connu étant très certainement la méthode de Cantor. Cependant, il n'est très souvent pas possible, à moins d'un travail assez complexe, de connaître une formule générale donnant le i -ème terme de cette énumération, ni même de donner le successeur dans l'énumération d'une fraction donnée.

Nous proposons dans la suite de répondre à ces questions en construisant un arbre, dit de Calkin-Wilf, permettant de manipuler cette énumération.

Définition 9 (Arbre binaire infini)

Un *arbre binaire homogène* est un arbre binaire dont tous les noeuds ont 0 successeur ou 2 successeurs, appelés fils gauche et droit. La *hauteur* h de l'arbre est la profondeur maximale des noeuds de l'arbre, c'est-à-dire la plus grande longueur d'un chemin de la racine vers une feuille de l'arbre. Lorsque h est infinie, on parle d'*arbre infini*.

Définition 10 (Arbre de Calkin-Wilf)

L'*arbre de Calkin-Wilf* est un arbre binaire homogène infini dont les noeuds sont des fractions positives. La racine de l'arbre est la fraction $1/1$. Chaque sommet n/d a deux descendants : un fils gauche $n/(n+d)$ et un fils droit $(n+d)/d$.

Ainsi, si $N = n/d$, les deux fils de N sont $\frac{N}{1+N}$ (gauche) et $1+N$ (droit).

- Q20.** Dessiner l'arbre de Calkin-Wilf jusqu'à une profondeur de 3. Par convention, la racine de l'arbre est au niveau 0.

On propose le type record :

```
type fraction = {n:int; d:int}
```

pour définir une fraction de type n/d . La déclaration d'une telle fraction sera donc du type :

```
let f = {n=2;d=3};;
```

l'accès au numérateur (respectivement dénominateur) s'opérant par $f.n$; ; (resp. $f.d$; ;).

- Q21.** Proposer un type OCaml récursif permettant de décrire l'arbre de Calkin-Wilf.
- Q22.** Montrer par récurrence sur le niveau d'exploration de l'arbre que si n/d est un noeud de l'arbre, alors n et d sont premiers entre eux.
- Q23.** Écrire une fonction récursive OCaml de signature $pgcd : \text{int} * \text{int} \rightarrow \text{int}$ qui calcule le PGCD de deux entiers naturels.
- Q24.** Écrire une fonction récursive OCaml de signature $\text{fraction:int} \rightarrow \text{int} \rightarrow \text{fraction}$ qui construit une fraction positive irréductible à partir d'un numérateur n et un dénominateur d . La fonction vérifie que $n > 0$ et $d > 0$. Au besoin, elle simplifie la fraction par $n \wedge d$.

Par la suite, on ne construira des valeurs de type fraction que par l'intermédiaire de la fonction fraction, ce qui permettra de garantir l'invariant de type suivant : "pour toute valeur $f : \text{fraction}$, $f.n$ et $f.d$ sont premiers entre eux".

- Q25.** Soient deux noeuds k/l et n/d de l'arbre ayant des fils gauches (ou droits) identiques. Montrer qu'alors $k = n$ et $l = d$.
- Q26.** Soient N un noeud et $k \in \mathbb{N}$. Montrer que le noeud provenant d'une suite de k fils gauches de N est le noeud $\frac{N}{1+kN}$. Donner sans démonstration le noeud provenant d'une suite de k fils droits de N .

On définit alors une suite $(v_i)_{i \in \mathbb{N}}$, avec $v_0 = 0$, puis en effectuant un parcours en largeur, de gauche à droite, de l'arbre de Calkin-Wilf pour définir les termes de la suite.

- Q27.** Donner les huit premiers termes de la suite $(v_i)_{i \in \mathbb{N}}$.

Q28. Soient $n, d \in \mathbb{N}^*$ premiers entre eux. Montrer par récurrence sur $n + d$ que toute fraction n/d apparaît dans l'arbre.

Q29. Montrer qu'aucune fraction n'apparaît deux fois dans l'arbre.

Q30. Déduire des questions précédentes que la suite $(v_i)_{i \in \mathbb{N}}$ est une bijection de \mathbb{N} dans \mathbb{Q}^+ .

La suite $(v_i)_{i \in \mathbb{N}}$ permet donc d'affirmer que \mathbb{Q}^+ est dénombrable. Nous allons utiliser $(v_i)_{i \in \mathbb{N}}$ pour déterminer, dans l'énumération des fractions produite par cette suite, la position de n'importe quelle fraction positive n/d . En d'autres termes, étant donnée une fraction positive n/d , on se propose de rechercher i tel que $n/d = v_i$.

Q31. Soit $i \in \mathbb{N}^*$. Montrer que le fils gauche du nœud de valeur v_i a pour valeur v_{2i} . Montrer de même que le fils droit a pour valeur v_{2i+1} .

Pour pouvoir énoncer le i -ième terme dans cette énumération, on introduit alors une suite auxiliaire, aux nombreuses propriétés arithmétiques et liens avec d'autres objets mathématiques.

La *suite diatomique de Stern* (ou *suite de Stern-Brocot*) doit son nom à Moritz Stern (1807-1894), élève de Gauss et Achille Brocot (1817-1878), horloger qui s'intéressait aux fractions pour la fabrication d'horloges avec des engrenages comportant peu de dents, donc simples à fabriquer.

Définition 11 (Suite diatomique de Stern ou suite de Stern-Brocot)

La *suite diatomique de Stern* $(s_i)_{i \in \mathbb{N}}$ est définie par $s_0 = 0$, $s_1 = 1$ et pour tout $i \geq 1$:

$$\begin{cases} s_{2i} &= s_i \\ s_{2i+1} &= s_i + s_{i+1} \end{cases}.$$

Q32. Donner les dix premiers termes de la suite $(s_i)_{i \in \mathbb{N}}$.

Q33. Écrire une fonction récursive OCaml de signature `stern : int -> int` permettant de calculer les termes de cette suite.

Q34. Déduire par récurrence de la **Q31** que : $\forall i \in \mathbb{N}, v_i = \frac{s_i}{s_{i+1}}$.

La suite diatomique de Stern permet d'exprimer le i -ième terme dans l'énumération des fractions positives qui est induite par le parcours en largeur de l'arbre de Calkin-Wilf décrit ci-dessus.

Voyons maintenant comment obtenir de manière rapide le successeur d'une fraction v_i donnée, sans connaître i , dans cette énumération. Trois cas peuvent se présenter :

- premier cas : les nœuds de valeur v_i et v_{i+1} sont à même profondeur k , fils d'un même nœud N ,
- deuxième cas : le nœud de valeur v_i est le dernier nœud à droite à la profondeur k ,
- troisième cas : les nœuds de valeur v_i et v_{i+1} sont à même profondeur k , mais ne sont pas fils d'un même nœud.

On pose pour tout $x > 0$, $f(x) = \frac{1}{1 + 2\lfloor x \rfloor - x}$.

Q35. Montrer que dans le premier cas, $v_{i+1} = f(v_i)$.

Q36. Montrer, en utilisant la **Q26**, que dans le deuxième cas on a encore $v_{i+1} = f(v_i)$.

On étudie enfin le dernier cas : les nœuds de valeur v_i et v_{i+1} sont sur une même profondeur k , mais ne sont pas les fils d'un même nœud. On va donc passer par la recherche d'un ancêtre commun de ces deux nœuds. Dans la suite, on s'intéressera toujours au premier ancêtre commun, c'est-à-dire celui de profondeur maximale.

En partant de la racine r , il est possible d'atteindre n'importe quel nœud $N = n/d$ de l'arbre par une suite de déplacements vers la gauche (G) ou vers la droite (D). Le *chemin* de r vers N peut donc être codé par un mot sur l'alphabet {G, D}.

Si un déplacement est représenté en OCaml par un type

```
type direction = G | D
```

alors un chemin est une liste direction list.

- Q37.** Écrire une fonction OCaml de signature `chemin : fraction -> direction list` qui calcule le chemin de la racine à un nœud quelconque de l'arbre. Cette fonction fera appel à une fonction auxiliaire récursive. Ainsi `chemin n d` calcule la liste des directions à prendre pour passer de r au nœud n/d . On pourra supposer l'existence d'une fonction de signature `rev : direction list -> direction list` qui renverse une liste.
- Q38.** Écrire une fonction OCaml de signature `noeud : direction list -> fraction` qui détermine le nœud obtenu en effectuant une série de déplacements depuis la racine. Ainsi `noeud [D;D;G;D]` renverra un couple d'entiers (n, d) correspondant au nœud de valeur n/d . Cette fonction fera appel à une fonction auxiliaire récursive.
- Q39.** Utiliser les deux fonctions précédentes pour écrire une fonction OCaml `ancetre` de signature `ancetre : fraction -> fraction -> fraction` qui détermine le premier ancêtre commun entre deux nœuds. Ainsi `ancetre (n,d) (p,q)` détermine le premier ancêtre commun à n/d et p/q . Cette fonction pourra utiliser une fonction auxiliaire récursive.
- Q40.** On suppose que le nœud de valeur v_p , le premier ancêtre commun des nœuds de valeur v_i et v_{i+1} , est à k' niveaux au-dessus d'eux. Donner le chemin entre v_p et v_i sous la forme d'une liste de direction. Donner de même le chemin entre v_p et v_{i+1} .
- Q41.** À partir de l'expression des fils gauche et droit de v_p , montrer que l'on a encore $v_{i+1} = f(v_i)$.

FIN



EBE NSI 1

SESSION 2022

CAPES ET CAFEP/CAPES**CONCOURS EXTERNE
TROISIÈME CONCOURS****Section****NUMERIQUE ET SCIENCES INFORMATIQUES****ÉPREUVE ÉCRITE DISCIPLINAIRE**

Durée : 5 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique est rigoureusement interdit.

- Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence.
- De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

Épreuve disciplinaire

Étude de protocoles de traçage de contacts numériques

Dans le domaine de la santé, le *traçage de contacts* (dénommé TC par la suite) est un processus permettant d'identifier les individus qui ont pu être en contact avec une personne infectée. L'objectif du TC au cours d'une épidémie est de réussir à identifier rapidement les personnes ayant pu être contaminées afin de les tester, les traiter si besoin et/ou les isoler.

Nous nous intéressons à deux protocoles de TC automatisés (les protocoles HAMAGEN et ROBERT) qui ont été mis en œuvre durant la pandémie de la Covid19 dans différents pays. Le but est d'en proposer une implémentation et d'en discuter les propriétés en terme de sécurité.

Par la suite, il sera supposé que les individus disposent tous d'un téléphone portable, et qu'il y a une infrastructure centralisatrice (par exemple un cloud ou un serveur), appelée IC par la suite, qui est sous le contrôle d'une entité bien identifiée (par exemple le ministère de la santé).

Toutes les fonctions demandées par la suite devront être rédigées en langage Python.

Le problème est constitué de 6 exercices qui peuvent être traités de manière indépendante, en admettant le résultat des questions précédentes si nécessaire. Certains concepts sont utilisés tout au long du problème, ils sont présentés dans la première partie du sujet.

Les réponses aux questions devront être précises et rédigées avec soin.

Concepts et notations nécessaires

Les protocoles de traçage de contacts reposent sur des concepts décrits par la suite. Nous donnons ici la manière dont les structures de données, associées à ces concepts, sont représentées en Python.

Point de localisation : la localisation des individus est réduite à un point, représenté en Python par un couple de réels (`lat`, `long`) désignant la latitude et la longitude de l'individu sur le globe terrestre.

Temps : le protocole nécessite de mémoriser la position passée des individus sur leur téléphone avec une périodicité P exprimée en minutes. Pour plus de simplicité, on se donne une date origine D_0 commune à tous les individus et on compte les dates de mémorisation D_e à partir de cette date d'origine, de la manière suivante :

$$D_e = D_0 + e \times P$$

où l'entier naturel e (dénommé *époque* par la suite) désigne le nombre d'intervalles écoulés depuis D_0 .

Dans la suite, nous supposerons que la périodicité est de 15 minutes ($P = 15$).

Trace : le téléphone de chaque utilisateur contient un dictionnaire `trace` dont la clé est une époque (un entier naturel) et la valeur est un point de localisation (un couple de réels (`lat`, `long`)) désignant la position de l'individu à l'époque considérée. Pour des raisons de simplicité, on ne considérera ici que des latitudes Nord et longitudes Est. Ainsi le dictionnaire `{4: (10.6, 12.3), 5: (7.2, 1.0)}` signifie qu'à l'époque 4, l'individu était localisé à la latitude $10^{\circ}6'N$ et la longitude $12^{\circ}3'E$ tandis qu'à l'époque 5, il était situé à la latitude $7^{\circ}2'N$ et la longitude $1^{\circ}0'E$.

Durée de contagiosité : on fait l'hypothèse qu'il existe une durée maximale pendant laquelle les personnes sont contagieuses. Cette durée est notée `Cont` et est exprimée en nombre de jours.

Traces à risques : l'infrastructure centralisatrice maintient de son côté une liste `listeAR` mémorisant les traces considérées à risque. Une trace à risque correspond à une trace d'une personne qui a été testée positive. La structure de données `listeAR` est donc d'une liste de dictionnaires correspondant aux traces de mobilité des personnes ayant été testées positivement à la Covid19. La durée des traces correspond à la durée où les personnes étaient potentiellement contagieuses. Cette liste peut ensuite être téléchargée librement par n'importe quel individu. Les personnes sont identifiées par leur position (arbitraire) dans la liste.

Ainsi la liste `[{4: (10.6, 12.3), 5: (7.2, 1.0)}, {1440: (243.0, 12.1)}]` indique les traces de deux individus avec leurs positions aux époques mémorisées par le système de traçage et pour lesquelles ils sont considérés à risque. Les données du premier individu concernent les époques 4 et 5, les données du 2e individu ne concernent que l'époque 1440.

Exercice 1 : Protocole *HaMagen*

Le protocole de traçage de contacts *HaMagen* repose uniquement sur les notions présentées juste avant dans la partie introductive. Dans cette partie nous nous intéressons à son implémentation en Python dans une version simplifiée.

Question 1.1

Avec les structures de donnée Python choisies, est-il possible de stocker la position de l'individu aux époques 4 et 6, sans connaître sa position à l'époque 5 ? Justifier brièvement.

Principe du protocole *HaMagen*.

Le fonctionnement du protocole *HaMagen* repose sur le principe suivant : la trace d'un individu est stockée sur son téléphone portable. Si jamais l'utilisateur est testé positif à la Covid19 à une époque e , alors les 14 derniers jours de sa trace sont téléchargés sur l'infrastructure centralisatrice ($Cont = 14$). La liste des traces à risque, `listeAR`, est mise à jour en ajoutant cette nouvelle trace à la fin de `listeAR`.

Indépendamment, toute personne peut télécharger régulièrement la liste des traces à risque, `listeAR`, depuis l'infrastructure centralisatrice sur son téléphone. Ceci lui permet de comparer les traces à risque téléchargées avec sa propre trace stockée sur son téléphone. Si jamais certains points de localisation de sa trace sont proches d'une ou plusieurs traces à risque, alors la personne est identifiée comme cas contact.

Question 1.2

Écrire une fonction python `débutAR` qui, étant donnée une époque e à laquelle un individu est testé positif, renvoie l'époque correspondant au début de la période à risque, c'est-à-dire l'entier naturel désignant l'époque 14 jours avant e .

Remarque : On rappelle que la période d'enregistrement des localisations est supposée de 15 minutes et que la valeur de l'époque renournée doit être supérieure ou égale à 0.

Question 1.3

En déduire une fonction python `extraitTraceAR` qui, étant données la trace `tr` d'un individu et une époque e à laquelle il est détecté positif, renvoie un dictionnaire construit à partir de `tr` contenant uniquement les époques et localisations considérées comme à risque.

Question 1.4

Montrer que la taille du dictionnaire construit avec la fonction `extraitTraceAR` est bornée par une constante qui ne dépend que de P et $Cont$. Dans quel cas cette borne est-elle atteinte ?

Implémentation du protocole *HaMagen*.

Chaque personne peut donc à tout moment vérifier si elle est devenue une personne à risque via l'algorithme 1 décrit ci-dessous.

Dans cet algorithme, on suppose implémentées les fonctions `époque()`, `posGPS()` et `recupTraces()` qui permettent respectivement d'obtenir l'époque courante, la position courante du téléphone portable ainsi que la liste, récupérée depuis l'infrastructure centrale, contenant les traces considérées à risque.

La dernière fonction, `aCroise(t1, t2, dContact)`, reste à implémenter. Elle permet de savoir si l'individu ayant produit la trace `t1` a croisé l'individu ayant produit la trace `t2` à une distance inférieure ou égale à `dContact` à au moins une époque par le passé.

Algorithme 1 : Protocole HaMagen simplifié

Entrée : `dContact` : réel exprimant la distance, en mètre, en dessous de laquelle deux personnes sont considérées être en contact

Structure de données : `maTrace` : dictionnaire (au départ vide) contenant la trace de l'utilisateur

```

1 maTrace = dict() ;
2 aRisque = False ;
3 Tant que not aRisque faire
4   | maTrace[époque()] = posGPS() ;
5   | listeAR = recuperTraces() ;
6   | Pour tr in listeAR faire
7     |   | Si aCroise(tr, maTrace, dContact) alors
8     |   |   | aRisque = True;
9     |   | Fin
10    | Fin
11    attendre 15 minutes ;
12 Fin
13 print('Vous venez d'être identifié comme cas contact.') ;

```

Question 1.5

Soient deux points $p_1 = (\text{lat}_1, \text{long}_1)$ et $p_2 = (\text{lat}_2, \text{long}_2)$ sur la Terre, tous deux localisés par rapport au Nord et à l'Est, avec des angles compris entre 0° et 90° . La distance entre ces deux points peut se calculer en première approximation à l'aide du Théorème de Pythagore (pour des points assez proches) par :

$$d = k \times \sqrt{x^2 + y^2}$$

avec $x = \text{lat}_1 - \text{lat}_2$, $y = (\text{long}_1 - \text{long}_2) \times \cos\left(\frac{\text{lat}_1 + \text{lat}_2}{2}\right)$ et $k = 111\,120$ un facteur multiplicatif permettant d'exprimer le résultat en mètres.

Écrire une fonction Python `proximite` qui, étant donnés deux points `p1` et `p2` ainsi qu'une distance `dContact` exprimée en mètres, renvoie `True` si les points sont à une distance inférieure ou égale à `dContact` et `False` sinon.

Question 1.6

En déduire l'écriture d'une fonction Python `aCroise(t1, t2, dContact)` qui, étant données deux traces `t1` et `t2` ainsi qu'une distance `dContact` exprimée en mètres, renvoie `True` s'il existe au moins une époque durant laquelle un point p_1 de `t1` et un point p_2 de `t2` sont à une distance inférieure à la distance `dContact`. Elle renvoie `False` sinon.

Question 1.7

Quels sont les paramètres nécessaires pour exprimer la complexité de la fonction `aCroise`? Quelle est la complexité de la fonction dans le pire des cas?

Exercice 2 : Requêtes SQL : stockage des données sur l'infrastructure centralisatrice

La fonction `recuperTraces()` utilisée dans l'algorithme 1 effectue un appel à l'infrastructure centralisatrice (IC) pour récupérer les traces de tous les utilisateurs positifs à la Covid19 qui ont accepté de téléverser leurs données sur le serveur. Nous faisons l'hypothèse que ces informations sont stockées sur l'IC dans une base de données relationnelle, dont le schéma est le suivant (clés primaires soulignées, clés étrangères indiquées par une contrainte REFERENCES) :

- `users` (`id`, `numTel`)
- `traces` (`id`, `époque`, `latitude`, `longitude`)
- `traces.id` REFERENCES `users.id`

La base de données est donc composée de deux tables : `users` qui fait le lien entre un numéro de téléphone et un identifiant arbitraire généré par l'IC (`id`), et `traces` qui contient les informations de position GPS pour un `id` spécifique et une `époque` donnée. Le numéro de téléphone ne sera pas transmis à d'autres utilisateurs de l'application et reste simplement sur l'IC. Il sert à vérifier, au moment du téléchargement, que toutes les traces d'un même utilisateur sont bien affectées au bon `id`.

L'algorithme de téléversement est donc le suivant : un utilisateur détecté positif à la Covid19 télécharge ses positions GPS une par une, en associant son numéro de téléphone. Lorsque le serveur reçoit une position, il vérifie si le numéro de téléphone existe déjà. S'il n'existe pas, il génère un nouvel identifiant **id**, insère le couple (**id**, **numTel**) dans la table **users**, puis insère la position dans la table **traces** en utilisant ce nouvel identifiant. Si le numéro de téléphone existe déjà, l'IC retrouve l'**id** correspondant, n'insère rien dans la table **users**, et insère la position dans la table **traces** avec le bon identifiant.

Dans les questions qui suivent, on ne s'intéresse pas à la manière dont les traces sont échangées entre le téléphone et l'IC, mais simplement à la manipulation de ces traces sur le serveur.

Question 2.1

Donner la requête SQL qui retourne l'**id** correspondant au numéro de téléphone 0123456789, si celui-ci est présent dans la base.

Question 2.2

Donner la requête SQL qui retourne combien d'utilisateurs (identifiés de manière unique par leur numéro de téléphone) sont présents dans la base.

Question 2.3

Donner la requête SQL qui retourne l'ensemble des positions (latitude et longitude) de l'utilisateur correspondant au numéro de téléphone 0123456789.

Question 2.4

Donner la requête SQL qui retourne le nombre de positions enregistrées sur le serveur pour chaque utilisateur, identifié de manière unique par son numéro de téléphone.

Question 2.5

Donner la requête SQL qui retourne combien d'utilisateurs ont laissé plus de 1000 enregistrements de position sur le serveur.

Question 2.6

On suppose la base de données vide, et qu'un **commit** est exécuté après chaque insertion. L'IC exécute les instructions suivantes :

1. INSERT INTO traces VALUES(1, 0, 1.52, 1.55)
2. INSERT INTO traces VALUES(1, 1, 1.53, 1.55)
3. INSERT INTO users VALUES(1, 0123456789)

Indiquer le contenu de la base de données une fois ces trois instructions exécutées. Expliquer.

Question 2.7

On suppose la base de données vide, et qu'un **commit** est exécuté après chaque insertion. L'IC exécute les instructions suivantes :

1. INSERT INTO users VALUES(1, 0123456789)
2. INSERT INTO users VALUES(2, 0987654321)
3. INSERT INTO traces VALUES(1, 0, 1.52, 1.55)
4. INSERT INTO traces VALUES(1, 1, 1.53, 1.55)
5. INSERT INTO traces VALUES(2, 0, 1.52, 1.55)
6. INSERT INTO traces VALUES(2, 0, 1.53, 1.55)

Indiquer le contenu de la base de données une fois ces six instructions exécutées. Expliquer.

Exercice 3 : Sécurité et vie privée : confidentialité des données sur le serveur

La requête proposée à la question 2.3 montre que l'administrateur de la base de données est capable de retrouver toutes les traces de tous les individus dont il connaît le numéro de téléphone. On cherche donc à modifier l'infrastructure pour résoudre ce problème en chiffrant l'information sensible qu'est le numéro de téléphone.

Question 3.1

Est-ce que chiffrer les communications entre les individus et l'IC, comme le fait le protocole HTTPS pour le Web, permettrait de résoudre le problème mentionné ci-dessus ? Justifier.

Chiffrement

On décide désormais de ne pas stocker le numéro de téléphone `numTel`, mais plutôt sa valeur chiffrée $E(\text{numTel})$, en utilisant un algorithme de chiffrement symétrique, paramétré par une clé de chiffrement KS , que l'on supposera inaccessible à quiconque, y compris les administrateurs de l'infrastructure centralisée. Ainsi un ajout dans la base de données se fera en rajoutant un appel à une fonction E de chiffrement avec la commande `INSERT INTO users VALUES(1, E('0123456789'))`.

Pour vérifier si un numéro de téléphone est déjà présent dans la base, l'IC cherche si la valeur chiffrée de ce numéro de téléphone est présent dans la table `users`.

On suppose par ailleurs que la fonction de chiffrement E est une fonction qui peut être appelée depuis n'importe quelle requête SQL, sans avoir besoin de connaître la clé KS .

Question 3.2

On considère que E est à valeurs dans un ensemble fini. Est-il possible d'avoir la propriété suivante, correspondant à une *collision* : il existe X, Y deux numéros de téléphone tels que $E(X) = E(Y)$ et $X \neq Y$ si E est à valeurs dans un ensemble fini de taille 2^{16} ? Que se passerait-il si E était à valeurs dans un ensemble fini de taille 2^{64} ? On ne demande pas de calculer la probabilité de collision.

Question 3.3

Est-il possible pour l'administrateur de la base de données de retrouver les traces d'un individu dont il connaît le numéro de téléphone, maintenant que les numéros de téléphone sont chiffrés dans la base ? Justifier.

Sécurité du protocole *HaMagen*.

Tel qu'il est défini dans l'algorithme 1, le protocole présente la vulnérabilité suivante : les points de localisation correspondant à un individu peuvent tous être liés à la même trace (celle de l'individu). Ainsi, si on est capable de retrouver la position d'un individu à un moment donné, on sera capable de connaître l'ensemble de sa trace (sur les $Cont = 14$ derniers jours).

Principe de l'attaque du protocole *HaMagen*.

Mme X (une espionne par exemple) achète un téléphone et installe l'application. Elle allume ensuite le GPS et l'application pendant k époques T_{on} à T_{on+k-1} , durant lesquelles elle se retrouve seule à proximité d'un individu cible (Mr Y) qu'elle souhaite pister. Elle éteint ensuite son GPS et son téléphone, pour ne le rallumer que lorsqu'elle est sûre d'être seule, afin de récupérer l'ensemble des traces à risque depuis le serveur. Si à un moment donné, dans les traces récupérées sur son téléphone, elle retrouve une trace qui a croisé la sienne, alors elle peut être sûre de deux choses : 1) Mr Y a été testé positif à la Covid19 et a envoyé sa trace à l'IC, et 2) tous les points de la trace qu'elle a croisés sont les points de Mr Y. Mme X sera donc en mesure de retracer les déplacements des 14 derniers jours de Mr Y.

Question 3.4

En vous inspirant de l'algorithme 1, proposer un algorithme renvoyant la trace de l'individu ciblé en réalisant l'attaque décrite ci-dessus.

Exercice 4 : Détection de "hot spots"

L'autorité gérant l'IC possède de nombreuses données sur les personnes ayant été testées positives à la Covid19 et s'étant déclarées sur l'application *HaMagen*.

L'IC souhaite détecter des *hot spots*, c'est-à-dire des endroits où de nombreuses personnes testées positives à la Covid19 se sont trouvées. On fait l'hypothèse que si une trace est présente dans la base de données de l'IC, alors la personne était positive à la Covid19, et donc que l'IC ne contient que des traces de personnes positives à la Covid19 (il n'y a donc pas de faux positifs).

Remarque : Dans toute ce qui suit, vous pouvez, si besoin, vous servir des fonctions proposées dans l'exercice 1.

Question 4.1

Ecrire une fonction Python `estHotspotEpoque(p, listeAR, k, dContact, e)` qui, étant donnés un point de localisation `p`, une liste de dictionnaires `listeAR` contenant les traces à risques, un entier `k`, un réel `dContact` et un entier `e`, renvoie `True` si au moins `k` individus présents dans la liste `listeAR` sont présents à l'époque `e` à une distance inférieure ou égale à `dContact` du point `p`.

Question 4.2

Donner la complexité de votre fonction dans le pire des cas. Justifier.

Question 4.3

En déduire une fonction Python `estHotspot(p, listeAR, k, dContact)` qui renvoie `True` si le point `p` de localisation est un hotspot, c'est-à-dire si au moins `k` individus de la liste `listeAR` sont présents à une même époque, à une distance inférieure ou égale à `dContact` de `p`.

Question 4.4

Donner la complexité de cette fonction dans le pire des cas. Justifier.

On s'interroge maintenant sur les points de localisation à tester avec la fonction `estHotspot`. On pourrait par exemple tester toutes les positions de France avec une certaine granularité mais ce serait peu efficace.

Question 4.5

Dans un premier temps, on se propose d'utiliser la fonction `mystere(listeAR, k, dContact)` suivante, dont le programme est donné en Python :

```
def mystere(listeAR, k, dContact):
    IHP = []
    for tr in listeAR:
        for e in tr:
            p = tr[e] # prochain point à tester
            if estHotspot(p, listeAR, k, dContact):
                IHP.append(p)
    return IHP
```

Expliquer ce que fait cette fonction et ce qu'elle renvoie.

Question 4.6

Quelle est la complexité de la fonction `mystere(listeAR, k, dContact)` dans le pire des cas ?

On se propose d'éviter, par la suite, de considérer comme hotspots des points qui sont trop proches les uns des autres.

Question 4.7

Proposer une amélioration de la fonction précédente en écrivant une fonction `listHotspotsDistincts(listeAR, k, dContact)` qui renvoie une liste contenant les points de localisation de `listeAR` qui sont des hotspots mais pour lesquels aucun autre point de la liste renvoyée n'est à une distance inférieure ou égale à `dContact`.

Question 4.8

Compléter les blocs ***(1)*** et ***(2)*** de la fonction Python `recupTraces()` donnée ci-dessous. Cette fonction retourne une liste `listeAR` à partir des données stockées dans la base de données de l'IC. Le bloc ***(1)*** représente une requête SQL, et le bloc ***(2)*** peut contenir plusieurs instructions Python.

Note : étant donnée une requête SQL de la forme `SELECT A0, A1, ...AN FROM T WHERE ...`, l'instruction `ligne[i]` permet d'accéder à la valeur de l'attribut A_i de la clause `SELECT` si on l'utilise dans une boucle `for ligne in liste:`.

```
import sqlite3
def recuperTraces():
    conn = sqlite3.connect('baseDonnees.db')
    cur = conn.cursor()
    cur.execute(***(1)***)
    liste = cur.fetchall()
    ****(2)***"
    conn.close()
    return listeAR
```

Exercice 5 : Algorithmique de graphes

On souhaite maintenant étudier la structure relationnelle établie entre les personnes contaminées afin d'identifier celles qui sont les plus centrales. Pour cela, on construit un graphe social non orienté et non pondéré `GraphCovid` dont les sommets sont des personnes (identifiées par un entier) et dont les arêtes relient deux personnes qui ont été en contact à une même époque.

On supposera par la suite que le graphe est décrit par une liste d'adjacence, implémentée en Python par un dictionnaire dont les clés sont les identifiants des personnes et les valeurs sont les listes de sommets voisins dans le graphe. On supposera par ailleurs le graphe non vide et connexe.

On considérera par la suite le graphe `GraphCovid` décrit par la structure suivante :

```
GraphCovid = {
    1: [2, 3, 4, 5],
    2: [1],
    3: [1],
    4: [1],
    5: [1, 6],
    6: [5, 7, 8],
    7: [6, 9, 10],
    8: [6, 10, 11],
    9: [7],
    10: [7, 8],
    11: [8]
}
```

Question 5.1

Donner une représentation graphique de la structure `GraphCovid`.

Pour identifier les personnes les plus centrales dans un graphe social $G = (V, E)$, plusieurs métriques peuvent être utilisées comme :

Degré : le degré d'un sommet s correspond à son nombre de voisins dans le graphe.

Centralité de proximité : la centralité de proximité d'un sommet s est définie par l'inverse de la somme des distances de ce sommet vers tous les autres sommets du graphe :

$$C_{prox}(s) = \frac{1}{\sum_{x \in V \setminus \{s\}} dist(s, x)}$$

où $dist(s, x)$ correspond à la longueur d'un plus court chemin entre s et x dans G .

Question 5.2

Sur le graphe **GraphCovid** précédent, quel sommet a le plus fort degré et quelle est sa valeur ? Quel sommet a la plus forte centralité de proximité et quelle est sa valeur parmi les sommets 1, 5 et 6 ?

Question 5.3

Écrire une fonction Python **plusFortDegre(G)** qui, étant donné un graphe G , renvoie l'identifiant du sommet de plus fort degré.

On souhaite maintenant déterminer un sommet de plus forte centralité de proximité. Pour cela nous allons nous appuyer sur l'algorithme de parcours en largeur d'un graphe, qui est donné ci-dessous :

Algorithme 2 : Parcours en largeur d'un graphe

Entrée : G , un graphe

Entrée : s , identifiant du sommet de départ

```

1 F = FileVide();
2 Enfiler(F,s);
3 Marquer(s);
4 Tant que F non vide faire
5   x=Defiler(F);
6   Pour y voisin de x dans G faire
7     Si y non marqué alors
8       Enfiler(F,y);
9       Marquer(y);
10    Fin
11  Fin
12 Fin
```

Question 5.4

Pourquoi l'algorithme de parcours en largueur est utile pour calculer la centralité de proximité d'un sommet ?

Question 5.5

En adaptant l'algorithme 2, écrire un algorithme **Dist** qui, étant donné un graphe G et un sommet de départ s , permet de calculer et de renvoyer les distances entre s et tous les autres sommets de G .

Question 5.6

En déduire un algorithme **SommetProxMax** qui, étant donné un graphe G , renvoie un sommet de centralité de proximité maximale dans G .

Question 5.7

Que peut-on dire de l'intérêt des deux métriques proposées dans le contexte de la propagation de proche en proche d'une épidémie ? L'une des métriques semble-t-elle plus pertinente que l'autre pour identifier les individus cruciaux dans la dynamique de propagation ? Pourquoi ?

Exercice 6 : Protocole ROBERT

Le principal problème qu'on retrouve dans l'algorithme HaMagen est que les données collectées sur les personnes atteintes de la Covid19 peuvent faire fuiter des données privées, *i.e.* leur position à une époque donnée. Afin de ne pas collecter des données aussi précises que la position GPS, d'autres algorithmes respectant le RGPD ont été proposés, notamment par un consortium d'industriels (Apple et Google), mais aussi par des équipes de recherche (avec le protocole ROBERT). Ces deux protocoles (celui proposé par des industriels et le protocole ROBERT) n'utilisent pas la position GPS des individus, mais la technologie de communication *Bluetooth*, qui va permettre de détecter d'autres appareils à proximité, sans pour autant connaître la position où ils se trouvent.

Question 6.1

Qu'est-ce que le RGPD et quel est le principe appliqué ici ?

Protocole ROBERT simplifié

1. Un utilisateur u s'inscrit au protocole, en contactant l'IC. L'IC génère un identifiant unique ID_u pour l'utilisateur u et lui communique cet identifiant. L'IC génère également une clé symétrique SK_u et envoie SK_u à l'utilisateur u . Pour simplifier, on considérera que SK_u permet à la fois de chiffrer et d'authentifier les messages entre l'IC et l'utilisateur u . L'application de l'utilisateur u conserve au secret la clé SK_u . L'IC connaît également une clé secrète K_S .
2. Pour chaque époque i , l'IC calcule un identifiant dit “éphémère” déduit de ID_u par i applications d'une fonction Φ :

$$EphID_i(u) = \Phi^i(ID_u \oplus K_S)$$

\oplus étant l'opérateur logique XOR.

La fonction Φ est une fonction de *hachage cryptographique* à valeurs dans un ensemble de taille 2^{64} , et utilisant une clé secrète K_S de même longueur que ID_u . Ces identifiants sont communiqués par groupe de d identifiants (par exemple une fois par jour l'IC envoie à chaque utilisateur $d = 96$ identifiants à utiliser pendant la journée correspondant aux identifiants éphémères qui seraient calculés aux 96 époques de la journée). L'envoi de ces identifiants est chiffré par le serveur avec la clé SK_u .

3. Entre chaque époque i et chaque époque $i + 1$, le téléphone de l'utilisateur u émet régulièrement (par exemple chaque seconde) $EphID_i(u)$ via la technologie de communication Bluetooth.
4. Chaque téléphone stocke l'ensemble des $EphID_i$ distincts qu'il capte à proximité via Bluetooth pour chaque époque i .
5. Si jamais l'utilisateur v est diagnostiqué positif à la Covid19, alors le téléphone de l'utilisateur v envoie à l'IC un message incluant l'ensemble ϵ_v contenant tous les $EphID$ qu'il a reçus et les époques associées au cours des $Cont = 14$ derniers jours. L'infrastructure centralisatrice est alors capable de retrouver l'ensemble des utilisateurs qui ont été exposés, en retrouvant les ID_u à partir des listes envoyées par les individus ayant été testés positifs à la Covid19.
6. Régulièrement (par exemple une fois par jour), le téléphone de chaque utilisateur se connecte et demande s'il a été identifié parmi la liste des contacts possibles. Si c'est le cas, l'IC l'en informe, et l'utilisateur est invité à se faire tester.

On rappelle qu'une fonction de hachage est une fonction dont le code est public et qui est à valeurs dans un ensemble fini de taille Ω (par exemple $\Omega = 2^{64}$). Comme il faut utiliser le résultat de l'application de Φ comme paramètre d'entrée de Φ , les ID_u peuvent donc être codés sur $\log_2(\Omega)$ bits au plus. On fait également ici l'hypothèse que Φ est bijective sur Ω .

Question 6.2

Comme la fonction Φ est publique et que l'utilisateur connaît son ID_u , est-ce qu'un utilisateur u est capable de calculer $EphID_i(u)$ étant donné i ?

Question 6.3

On note n le nombre d'utilisateurs au système de traçage ROBERT. Donner, en fonction de n, Ω et du nombre d'époques qui se sont écoulées (qu'on notera \mathcal{E}), la probabilité P_{dist} de ne générer que des valeurs distinctes pour les identifiants éphémères des n utilisateurs pour les \mathcal{E} applications successives de Φ .

Question 6.4

Si on se donne $n = 2$ et $\Omega = 2^{10}$, $P_{dist} \approx 0.994$ pour $\mathcal{E} = 2$ et $P_{dist} \approx 0.985$ pour $\mathcal{E} = 3$. Que pouvez-vous en déduire ? Justifier l'intérêt de ce calcul de probabilité.

Question 6.5

On fait l'hypothèse que w_1 et w_2 ont croisé au même moment une personne contaminée (utilisateur v). Est-ce qu'il est possible pour l'IC de savoir que w_1 et w_2 ont été en contact ? Est-il possible pour l'IC de savoir que w_1 , w_2 et v ont été en contact ?

Question 6.6

Le protocole ROBERT complet impose des contraintes supplémentaires lors du téléversement des données par un utilisateur sur l'IC :

1. la liste ϵ_v n'est pas envoyée en une seule fois, mais chaque *EphID* de la liste ϵ_v , avec l'époque associée, est envoyé par un message indépendant $m_{v,j}$ (pour $j \in [1; |\epsilon_v|]$) ;
2. chaque message $m_{v,j}$ est envoyé via un *mixnet*.

Un *mixnet* est une structure d'anonymisation qui s'intercale entre un expéditeur et un récepteur et qui ne permet pas au récepteur de connaître l'adresse IP de l'expéditeur. D'autre part, un *mixnet* n'envoie pas forcément tous les messages dès qu'il les reçoit. Par exemple, un *mixnet* peut attendre avant d'envoyer un message, ou encore inverser l'ordre d'envoi des messages provenant d'utilisateurs différents ou du même utilisateur.

Expliquer comment ces deux contraintes supplémentaires participent à la sécurité du protocole ROBERT.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.
Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours externe du CAPES

- Enseignement public :

Concours

E	B	E
---	---	---

Section/option

6	9	00	E
---	---	----	---

Epreuve

1	0	1
---	---	---

Matière

9	3	11
---	---	----

- Enseignement privé :

Concours

E	B	F
---	---	---

Section/option

6	9	00	E
---	---	----	---

Epreuve

1	0	1
---	---	---

Matière

9	3	11
---	---	----

Troisième concours du CAPES

- Enseignement public :

Concours

E	B	V
---	---	---

Section/option

6	9	00	E
---	---	----	---

Epreuve

1	0	1
---	---	---

Matière

9	3	11
---	---	----



EBE NSI 2

SESSION 2022

CONCOURS EXTERNE CAPES ET CAFEP/CAPES CORRESPONDANTS

Section

NUMERIQUE ET SCIENCES INFORMATIQUES

ÉPREUVE DISCIPLINAIRE APPLIQUÉE

Durée : 5 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique est rigoureusement interdit.

- Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence.
- De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

A

Épreuve disciplinaire appliquée

Préambule : cette épreuve est constituée de deux problèmes indépendants. Les réponses aux questions devront être précises et rédigées avec soin.

Problème 1 - Gestion des données

1 Introduction à la problématique des données

Vous souhaitez aborder la notion de données et l'importance que les données ont pris dans notre société avec vos élèves de SNT en seconde.

1. Citer un exemple que vous pourriez utiliser devant les élèves pour illustrer l'importance de l'exploitation des données dans leur vie quotidienne. Vous justifierez brièvement votre choix.
2. Expliquer, en quelques lignes et comme vous l'expliqueriez à vos élèves, pourquoi le stockage de données sur le « cloud » (donc dans les datacenters) a bien un impact négatif sur l'environnement. Vous donnerez deux impacts différents sur l'environnement.
3. Donner deux conseils concrets que vous pourriez donner à vos élèves pour limiter facilement l'impact de leurs données sur l'environnement.
4. Expliquer en quoi disposer de données massives (appelées aussi Big Data) peut être un avantage dans certains enjeux de société. Illustrer votre réponse avec un exemple que vous pourriez proposer à vos élèves.
5. Citer un danger (autre que l'impact sur l'environnement) qui est apparu ou peut apparaître avec les données massives. Vous expliquerez, comme vous l'expliqueriez à vos élèves, ce danger en quelques lignes.

2 Utilisation des fichiers csv

Vous voulez maintenant traiter la notion de données structurées toujours avec vos élèves de seconde.

6. Définir la notion de descripteur d'objet.
7. Proposer une activité débranchée que vous pourriez réaliser en classe pour leur faire comprendre la notion de descripteur d'objet. Cette activité devra durer entre 30 min et 1h. On attend une description détaillée du déroulement de l'activité et de ses attendus.

Vous abordez ensuite le traitement de ces données structurées. Voici l'extrait du programme de seconde SNT sur ce point :

Contenus	Capacités attendues
Traitement de données structurées	Réaliser des opérations de recherche, filtre, tri ou calcul sur une ou plusieurs tables

Pour cela, vous décidez de les faire travailler sur le fichier csv suivant : « Donnees_capteurs_22.09.21.csv » (cf Annexe A). Ce fichier contient toutes les mesures réalisées par les capteurs d'une maison domotisée le 22/09/2021. Chaque capteur envoie ses mesures toutes les 5 minutes. Ce sont des capteurs polyvalents placés sur les ouvertures d'une maison (fenêtres/portes) qui permettent de détecter l'ouverture ou non mais aussi le mouvement et la température. Les différentes colonnes disponibles dans ce fichier sont :

- Identifiant_capteur (entier) : identifiant unique du capteur.
- Piece (texte) : pièce dans laquelle se trouve le capteur.
- Num_mesure (entier) : numéro de la mesure de la journée pour le capteur (1 à minuit et s'incrémentera à chaque mesure jusqu'à la fin de la journée).

- Ouverture (booléen) : vaut True si la porte/fenêtre est ouverte et False sinon.
- Presence (booléen) : vaut True si le capteur détecte un mouvement et False sinon.
- Temperature (flottant) : indique la température mesurée à 0.1°C près.

Le fichier csv est pour l'instant ouvert par les élèves dans un tableur comme Calc/Excel.

8. Donner 3 questions que vous pourriez poser aux élèves pour les faire travailler, sur ordinateur, sur le traitement de données structurées, le but étant ici de leur faire découvrir les traitements possibles. Vous expliquerez à chaque fois brièvement l'intérêt de la question pour l'acquisition de compétence des élèves. Chaque question devra faire travailler les élèves sur une opération différente.

Vous souhaitez maintenant les faire travailler sur le traitement du fichier à l'aide de Python en utilisant la bibliothèque pandas. Vous trouverez en annexe une description des fonctions utiles de pandas (cf Annexe B).

Pour une exploitation du fichier « Donnees_capteurs_22_09_21.csv » avec la bibliothèque pandas, on s'est assuré que les nombres réels sont désormais écrits avec des points et non des virgules.

Le but de votre exercice est que les élèves puissent répondre de manière automatisée grâce à Python aux questions suivantes (pour le jour considéré) :

- Donner la température minimale de la cuisine quand le capteur 3 indique la présence de quelqu'un.
 - Afficher toutes les températures de la maison triées de manière croissante.
9. Écrire, tel qu'un élève pourrait le faire, le programme permettant de répondre à ces questions à l'aide de la bibliothèque pandas.
 10. Décrire la structure d'une activité de 1h30 sur machine permettant à vos élèves, qui n'ont jamais utilisé pandas, d'être capables de répondre à ces questions. On ne demande pas ici de rédiger complètement le document qui serait fourni aux élèves mais d'indiquer clairement l'organisation de l'activité en précisant notamment les questions posées aux élèves.

3 Traitement de données sous forme de tables

Vous travaillez sur le traitement de données en tables avec vos élèves de première. Pour travailler sur les données en tables, vous avez choisi de travailler avec des tableaux de p-uplets. Avec les élèves, les p-uplets seront codés, en Python, à l'aide de dictionnaires. Dans cet objectif, vous préparez un cours sur les dictionnaires.

11. Présenter, tel que vous le feriez avec vos élèves, la différence entre le type dictionnaire et le type liste.
12. Expliquer brièvement comment les dictionnaires sont implémentés en Python.
13. Citer un exemple d'utilisation de dictionnaire pour illustrer votre cours sur les dictionnaires. Vous justifierez votre choix en expliquant notamment pourquoi l'exemple est pertinent pour ce cours de première. Qu'est-ce qui pourrait être un mauvais exemple d'un point de vue pédagogique et pourquoi ?

Vous décidez de travailler avec vos élèves à partir de deux fichiers csv présentés en annexe pour démarrer le traitement des données en tables (annexe C). Pour que les élèves comprennent bien les mécanismes en jeu, vous n'utilisez pas ici la bibliothèque Python pandas.

Vous vous êtes inspirés de la situation sanitaire actuelle pour créer deux fichiers (qui ne sont bien-sûr que des fichiers exemples) : l'un correspondant à la liste des personnes testées positives à la Covid19 et l'autre à la liste des personnes ayant mangé au restaurant. Ces fichiers sont supposés être stockés par l'assurance maladie, le second servant à contacter les personnes contacts à risques.

Le premier fichier nommé « positifs.csv » (cf Annexe C.1) contient les informations suivantes :

- le numéro de sécurité social du contaminé (descripteur : num_SS ; type : entier),
- son âge (descripteur : age ; type : entier),
- le jour du test positif (descripteur : jour_test ; type : entier),
- le mois du test positif (descripteur : mois_test ; type : entier),
- son sexe (descripteur : sexe ; type : caractère),
- s'il a été vacciné ou non (descripteur : vaccine ; type : booléen).

Le deuxième fichier nommé « restaurants.csv » (cf Annexe C.2) contient les informations suivantes :

- l'identifiant de l'enregistrement (descripteur : id_enregistrement ; type : entier),
- l'identifiant du restaurant (descripteur : id_restaurant ; type : entier),
- le jour du repas (descripteur : jour_repas ; type : entier),
- le mois du repas (descripteur : mois_repas ; type : entier),

- la tranche horaire du repas (descripteur : tranche_horaire ; type : entier),
- le numéro de la table (descripteur : num_table ; type : entier),
- le numéro de sécurité social du client (descripteur : num_SS ; type : entier).

Pour cet exercice, on ne considèrera que 2 tranches horaires (la 1 de 12h à 14h et la 2 de 19h à 21h) et on considérera qu'une même personne ne peut pas être testée deux fois positive à la Covid19.

14. Écrire le programme Python commenté qui permettrait à vos élèves de récupérer sous forme de liste de dictionnaires, nommée **positifs**, les informations contenues dans le fichier « positifs.csv ». Chaque élément de la liste est un dictionnaire comprenant les valeurs d'une ligne de la table et dont les clés sont les descripteurs de la table. On utilisera pour cela notamment la fonction `readlines`.
15. Quels sont pour vous les points importants à aborder avec vos élèves et/ou ceux qui risquent de leur poser problème lorsque vous abordez pour la première fois le programme de la question 14 avec vos élèves ?

Vous souhaitez faire travailler vos élèves sur la recherche, dans une table, des lignes vérifiant des critères exprimés en logique propositionnelle en utilisant les fichiers décrits ci-dessus. On supposera, pour la suite de l'exercice, que les élèves ont implémenté le programme Python retournant une liste de dictionnaires **restaurants** contenant les informations contenues dans le fichier « restaurants.csv ».

16. Vous rédigez maintenant une partie de cet exercice.
 - (a) Proposer 3 questions de difficulté croissante (dont au moins une nécessitant la fusion des tables) où les élèves doivent rédiger des programmes en Python. On suppose ici que les listes de dictionnaires **positifs** et **restaurants** sont déjà construites.
 - (b) Proposer pour chaque question une correction. Décrire brièvement les points sur lesquels vous serez vigilant lors de la correction de ces questions.

4 Utilisation des bases de données

Dans cette partie, vous travaillez sur les bases de données avec vos élèves de terminale.

17. Expliquer tel que vous le feriez avec vos élèves l'intérêt des bases de données par rapport à ce que les élèves ont fait en première (avec les tables sous forme de fichiers csv).

Voici un extrait du programme de NSI de terminale.

Contenus	Capacités attendues	Commentaires
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.

Vous avez choisi, pour travailler sur le langage SQL, un exercice avec une base de données simplifiée qui stocke les données d'un service proposant des films en streaming. La base de données contient les tables suivantes :

Films = {Id, Titre, Année, Pays}
 Clients = {Id, Nom, Prénom, AdresseMail, TypeAbonnement}
 Visionnage = {Id, IdClient, IdFilm, Date, Heure, TempsVisionnage}
 Facturation = {Id, IdClient, Montant, Date, Statut}

Un extrait de cette base de données est fournie dans l'annexe D. Le temps de visionnage est stocké en minutes. Il existe deux types d'abonnement : le "basic" et le "premium". Les factures peuvent être "Payée" ou "En attente".

18. Voici l'exercice que vous avez donné aux élèves pour travailler sur les requêtes en SQL :
 - (a) Écrire la requête SQL permettant d'ajouter le film suivant à la base de données : « Wicked » un film réalisé aux USA et sorti en 2021. Son identifiant est 20356.
 - (b) Écrire la requête SQL permettant d'indiquer que la facture d'identifiant 2564 a été payée.
 - (c) Écrire la requête SQL permettant d'obtenir la liste des identifiants de facturation qui sont en attente et qui ont un montant strictement supérieur à 5 euros.

- (d) Écrire la requête SQL permettant d'obtenir la liste, par ordre alphabétique, des pays d'origine des films de la plateforme de l'année 2015. Chaque pays ne sera bien sûr indiqué qu'une seule fois.
- (e) Écrire la requête SQL permettant d'obtenir la liste des noms, prénoms et adresses mails des clients ayant une facturation en attente.
- (f) Écrire la requête permettant de déterminer le temps moyen de visionnage du film Titanic.
- (g) Écrire la requête permettant de donner la liste des identifiants des clients qui ont regardé Titanic plus longtemps que la moyenne des utilisateurs.

Donner les réponses que vous pourriez attendre pour chacune de ces questions.

19. Des élèves peinent à rédiger la requête d'interrogation de données de la question 18(e). Quelle aide leur proposez-vous ?
20. Donner un barème pour la correction de la question 18(g) Vous détaillerez les différentes compétences qu'il vous semble important d'évaluer.
21. Proposer, à l'aide de cette base de données, 3 questions qui vous permettraient de valider les compétences de vos élèves sur les requêtes SQL lors d'une interrogation. Vous justifieriez, pour chaque question, en quoi elle est pédagogiquement intéressante en indiquant notamment la (ou les) compétence(s) évaluée(s).

Problème 2 - Algorithmes de tri

Trier une collection de données avant de résoudre un problème algorithmique permet souvent d'élaborer des solutions beaucoup plus efficaces. Par exemple, on peut effectuer une recherche dichotomique au lieu d'effectuer un parcours linéaire dans une collection triée ou le tri d'un nuage de points permet de calculer efficacement l'enveloppe convexe de ce nuage de points (parcours de Graham).

De nombreux algorithmes de tri procèdent par comparaisons successives d'éléments entre eux. Pour évaluer l'efficacité de ces algorithmes et les comparer, on évalue souvent le nombre de comparaisons effectuées lors de leur exécution : on parle alors de **complexité en nombre de comparaisons**. Si l'on considère d'autres opérations élémentaires telles que les affectations, on dit que l'on évalue la **complexité temporelle**. Dans ce sujet, en réponse aux questions de complexité, on attend des ordres de grandeur en notation Landau (par exemple $O(n \log n)$ ou $O(n^2)$ si n est le nombre d'éléments à trier). Certains algorithmes de tri peuvent avoir des ordres de grandeur différents pour la complexité en nombre de comparaisons et la complexité temporelle. Nous garderons cela à l'esprit lors de l'analyse des algorithmes de tri de ce sujet. Enfin, on rappelle qu'un algorithme de tri par comparaisons correct nécessite au moins $n \log n$ comparaisons dans le pire cas. Si nécessaire, vous pourrez vous référer à cette borne considérée comme admise par la suite.

La dernière section de cette partie concerne des algorithmes de tri qui n'utilisent pas de comparaison, nous évaluerons alors seulement la complexité temporelle.

5 Tris naïfs

Dans cette section, deux algorithmes de tri par comparaisons du programme de première NSI sont considérés : le tri par insertion et le tri par sélection. Ces tris sont dits naïfs car leur complexité n'est pas optimale.

5.1 Tri par insertion

22. Quatre élèves de première NSI vous ont rendu leur implémentation du tri par insertion. Pour chaque production, rédiger une appréciation concise, donnant les erreurs éventuelles et les améliorations possibles afin que les élèves puissent se corriger par elles-mêmes.

<pre>def tri_eleve1(t) : n = len(t) for i in range(n) : for k in range(i, 1, -1) : if t[k] < t[k-1] : t[k] = t[k-1] t[k-1] = t[k]</pre>	<pre>def insere_eleve2(t, k) : if t[k] < t[k-1] : t[k], t[k-1] = t[k-1], t[k] insere_eleve2(t, k-1) def tri_eleve2(t, i=0) : insere_eleve2(t, i) tri_eleve2(t, i+1)</pre>
<pre>def tri_eleve3(t) : n = len(t) def insere_eleve3(i) : k = i while t[k] < t[k-1] : t[k], t[k-1] = t[k-1], t[k] k = k-1 for i in range(n) : t = insere_eleve3(i)</pre>	<pre>def tri_eleve4(t) : n = len(t) i = 0 while i < n : k = i while k > 0 and t[k] < t[k-1] : t[k], t[k-1] = t[k-1], t[k] i = i + 1</pre>

23. Pour que chacune puisse apprendre des erreurs des autres, rédiger une synthèse succincte à destination des élèves de première NSI dont vous venez de corriger les productions. L'idée est de s'appuyer sur les erreurs observées pour faire des rappels généraux sur les points clés à vérifier dans un programme afin qu'il fonctionne correctement et efficacement.
24. Une élève vous demande pourquoi on n'utilise pas un algorithme de dichotomie pour améliorer la complexité de l'insertion des éléments. Que lui répondez-vous ?

5.2 Tri par sélection

25. On rappelle succinctement le fonctionnement du tri par sélection au travers du pseudo-code suivant :

```
def tri_sel(t) :
    pour k de 0 à n-1
        sélectionner l'indice i0 du minimum des éléments entre les indices de k à n-1
        échanger t[k] et t[i0]
```

Écrire l'énoncé d'un devoir maison permettant de faire étudier cet algorithme à vos élèves de première NSI. Cet énoncé devra les guider pour :

- découvrir cet algorithme,
- l'implémenter,
- prouver sa correction,
- estimer sa complexité en nombre de comparaisons.

Attention, le contenu de cet énoncé devra se suffire à lui-même, et ne dépendre d'aucune ressource externe ou interventions spécifiques de votre part.

26. À la fin de la séance, une élève vous demande pourquoi on n'utilise pas un algorithme de dichotomie pour la sélection des éléments. Que lui répondez-vous ?

6 Tris par fusion

Cette section s'articule autour du tri partition-fusion du programme de terminale NSI et d'une version simplifiée du tri *Timsort* qui agit également par fusions successives.

6.1 Tri partition-fusion

Lors d'un devoir sur table en classe de terminale NSI, vous avez demandé à vos élèves d'implémenter le tri partition-fusion sans plus de précision. Trois productions d'élèves sont reproduites ci-dessous.

Élève 1 :

```
def fus1(t1,t2):
    n1,n2 = len(t1),len(t2)
    if n1 = 0 : return t2
    if n2 = 0 : return t1
    if t1[0]<t2[0] :
        return [t1[0]]+fus1(t1[1:],t2)
    else :
        return [t2[0]]+fus1(t1,t2[1:])

def tri_f1(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        return fus1(tri_f1(t[:m]),tri_f1(t[m:]))
```

Élève 2 :

```
def fus2(t1,t2):
    n1,n2 = len(t1),len(t2)
    while i1 < n1 and i2 < n2 :
        if t1[i1]<t2[i2] :
            res.append(t1[i1])
            i1 = i1 + 1
        else :
            res.append(t2[i2])
            i2 = i2 + 1
    if i1 == n1 : res.extend(t2[i2:])
    else : res.extend(t1[i1:])
    return res

def tri_f2(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        t1,t2 = t[:m],t[m:]
        tri_f2(t1)
        tri_f2(t2)
        return fus2(t1,t2)
```

Élève 3 :

```
def fus3(t,g,m,d):
    aux = []
    i1,i2 = g,m
    while i1<m and i2<d :
        if t[i1]<t[i2] :
            aux.append(t[i1])
            i1 = i1 + 1
        else :
            aux.append(t[i2])
            i2 = i2 + 1
    if i1<m : aux.extend(t[i1:m])
    for i in range(len(aux)):
        t[i+g] = aux[i]

def tri_f3(t):
    def aux(g,d):
        if g < d-1 :
            m = (g+d)//2
            aux(g,m)
            aux(m,d)
            fus3(t,g,m,d)
        aux(0,len(t))
```

27. Élaborer un barème détaillé pour cette question.

28. Corriger ces trois productions, c'est-à-dire donner la note détaillée selon le barème et l'appréciation associée pour chacune des productions d'élèves. **Les corrections seront données directement sur les productions d'élèves fournies dans le document réponse 1 (Annexe E). Ce document réponse est à rendre avec votre copie.**

6.2 Tri *Timsort* simplifié

On présente ci-dessous une implémentation d'un tri *Timsort* simplifié.

```
def f1(l,a):
    i = a
    while i<len(l)-1 and l[i]<=l[i+1] :
        i += 1
    return i+1

def f2(l):
    i = 0
    res = [0]
    while i<len(l) :
        i = f1(l,i)
        res.append(i)
    return res

def f3(t, debut, milieu, fin) :
    i = debut
    j = milieu
    lt = []
    while i < milieu and j < fin :
        if t[i] < t[j] :
            lt.append(t[i])
            i += 1
        else :
            lt.append(t[j])
            j += 1
    for k in range (i, milieu) : lt.append(t[k])
    for k in range (debut, j) : t[k] = lt[k-debut]

def f4(t,rc):
    nrc = []
    i = 0
    n = len(rc)
    while i < n :
        nrc.append(rc[i])
        if i+2 < n :
            f3 (t, rc[i], rc[i+1], rc[i+2])
            i += 2
        else : i += 1
    return nrc

def f5(t):
    rc = f2(t)
    while len(rc) > 2 :
        rc = f4 (t, rc)
```

29. (a) Proposer des noms explicites pour ces fonctions.
 (b) Présenter le fonctionnement général de ces fonctions tel que vous le feriez à vos élèves.
 (c) Illustrer le principe de ce tri sur un ou plusieurs exemples pour que les élèves puissent l'implémenter par elles-mêmes.
30. Quelle est la complexité, dans le pire cas, en nombre de comparaisons de cet algorithme de tri ?

7 Tris sans comparaison

Votre classe de terminale NSI a bien compris les différents algorithmes du tri du programme. Vous décidez donc d'aller plus loin en leur faisant découvrir d'autres algorithmes de tri qui fonctionnent sans comparaison sur des listes d'entiers positifs.

7.1 Tri par dénombrement

Au lieu de comparer les éléments entre eux, le tri par dénombrement compte le nombre d'occurrences de chaque élément. Une fois les nombres d'occurrences obtenus, la liste est reconstituée dans l'ordre croissant en y plaçant tous les éléments présents dans l'ordre croissant avec le bon nombre d'occurrences.

Par exemple, pour trier $[1, 7, 5, 3, 7, 7, 1, 3, 3, 7, 5]$, on compte qu'il y a :

- 2 occurrences de 1
- 3 occurrences de 3
- 2 occurrences de 5
- 4 occurrences de 7

On peut alors reconstituer la liste triée : $[1, 1, 3, 3, 3, 5, 5, 7, 7, 7, 7]$.

Vous avez posé la question suivante à vos élèves :

Écrire la fonction Python tri_den qui prend en paramètre une liste t d'entiers positifs, ne renvoie aucune valeur mais y applique le tri par dénombrement. La fonction proposée aura une complexité temporelle en $O(n + vmax)$ où n est le nombre d'éléments dans t et vmax est la valeur maximale dans t.

Une élève vous rend le travail suivant :

```
def compte_occu(t,elt):
    c = 0
    for x in t :
        if x == elt : c = c+1
    return c

def tri_den_el(t):
    occ = [0]*10000
    res = []
    for elt in range(10000):
        occ[elt] = compte_occu(t,elt)
        res = res + [elt]*occ[elt]
    return res
```

31. Lister les indications à lui donner pour qu'elle puisse répondre correctement à votre question.
32. Écrire, sans commenter, le programme que vous espérez qu'elle écrive.
33. La complexité temporelle de cet algorithme contredit-elle la borne de complexité admise en introduction de ce sujet ? Pourquoi ?
34. Cet algorithme de tri est-il toujours plus efficace, en termes de complexité temporelle, que le tri fusion dont la complexité temporelle est en $O(n \log n)$? Justifier.

7.2 Tri par baquets

Dans cette partie, le tri par baquets est étudié, le principe est le suivant :

- on travaille avec des baquets numérotés de 0 à 9 (ces baquets sont implémentés par une liste de 10 listes) ;
- pour débuter, chaque élément de la liste initiale est placé dans le baquet numéroté par son chiffre des unités ;
- à chaque étape, les éléments sont parcourus baquet par baquet en partant de l'indice 0 du baquet 0 et rangés à nouveau dans les baquets en utilisant le chiffre suivant ;
- l'algorithme est terminé lorsque tous les éléments sont dans le baquet 0.

Par exemple : si

$t = [8295, 7971, 8806, 1203, 2916, 6231, 1112, 6131, 538, 3832, 1813, 6863, 6200, 9449, 1905, 1749]$

- au départ, les éléments sont rangés dans les baquets par chiffres des unités :

$[[6200], [7971, 6231, 6131], [1112, 3832], [1203, 1813, 6863], [], [8295, 1905], [8806, 2916], [], [538], [9449, 1749]]$

- puis on les place par chiffre des dizaines en préservant l'ordre sur les unités dans chaque baquet grâce à l'ordre de parcours des éléments :

```
[[6200, 1203, 1905, 8806], [1112, 1813, 2916], [], [6231, 6131, 3832, 538], [9449, 1749], [], [6863], [7971], [], [8295]]
```

- puis par chiffre des centaines en respectant l'ordre précédent dans chaque baquet :

```
[], [1112, 6131], [6200, 1203, 6231, 8295], [], [9449], [538], [], [1749], [8806, 1813, 3832, 6863], [1905, 2916, 7971]]
```

- puis par chiffre des milliers :

```
[[538], [1112, 1203, 1749, 1813, 1905], [2916], [3832], [], [], [6131, 6200, 6231, 6863], [7971], [8295, 8806], [9449]]
```

- à la dernière étape, tous les éléments se retrouvent dans le baquet 0 :

```
[[538, 1112, 1203, 1749, 1813, 1905, 2916, 3832, 6131, 6200, 6231, 6863, 7971, 8295, 8806, 9449], [], [], [], [], [], [], []]
```

35. Écrire une fonction Python `ch` qui prend en paramètre deux entiers et telle que l'appel `ch(k, i)` renvoie le `i`-ème chiffre de l'entier `k`. Par exemple, `ch(1345, 2) = 4`, `ch(1345, 4) = 1` et `ch(1345, 5) = 0`. Cette fonction pourra être utilisée par vos élèves pour l'implémentation du tri par baquets.
36. Vous avez fourni une implémentation à trous de l'algorithme de tri par baquets à vos élèves. En particulier, la fonction Python `tri_baquets` prend en paramètre une liste `t`, applique le tri par baquets à `t` et renvoie une liste triée contenant les mêmes éléments que `t`.

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[.....].append(.....)

def vider(bacs):
    for i in range(10):
        bacs[i] = .....

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[.....].append(.....)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = .....
        vider(.....)
        i += 1
    return .....
```

Compléter ce que vous attendez que les élèves écrivent dans les trous sans commenter. **Vous compléterez cet algorithme à trous fourni dans le document réponse 2 (Annexe F). Ce document réponse est à rendre avec votre copie.**

37. Quelle est la complexité temporelle de ce tri ?
38. Dans quels cas utiliser cet algorithme plutôt que le tri par dénombrement ?
39. Cet algorithme de tri est-il toujours plus efficace, en termes de complexité temporelle, que le tri fusion dont la complexité temporelle est en $O(n \log n)$?

Annexes

A Extrait du fichier Donnees_capteurs_22_09_21.csv (ouvert dans un tableau)

Le séparateur du fichier .csv est ";".

	A	B	C	D	E	F
1	Identifiant_capteur	Piece	Num_mesure	Ouverture	Presence	Temperature
2		1 Salon		1 False	False	18,7
3		3 Cuisine		1 False	False	18,8
4		2 Chambre 1		1 True	False	17,5
5		5 Salle de bain		1 False	False	19,1
6		6 Chambre 2		1 False	True	16,6
7		4 Entrée		1 False	False	17,4
8		1 Salon		2 False	False	18,7
9		3 Cuisine		2 False	False	18,8
10		2 Chambre 1		2 True	False	17,5

B Description de fonctions utiles de la bibliothèque pandas

- pandas.read_csv(nom,sep=...) prend en argument une chaîne contenant l'adresse du fichier csv à lire et un chaîne contenant le séparateur du fichier csv, et retourne une instance de type dataframe, initialisée avec l'ensemble des données présentes dans le fichier. Lors de la création du dataframe, un index est associé à chacune des lignes du fichier (en commençant à partir de 0).

Par la suite, pour illustrer l'utilisation de certaines fonctions, on suppose que la variable "df" pointe vers le dataframe obtenu à partir du fichier Donnees_capteurs_22_09_21.csv.

- df.loc(index_ligne,index_colonne) retourne, sous forme d'un objet de type dataframe, certaines lignes et colonnes du dataframe df. L'index des colonnes est le nom des colonnes dans le fichier csv, et l'index des lignes est l'index créé lors de l'initialisation du dataframe à la lecture du fichier.

Par exemple, print(df.loc[0,'Temperature']) affichera la température de la 1ère ligne du fichier.

Il est également possible d'effectuer des tests sur le contenu d'un dataframe.

Par exemple, print(df['Piece']=='Salon') affichera, sous forme de booléens, pour chaque ligne du fichier, si la pièce mentionnée est un salon.

Ces deux techniques peuvent être conjointement utilisées pour filtrer des lignes à partir de tests.

Par exemple, print(df.loc[df['Piece']=='Salon', :]) affichera toutes les lignes qui concernent le salon.

Il est possible de combiner plusieurs facteurs de filtrage en utilisant un "et"(&) ou un "ou"(|).

Par exemple, print(df.loc[(df['Piece']=='Salon')|(df['Piece']=='Cuisine'),'Presence']) affichera la colonne présence pour les lignes correspondant au salon ou à la cuisine.

- Il est possible de retourner le maximum (méthode max), le minimum (méthode min) et la moyenne (méthode mean) d'une donnée d'une dataframe.

Par exemple, print(df.loc[:, 'Temperature'].mean()) affichera la température moyenne mesurée sur l'ensemble des lignes du fichier.

- Pour retourner le contenu trié d'un dataframe, on utilise la méthode dataframe.sort_values(column, ascending=...). column est une chaîne indiquant la colonne utilisée pour le tri. Si ascending vaut True (valeur par défaut), le tri est croissant, s'il vaut False, il est décroissant.

Par exemple, df.sort_values(by=['Piece'], ascending=False) affiche les lignes du fichier triées par Piece dans l'ordre alphabétique inversé (en commençant par Z).

C Extraits des fichiers pour le traitement des données sous forme de table

C.1 Extrait du fichier positifs.csv

```
num_SS;age;jour_test;mois_test;sexe;vaccine
1630580265006;58;2;7;H;True
2750846580156;46;2;7;F;False
1950412410023;26;3;7;H;False
2570235820132;64;3;7;F;False
```

C.2 Extrait du fichier restaurants.csv

```
id_enregistrement;id_restaurant;jour_repas;mois_repas;tranche_horaire;num_table;num_SS
1;324;30;6;2;5;1850659240032
2;324;30;6;2;5;2860372541115
3;796;1;7;1;8;1950412410023
4;796;1;7;1;8;1951012410056
5;796;1;7;1;8;1951232564031
```

D Extrait de la base de données

Films			
Id	Titre	Annee	Pays
12546	Entre les murs	2008	France
2506	Fight Club	1999	USA
67895	La cité de la peur	1994	France
54781	Yes Day	2021	USA

Clients				
Id	Nom	Prenom	AdresseMail	TypeAbonnement
1761	Dupont	Martin	dm@capes.fr	Basic
564	Tartanpion	Lucie	TLucie@mail.com	Premium
153	Durant	Hervé	RVDurant@capes.fr	Premium
789	Martin	Nina	Nina@pro.fr	Basic

Visionnage					
Id	IdClient	IdFilm	Date	Heure	TempsVisionnage
123456	1761	5468	11/08/2021	10 :45	35
93720	2456	14803	09/11/2020	21 :10	124
115986	1602	14972	27/03/2021	08 :30	2
78439	2395	6730	16/05/2020	22 :35	48

Facturation				
Id	IdClient	Montant	Date	Statut
2761	1254	7.90	12/07/2021	Payee
1065	1542	7.90	05/08/2020	Payee
1753	864	11.90	03/04/2021	En attente
1334	741	11.90	09/10/2020	Payee

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

► Concours externe du CAPES de l'enseignement public :

Concours
EBE

Section/option
6200E

Epreuve
102

Matière
9312

► Concours externe du CAFEP/CAPES de l'enseignement privé :

Concours
EBF

Section/option
6200E

Epreuve
102

Matière
9312

Modèle CMEN-DOC v2 ©NEOPTEC	
Nom de famille : <small>(Suivi, s'il y a lieu, du nom d'usage)</small>	<input style="width: 100%; height: 20px; border: none; border-bottom: 1px solid black;" type="text" value=""/>
	Prénom(s) : <input style="width: 100%; height: 20px; border: none; border-bottom: 1px solid black;" type="text" value=""/>
	Numéro Inscription : <input style="width: 100%; height: 20px; border: none; border-bottom: 1px solid black;" type="text" value=""/>
<small>(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)</small>	
<small>(Remplir cette partie à l'aide de la notice)</small>	
Concours / Examen :	Section/Specialité/Série :
Epreuve :	Matière :
Session :	
CONSIGNES <ul style="list-style-type: none"> • Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES. • Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance. • Numérotter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre. • Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire. • N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon. 	

EBE NSI 2

DR1

**Tous les documents réponses sont à rendre,
même non complétés.**

Tournez la page S.V.P.

B

NE RIEN Ecrire DANS CE CADRE

E Document Réponse 1 : Correction de trois productions d'élèves

Document réponse de la question 28 de la section 6.1 (Problème 2). **Ce document réponse doit être rendu avec votre copie.**

Élève 1 :

```
def fus1(t1,t2):
    n1,n2 = len(t1),len(t2)
    if n1 = 0 : return t2
    if n2 = 0 : return t1
    if t1[0]<t2[0] :
        return [t1[0]]+fus1(t1[1:],t2)
    else :
        return [t2[0]]+fus1(t1,t2[1:])

def tri_f1(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        return fus1(tri_f1(t[:m]),tri_f1(t[m:]))
```

Élève 2 :

```
def fus2(t1,t2):
    n1,n2 = len(t1),len(t2)
    while i1 < n1 and i2 < n2 :
        if t1[i1]<t2[i2] :
            res.append(t1[i1])
            i1 = i1 + 1
        else :
            res.append(t2[i2])
            i2 = i2 + 1
    if i1 == n1 : res.extend(t2[i2:])
    else : res.extend(t1[i1:])
    return res

def tri_f2(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        t1,t2 = t[:m],t[m:]
        tri_f2(t1)
        tri_f2(t2)
        return fus2(t1,t2)
```

Élève 3 :

```
def fus3(t,g,m,d):
    aux = []
    i1,i2 = g,m
    while i1<m and i2<d :
        if t[i1]<t[i2] :
            aux.append(t[i1])
            i1 = i1 + 1
        else :
            aux.append(t[i2])
            i2 = i2 + 1
    if i1<m : aux.extend(t[i1:m])
    for i in range(len(aux)):
        t[i+g] = aux[i]

def tri_f3(t):
    def aux(g,d):
        if g < d-1 :
            m = (g+d)//2
            aux(g,m)
            aux(m,d)
            fus3(t,g,m,d)
        aux(0,len(t))
```


Modèle CMEN-DOC v2 ©NEOPTEC	
Nom de famille : <small>(Suivi, s'il y a lieu, du nom d'usage)</small>	<input type="text"/>
Prénom(s) :	<input type="text"/>
Numéro Inscription :	<input type="text"/> Né(e) le : <input type="text"/> / <input type="text"/> / <input type="text"/>
<i>(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)</i>	
(Remplir cette partie à l'aide de la notice)	
Concours / Examen :	Section/Specialité/Série :
Epreuve :	Matière :
	Session :
CONSIGNES <ul style="list-style-type: none"> • Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES. • Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance. • Numérotter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre. • Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire. • N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon. 	

EBE NSI 2

DR2

**Tous les documents réponses sont à rendre,
même non complétés.**

Tournez la page S.V.P.

C

NE RIEN Ecrire DANS CE CADRE

F Document Réponse 2 : Algorithme de tri par baquets à trous à compléter

Document réponse de la question 36 de la section 7.2. Ce document réponse doit être rendu avec votre copie.

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[.....].append(.....)

def vider(bacs):
    for i in range(10):
        bacs[i] = .....

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[.....].append(.....)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = .....
        vider(.....)
        i += 1
    return .....
```




EAE INF 1

SESSION 2022

AGREGATION CONCOURS EXTERNE

Section : INFORMATIQUE

COMPOSITION EN INFORMATIQUE

Durée : 5 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique (y compris la calculatrice) est rigoureusement interdit.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

A

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours

EAE

Section/option

6200A

Epreuve

101

Matière

9422

Dépendances. Ce sujet contient quatre parties indépendantes qui doivent être traitées toutes les quatre. On veillera à bien indiquer sur la copie les changements de partie.

Attendus. Il est attendu des candidates et des candidats des réponses construites. Ils seront aussi évalués sur la précision, le soin et la clarté de la rédaction.

Partie 1 : Système

Exercice 1 Droits et permissions d'accès aux fichiers.

Ci-dessous, le résultat de la commande linux `ls -al` en ligne de commande sur une installation linux standard :

```
[untel:~/Ag] ls -al
total 36
drwxrwxr-x 3 untel guntel 4096 nov. 7 18:10 .
drwxr-xr-x 75 untel guntel 20480 nov. 7 20:04 ..
lrwxrwxrwx 1 untel guntel     6 nov. 7 18:08 fc.txt -> fi.txt
-rw-rw-r-- 1 untel guntel   153 nov. 7 18:10 fi.txt
-rwxrw-r-- 1 untel otel      83 nov. 7 18:10 oui.sh
drwxrwxr-- 2 untel guntel 4096 nov. 7 18:08 sAg
```

- Quel est l'utilisateur qui utilise le terminal ?
- Quel est le propriétaire des fichiers ?
- Que sont `guntel` et `otel` ?
- Quel est le nom du répertoire courant ?
- Combien contient-il de fichiers ? de répertoires ?
- Pour `oui.sh` et `fc.txt`, expliquer les informations affichées. Des réponses précises sont demandées pour les colonnes 1, 3, 4 ainsi que la dernière.
- Quelles sont les opérations liées à `sAg` que peut faire un utilisateur dont l'identifiant est `tritel` (précisez les différentes possibilités) ? Par exemple, que se passe-t-il si, à partir du répertoire `Ag`, `tritel` tape la commande `cd sAg`.
- Pourquoi certains systèmes d'exploitation ont-ils une politique de gestion des droits et de permissions d'accès aux fichiers ?

Exercice 2 Système de Gestion de Fichiers.

On considère un système de gestion de fichiers `ext2` avec une taille de bloc de 2048 octets. Les caractéristiques d'un inode de cet `ext2` sont :

- les 12 (de 0 à 11) premiers champs pointent sur un bloc de données,
 - le champ 12 pointe vers 256 blocs de données (simple indirection),
 - le champ 13 pointe vers 256^2 blocs de données (double indirection),
 - le champ 14 pointe vers 256^3 blocs de données (triple indirection),
 - le champ longueur : 32 bits, mais le premier bit de poids fort n'est pas utilisé.
- Quelle est la taille maximum d'un fichier sur ce système de gestion de fichiers ?
 - Quel est le nombre maximum de blocs ?

B

- c. Soit un fichier contenant 4567 octets.
Combien de blocs occupe-t-il ?
- d. Même question pour un fichier de 573448 octets. Parmi ces blocs, combien y a-t-il de blocs directs, indirects, double indirects et triple indirects ?

Exercice 3 Performance

Voici un extrait d'article de Roberto Di Cosmo, intitulé *Piège dans le Cyberespace*, mis en ligne le 20 mars 1998.

"Imaginons maintenant un ministère qui garde ses dossiers dans une énorme armoire avec des millions de tiroirs : on aimerait bien, pour les mêmes raisons qu'avant, que les documents afférents à un même dossier se trouvent dans la mesure du possible rangés dans des tiroirs contigus. Vous devez embaucher une secrétaire et vous avez le choix entre deux candidates aux pratiques assez différentes : la première, quand un dossier est bouclé se limite à vider les tiroirs, et quand un nouveau dossier arrive elle le sépare en petits groupes de documents de la taille d'un tiroir, et range chaque groupe au hasard dans le premier tiroir vide qu'elle trouve dans l'armoire. Lorsque vous lui faites remarquer que ça risque alors d'être bien difficile de retrouver vite tous les documents du dossier du Crédit Lyonnais, elle répond qu'il faut engager tous les week-ends une dizaine de garçons pour tout remettre en ordre. La deuxième secrétaire, par contre, conserve sur son bureau une liste des tiroirs vides contigus, qu'elle met à jour toutes les fois qu'un dossier est clos et qu'on l'enlève des tiroirs ; quand un nouveau dossier arrive, elle cherche dans sa liste une suite de tiroirs vides contigus de taille suffisante et c'est là qu'elle place le nouveau dossier. Ainsi, vous explique-t-elle, s'il y a assez de mouvement, l'armoire restera toujours très bien rangée."

Expliquer par une description la plus précise possible et des exemples précis à quelle problématique et à quelles différentes technologies informatiques fait référence cet extrait d'article. Pour cela votre réponse devra au moins inclure des éléments concernant les questions suivantes :

- Que représente *l'énorme armoire du ministère* ?
- Que représentent les *dossiers à ranger dans cette armoire* ?
- Pourquoi *séparer un dossier en petits groupes de documents* ?
- Que représente la *dizaine de garçons qui remettent tout en ordre* ?
- Que représente la *liste des tiroirs vides mise à jour par la deuxième secrétaire* ?
- Quelle est la méthode la plus performante ?
- Si vous en connaissez, donnez des exemples concrets correspondants aux deux méthodes décrites.

Exercice 4 Processus.

- a. 1. Que fait le programme C suivant ? :

```
int main(void)
{
    for (;;)
        fork();
    return 0;
}
```

2. Est-ce un problème ? Si oui, donner une solution pour l'éviter.
- b. 1. Soit la fonction C qui calcule x^n de façon récursive en créant un fils qui va calculer $x^{n/2}$ et le reste par le père.

```

int calcul (int x, int n)
{
    int a, b, pid;
    int p = n / 2;
    int fds[2];
    if (n == 0) return 1;
    if (n == 1)
    {
        return x;
    }
    if (pipe (fds) == -1)
    {
        perror ("Erreur creation tube");
        exit (1);
    }
    pid = fork ();
    switch (pid)
    {
        case -1:
            perror ("Erreur creation processus fils");
            exit (1);
        case 0: // fils
            close (fds[0]);
            a = calcul (x, p);
            write (fds[1], &a, sizeof (int));
            exit (0);
        default: // pere
            close (fds[1]);
            read (fds[0], &a, sizeof (int));
            b = calcul (x, n - p);
            wait (NULL);
    }
    return (a * b);
}

```

Modifier la fonction en créant deux processus fils qui calculent respectivement $x^{n/2}$ et $x^{n-n/2}$ (la division étant entière), le père calculant le produit et ainsi de suite récursivement. On considère le cas où l'on fait le calcul dans deux fils dans tous les cas, même si n est pair.

Remarque et données : l'exactitude de la syntaxe C n'est pas le critère principal pour cette question. Voici les signatures des principales fonctions de gestion de processus utilisées ici :

```

pid_t fork(void);
int pipe(int pipefd[2]);
pid_t wait(int *status);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int close(int fd);

```

2. Que pouvez-vous dire de l'efficacité de ces deux fonctions calculant x^n ?

Partie 2 : Nombres flottants

Cet exercice s'intéresse aux nombres flottants et à leur représentation en mémoire. Un nombre flottant sur 64 bits est un vecteur de bits comme suit :



avec s le signe sur 1 bit ; E l'exposant est un entier en binaire non signé sur 11 bits ; f la fraction sur 52 bits.

Si $1 \leq E \leq 2^{11} - 2$, le nombre est dit normal et sa valeur est

$$(-1)^s \times 2^{E-1023} \times 1.f$$

où $1.f$ est un nombre en binaire à virgule fixe.

Si $E = 0$, le nombre est dit subnormal et sa valeur est

$$(-1)^s \times 2^{1-1023} \times 0.f$$

où $0.f$ est un nombre en binaire à virgule fixe.

D'autres valeurs spéciales sont représentées avec l'exposant maximal mais ne nous intéressent pas ici.

Question 1. Donner les valeurs réelles des nombres flottants dont les représentations sont :

Question 2. Donner la représentation binaire des valeurs réelles :

- a. 1
- b. -1
- c. 2^{-53}
- d. $1 + 2^{-50}$

Question 3. On considère un nombre flottant positif. On peut lire sa représentation mémoire comme étant celle d'un entier non signé 64 bits. Par exemple, $0 | 0 \cdots 0 | 0 \cdots 01$ serait interprété par 1 et $0 | 0 \cdots 01 | 0 \cdots 0$ par 2^{53} . On considère la fonction \mathcal{S} qui interprète un nombre flottant comme un entier non signé 64 bits, lui ajoute 1 et le réinterprète comme un nombre flottant. Soit x de signe s , d'exposant E et de fraction f .

- a. Supposons $s = 0$, $E = 0$ et $f \neq 1 \cdots 1$; que vaut $\mathcal{S}(x)$?
- b. Supposons $s = 0$, $1 \leq E \leq 2^{11} - 2$ et $f \neq 1 \cdots 1$; que vaut $\mathcal{S}(x)$?
- c. Supposons $s = 0$, $E = 0$ et $f = 1 \cdots 1$; que vaut $\mathcal{S}(x)$?
- d. Supposons $s = 0$, $0 \leq E \leq 2^{11} - 2$ et $f \neq 1 \cdots 1$; que vaut $\mathcal{S}(x)$?
- e. Supposons $s = 0$, $0 \leq E \leq 2^{11} - 2$ et $f = 1 \cdots 1$; que vaut $\mathcal{S}(x)$?
- f. Conclure sur \mathcal{S} .

Question 4. Dans la suite, on admet (sans le démontrer) que les flottants sont également l'ensemble des $n2^e$ avec n et e entiers, $|n| < 2^{53}$ et $-1074 \leq e \leq 970$.

- a. Prouver que 0 , 1 , $1/8$, 2^{52} et -2^{52} sont des flottants selon cette caractérisation.
- b. Donner le plus grand nombre flottant et le plus petit nombre flottant strictement positif.
- c. Donner tous les nombres flottants dans l'intervalle $[1 - 2^{-52}; 1 + 2^{-52}]$.

Dans la suite du problème, on s'intéresse aux effets de l'arrondi sur les opérations entre flottants. On note \mathbb{F} l'ensemble des flottants. On voit \mathbb{F} comme une partie de \mathbb{R} , l'ensemble des nombres réels. Ainsi, un flottant peut être vu, au besoin, comme un réel.

Il existe plusieurs modes d'arrondi. On considère ici l'arrondi par défaut, qui est l'arrondi au plus proche (en anglais *round to nearest*). Il s'agit de la fonction $\mathcal{N} : \mathbb{R} \rightarrow \mathbb{F}$ qui, pour un réel x , renvoie le nombre flottant le plus proche de x . En cas de point milieu (si x est à égale distance de deux nombres flottants), $\mathcal{N}(x)$ est celui qui a la fraction paire (fini par un zéro au sens de l'introduction de cet exercice).

La spécification des flottants garantit que l'addition sur \mathbb{F} , qu'on note \oplus , se comporte comme si le calcul avait d'abord été fait dans \mathbb{R} de façon exacte puis qu'on avait arrondi le résultat avec \mathcal{N} . On a donc la relation

$$\forall x \in \mathbb{F}, \forall y \in \mathbb{F}, \quad x \oplus y = \mathcal{N}(x + y).$$

On a des relations similaires pour la soustraction \ominus et la multiplication \otimes .

Question 5. Calculer $(-2^{52} \oplus 2^{52}) \oplus 1/8$. Calculer $-2^{52} \oplus (2^{52} \oplus 1/8)$. Conclure sur une propriété non respectée par \oplus .

Question 6. Trouver un nombre flottant x non nul tel que $x \otimes x = 0$.

Que penser donc du code Python suivant :

```
if (x != 0):
    z=1/(x*x)
```

Question 7.

- Trouver un nombre flottant x tel que $1 \oplus x = 1$.
- Trouver le plus grand nombre flottant x tel que $1 \oplus x = 1$. Justifier.

Question 8. Dans cette question, on considère l'arrondi vers $-\infty$, noté \mathcal{D} (en anglais *round down*) : pour tout réel x , $\mathcal{D}(x)$ est le plus grand nombre flottant inférieur ou égal à x .

- Trouver un nombre flottant x tel que $\mathcal{D}(1 + x) = 1$.
- Trouver le plus grand nombre flottant x tel que $\mathcal{D}(1 + x) = 1$. Justifier.

Question 9. Dans cette question, on considère l'arrondi vers $+\infty$, noté \mathcal{U} (en anglais *round up*) : pour tout réel x , $\mathcal{U}(x)$ est le plus petit nombre flottant supérieur ou égal à x .

- Trouver un nombre flottant x tel que $\mathcal{U}(1 + x) = 1$.
- Trouver le plus grand nombre flottant x tel que $\mathcal{U}(1 + x) = 1$. Justifier.

Question 10. Si le résultat d'une addition est suffisamment petit, alors il n'y a pas eu d'erreur d'arrondi. Prouver le lemme suivant : soient x et y deux nombres flottants, si $|x + y| \leq 2^{-1022}$ alors $x \oplus y = x + y$.

Question 11. Si l'on soustrait deux valeurs proches, mais résultats d'un calcul arrondi, on peut obtenir un résultat complètement faux. Comparer $(2^{52} \oplus 1/8) \ominus (2^{52} \ominus 1/8)$ et la valeur mathématique sans arrondi $(2^{52} + 1/8) - (2^{52} - 1/8)$.

Question 12. Mais ce n'est pas la dernière soustraction qui crée cette erreur, elle ne fait que mettre en lumière les erreurs précédentes. Prouver le lemme suivant : soient x et y deux nombres flottants, si $y/2 \leq x \leq 2y$, alors $x \ominus y = x - y$.

Partie 3 : Logique, déduction naturelle

Un rappel des règles de déduction naturelle se trouve en annexe de cette partie.

Exercice 1

Nous avons les faits suivants :

1. Si Informatix réussit sa preuve, il sera content.
2. S'il pleut, Informatix restera chez lui.
3. Si Informatix ne reste pas chez lui, alors il sera content.
4. Chez lui, Informatix s'entraîne.
5. Informatix réussira sa preuve s'il s'entraîne.

- a. Formaliser ces faits à l'aide de formules logiques propositionnelles.
- b. Montrer, en déduction naturelle, qu'Informatix sera content.

Exercice 2

- a. Montrer en déduction naturelle les deux séquents suivants. L'un des deux séquents est classique, dire lequel.
 1. $\Gamma \vdash (A \Rightarrow B) \Rightarrow (\neg A \vee B)$
 2. $\Gamma \vdash (\neg A \vee B) \Rightarrow (A \Rightarrow B)$
- b. Lorsque la formule à éliminer est une hypothèse, on donne les règles alternatives Hypothèses, en annexe.
En vous aidant des règles Hypothèses, montrer que : $\Gamma \vdash \exists x \neg P(x) \Rightarrow \neg(\forall x P(x))$
- c. A partir de la règle $\vdash A \vee \neg A$, est-il possible de montrer $\vdash \neg\neg A \Rightarrow A$?
Si oui, faire la démonstration, sinon expliquer pourquoi.
- d. Expliquer par un (contre-)exemple (intuitif) pourquoi $\forall x \exists y P(x, y)$ n'est pas la même chose que $\exists y \forall x P(x, y)$.

Annexe : règles de la déduction naturelle

Axiome	Classique	Coupage
$\frac{A \in \Gamma}{\Gamma \vdash A} Ax$	$\frac{}{\Gamma \vdash A \vee \neg A} EM$	$\frac{\Gamma, A \vdash B \quad \Gamma \vdash A}{\Gamma \vdash B} Cut$

	Introduction	Elimination
\perp		$\frac{\Gamma \vdash \perp}{\Gamma \vdash C} \perp e$
\neg	$\frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} \neg i$	$\frac{\Gamma \vdash \neg A \quad \Gamma \vdash C}{\Gamma \vdash C} \neg e$
\wedge	$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge i$	$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge eg \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge ed$
\vee	$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee ig \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee id$	$\frac{\Gamma, A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee e$
\Rightarrow	$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow i$	$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow e$
\forall	$\frac{\Gamma \vdash P \quad x \notin vl(\Gamma)}{\Gamma \vdash \forall x, P} \forall i$	$\frac{\Gamma \vdash \forall x, P}{\Gamma \vdash P[x \leftarrow t]} \forall e$
\exists	$\frac{\Gamma \vdash P[x \leftarrow t]}{\Gamma \vdash \exists x, P} \exists i$	$\frac{\Gamma \vdash \exists x, P \quad \Gamma, P \vdash C \quad x \notin vl(\Gamma, C)}{\Gamma \vdash C} \exists e$

	Hypothèses
\perp	$\overline{\Gamma, \perp} \vdash C \perp h$
\neg	$\frac{\Gamma, \neg A \vdash A}{\Gamma, \neg A \vdash C} \neg h$
\wedge	$\frac{\Gamma, A, B \vdash C}{\Gamma, A \wedge B \vdash C} \wedge h$
\vee	$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \vee B \vdash C} \vee h$
\Rightarrow	$\frac{\Gamma, B \vdash C \quad \Gamma, A \Rightarrow B \vdash A}{\Gamma, A \Rightarrow B \vdash C} \Rightarrow h$
\forall	$\frac{\Gamma, (\forall x, P), P[x \leftarrow t] \vdash C}{\Gamma, (\forall x, P) \vdash C} \forall h$
\exists	$\frac{\Gamma, P \vdash C \quad x \notin vl(\Gamma, C)}{\Gamma, (\exists x, P) \vdash C} \exists h$

Partie 4 : Réseaux

Dans cette partie, nous considérons un réseau composé de 3 machines : la machine A sur laquelle intervient un utilisateur ; et deux autres machines hébergeant respectivement un serveur DHCP et un serveur DNS. Le tout est relié à Internet à travers une passerelle (Routeur R1) afin que l'utilisateur profite notamment de deux autres services différents : WEB et FTP.

Les serveurs DNS, DHCP, WEB et FTP possèdent tous des adresses IP fixes. Le serveur DHCP attribue dynamiquement les adresses dans la plage 174.24.64.32 et 174.24.64.62 et la première adresse disponible est 174.24.64.45. Nous supposons qu'en dehors de la machine A, toutes les autres machines sont correctement configurées, viennent d'être installées (tous les caches sont vides) et opérationnelles. Leur fichier de configuration sont illustrés en partie sur la figure 1.

Nous rappelons que la réalisation d'un service donné sur ce réseau fait parfois appel à d'autres services générant des trafics additionnels. Des rappels sur les protocoles sont donnés en annexe.

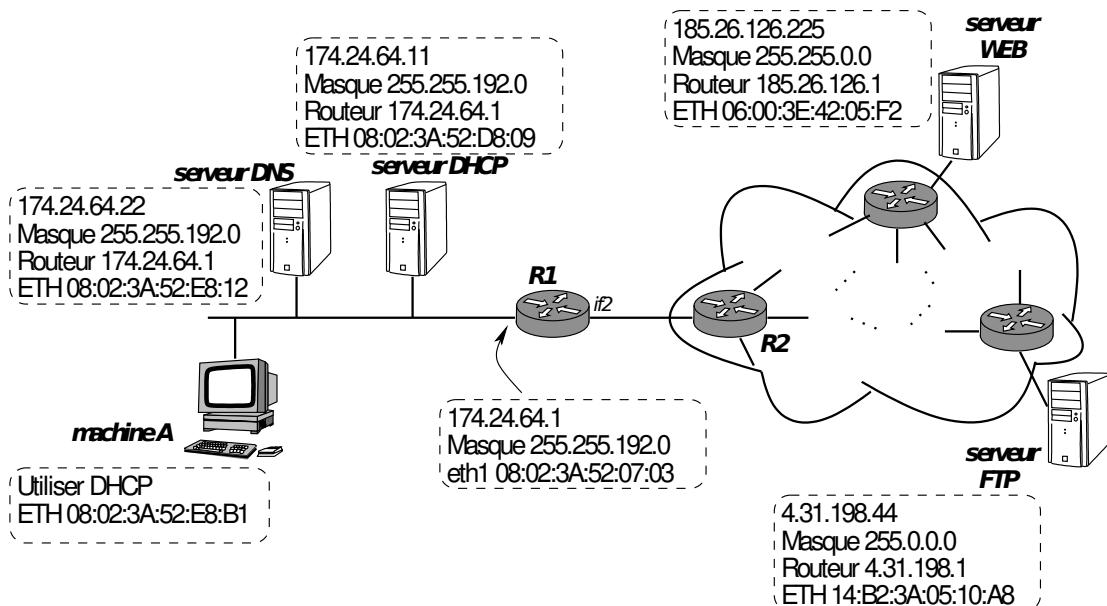


FIGURE 1 – Le réseau étudié.

- La machine A ne dispose pour l'instant que de son adresse MAC et sait qu'il faut utiliser DHCP pour obtenir une adresse IP. Décrire les trames transmises dans le réseau 174.24.64.0 pour que la machine A obtienne une adresse IP. Pour cela, indiquer le service auquel correspond chaque trame et le plus de détails possibles sur le type d'opération du service et les entêtes des unités de données encapsulées dans la trame (les adresses MAC, les adresses IP, et le cas échéant d'autres identifiants de niveau supérieur).

2. L'utilisateur de la machine A souhaite consulter le site web www.agreg-info.org. Décrire les trames échangées lorsque la machine A cherche à établir une session WEB sur le serveur (www.agreg-info.org). Nous supposons que le serveur DNS situé sur le même réseau que l'utilisateur dispose, lui, de l'information pour résoudre directement le nom de domaine www.agreg-info.org en 185.26.126.225.
3. Pour les trames identifiées dans les deux premières questions et encapsulant des messages de haut niveau (applicatifs), est ce qu'elles encapsulent toutes des segments du même protocole de transport ? Justifier.
4. Suite à une analyse de trafic sur l'interface *if2* du routeur R1 nous avons capturé les deux datagrammes représentés en figure 2. Visiblement il s'agit de deux fragments d'un même datagramme (numéro 85). En effet, le protocole IP au niveau de R2 a dû fragmenter le datagramme numéro 85 avant de le transmettre sur le lien R2 – R1.

Ver.	IHL	TOS	Total Length		Ver.	IHL	TOS	Total Length	
4	5	0	1020		4	5	0	480	
Identification 85		Flags	Fragment Offset 1 0		Identification 85		Flags	Fragment Offset 0 125	
TTL 55	Protocol	Header Checksum 615356		TTL 55	Protocol	Header Checksum 725354		Source IP address 185.26.126.225	
Source IP address 185.26.126.225		Destination IP address 174.24.64.45		Destination IP address 174.24.64.45		Data 'x' octets		Data 'y' octets	

FIGURE 2 – Deux datagrammes. Les acronymes des champs sont comme suit : Ver. (Version), IHL (Internet Header Length), TOS (Type of Service), TTL (Time To Live).

- a– Donner l'unité maximale de transmission (MTU) sur le lien R2 – R1. Pour quelles raisons peut-on limiter la taille maximale des trames sur les supports de transmission ?
- b– Même si la MTU dans le réseau 174.24.64.0 est de 1500 *octets*, le protocole IP ne réassemble les fragments qu'à la destination finale. Pour quelles raisons ?
- c– Le champs TTL est utilisé pour détruire les datagrammes lorsque sa valeur arrive à 0. Sachant que la valeur du TTL du datagramme original (à la source) était de 64 au départ, combien de routeurs ont été traversés par ces deux datagrammes ?
- d– Quelles sont les valeurs de '*x*' et '*y*' dans les champs data des deux fragments ?
- e– La somme de contrôle "Header Checksum" sert à la fonction du contrôle d'erreurs au niveau de l'entête du datagramme IP. Cette fonction est-elle réalisée uniquement au niveau de la destination ou au niveau de chaque routeur traversé ? Justifier.
5. À présent, l'utilisateur souhaite faire un transfert de fichiers vers le serveur FTP.
- a– Ce service nécessite également l'établissement d'une connexion. Quelle est l'entité de protocole qui se charge de l'établissement de cette connexion ? Cette même entité existe sur R1, sur R2, ou sur tous les routeurs ?
- b– Vu que la machine A dispose déjà d'une connexion TCP avec le serveur WEB, peut-il établir une seconde connexion pour le service FTP ? Si oui, comment l'entité protocolaire distingue-t-elle les deux connexions ?

- c– On considère dans la suite que la machine A a normalement terminé le téléchargement et a fermé sa connexion avec le service FTP. La connexion avec le service WEB s'est brutalement interrompue et la machine A en démarre une nouvelle (toujours vers www.agreg-info.org). Est-il possible que des segments de la première connexion du service WEB interfèrent avec ceux de la seconde ? Justifier.
6. Nous considérons la table de routage du routeur R2. Quatre datagrammes arrivent sur R2 et ont respectivement comme adresse IP destination : 174.24.64.45, 188.114.255.2, 188.114.255.4, 67.44.21.97.
- | Destination | Prochain saut | Interface |
|------------------|---------------|-----------|
| 174.24.64.0/18 | R1 | if1 |
| 67.44.16.0/20 | R3 | if3 |
| 67.44.16.0/22 | R4 | if4 |
| 188.114.255.0/30 | R3 | if3 |
| 0.0.0.0/0 | R5 | if2 |
- a– Donner le prochain saut de chacun de ces paquets en expliquant le principe de fonctionnement suivi et les opérations réalisées dans la table.
b– Nous avons, par la suite, constaté d'autres paquets destinés toujours à la machine 174.24.64.45. Ces paquets proviennent de la même source et ont exactement la même taille. En revanche leur latences respectives étaient différentes. Donner les raisons de cette variabilité en décrivant les facteurs impactant les délais de transmission de bout-en-bout.
7. Nous considérons cette fois un système autonome constitué de six routeurs nommés R1, R2, R3, R4, R5 et R6 (voir figure 3). Le protocole de routage utilisé est RIP (Routing Information Protocol) s'appuyant sur l'algorithme de détermination des routes décentralisé Bellman-Ford.

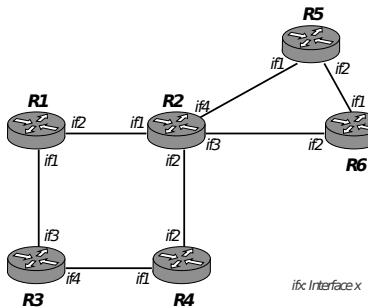


FIGURE 3 – Système autonome à six routeurs.

- a– Établir les tables de routage des routeurs R2 et R3. L'ordre dans lequel circulent les messages RIP a-t-il de l'importance ?
b– La liaison entre R2 et R6 tombe en panne. Expliquer comment les routeurs R2 et R3 détectent la rupture et montrer comment l'information se propage aux autres routeurs.
c– Expliquer la différence entre une panne d'un routeur et une rupture de lien avec l'incidence que cela pourrait avoir sur les tables de routage.
8. Supposons que la machine A et les deux serveurs DNS et DHCP sont sur un réseau ETHERNET 1 Gbit/s qui utilise une trame minimale de 512 octets.
a– Quel est le temps d'émission d'une trame de longueur minimale ?

- b– Peut-on en déduire la période de vulnérabilité dans un tel réseau ?
 c– Décrire la différence entre ce débit support, le débit utile et le débit réel en s'appuyant sur les facteurs influençant les variations du débit.

Annexes partie 4

Annexe A - Encapsulation dans TCP/IP et dans UDP/IP

Dans la figure 4, nous présentons un exemple d'encapsulation dans TCP/IP. Dans cet exemple, un segment TCP contenant un message de données est encapsulé dans datagramme IP, lui même encapsulé à son tour dans une trame ETHERNET. Un segment UDP serait encapsulé de la même manière qu'un segment TCP.

Chacune de ces couches (TCP|UDP, IP, ETHERNET) sont gérées par une entité protocolaire associée : le processus décodant les informations spécifiques à cette couche.

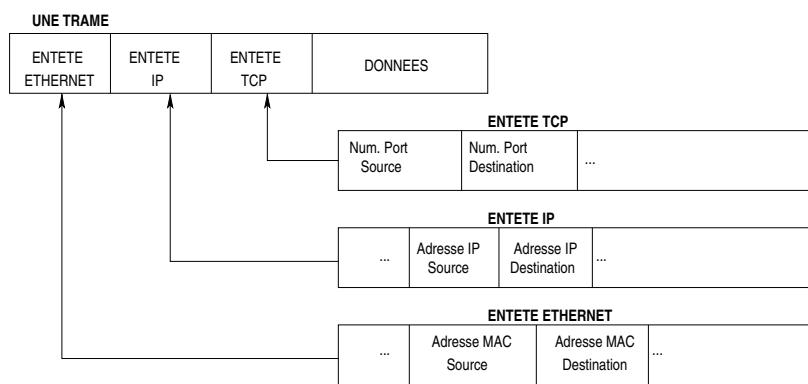


FIGURE 4 – Encapsulation de données dans TCP/IP.

Annexe B - Principe de DHCP

Le serveur applicatif DHCP attribue dynamiquement des adresses IP dans une plage d'adresses libres. Le serveur fonctionne sur un port UDP 67. Pour la mise en œuvre d'un tel service quatre types de messages sont utilisés :

- Le client diffuse un datagramme *DHCP_DISCOVER*.
- Tout serveur DHCP ayant reçu ce datagramme, envoie une offre *DHCP_OFFER* s'il est en mesure de proposer une adresse sur le réseau auquel appartient le client.
- Le client retient une des offres reçues et diffuse sur le réseau un datagramme de requête *DHCP_REQUEST*.
- Le serveur DHCP élabore un datagramme d'accusé de réception *DHCP_ACK* qui assigne au client l'adresse IP et son masque de sous-réseau, la durée du bail de cette adresse, etc.

Annexe C - Principe de DNS

Un serveur DNS (*Domain Name System ou Système de noms de domaine*) opère sur un port UDP numéro 53. Il se charge de la traduction des noms de domaine Internet en adresse IP. Pour la mise en œuvre d'un tel service, nous nous limiterons à l'échange suivant :

- Le client DNS de la machine envoie une requête DNS "Je recherche l'adresse IP de *www.agreg-info.org*".
- Le serveur DNS envoie une réponse au client (*www.agreg-info.org = @IP 185.26.126.225*).



EAE INF 2

SESSION 2022

AGREGATION CONCOURS EXTERNE

Section : INFORMATIQUE

ÉTUDE D'UN PROBLÈME INFORMATIQUE

Durée : 6 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique (y compris la calculatrice) est rigoureusement interdit.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

A

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	6200A	102	9423

De très grands entiers

Cette épreuve a pour objet la réalisation d'une bibliothèque d'entiers de précision arbitraire. On se place dans le cadre du langage Python et on se propose donc de construire une alternative aux entiers natifs de Python, dont on rappelle qu'ils sont déjà de précision arbitraire.

Préliminaires

Python. Les entiers natifs de Python, de type `int`, sont de précision arbitraire. Si n et m sont deux entiers Python positifs ou nuls, alors $n \ll m$ est l'entier $n \times 2^m$ et $n \gg m$ est l'entier $\lfloor n/2^m \rfloor$. L'entier $n \& m$ (resp. $n \mid m$ et $n \wedge m$) est le ET logique (resp. le OU logique et le OU exclusif logique) des entiers n et m , c'est-à-dire que le i -ième bit de $n \& m$ (resp. $n \mid m$ et $n \wedge m$) est obtenu en faisant le ET (resp. OU et OU exclusif) des i -èmes bits de n et m . L'addition de deux entiers n et m a un coût en $O(\max(\log n, \log m))$. La représentation d'un entier n occupe un espace $O(\log n)$.

Dans le langage Python, toute valeur est représentée par un objet, dont l'identité peut être obtenue avec la fonction `id`. Il s'agit là d'un entier, garanti unique et constant pendant toute la durée de vie de cet objet. Par ailleurs, on peut déterminer si deux objets x et y sont identiquement les mêmes avec le booléen `x is y`, ou au contraire distincts avec `x is not y`.

Le langage Python fournit nativement une structure de *dictionnaire*. On crée un nouveau dictionnaire, vide, avec `{}`. Si d est un dictionnaire et k une clé, on teste la présence d'une valeur associée à la clé k avec `k in d` et, le cas échéant, on récupère la valeur associée avec `d[k]`. On associe une valeur v à la clé k avec `d[k] = v` (et toute valeur précédemment associée à k , le cas échéant, est écrasée). L'ajout se fait en place. De même, le langage Python fournit nativement une structure d'*ensemble*. On crée un nouvel ensemble, vide, avec `set()`. Si s est un ensemble, on teste la présence d'un élément x dans s avec `x in s` et on ajoute l'élément x à s avec `s.add(x)`. L'ajout se fait en place.

En interne, un dictionnaire ou un ensemble est réalisé par une *table de hachage*, sur la base des méthodes `__hash__` et `__eq__` fournies par la classe des clés (pour un dictionnaire) ou des éléments (pour un ensemble). Ces deux méthodes se doivent d'être *cohérentes*, c'est-à-dire

$$\text{pour tous } x \text{ et } y, \text{ si } x.\text{__eq__}(y) = \text{True} \text{ alors } x.\text{__hash__}() = y.\text{__hash__}(). \quad (1)$$

Le langage Python fournit nativement une structure de *tableau redimensionnable*, appelé « liste ». On crée une liste vide avec `[]`. Si t est une liste, sa longueur est donnée par `len(t)`. Pour un entier k tel que $0 \leq k < \text{len}(t)$, on accède au k -ième élément de t avec `t[k]` et on le modifie avec `t[k] = v`. Ces deux opérations se font en temps constant. On étend la liste t avec un nouvel élément v , à la position `len(t)`, avec `t.append(v)` (et `len(t)` est incrémenté). Inversement, l'opération `t.pop()` supprime et renvoie le dernier élément de la liste t . En pratique, on considérera que les deux opérations `append` et `pop` se font également en temps constant.

Pour ouvrir un fichier texte `file` en lecture, on peut utiliser `open(file, 'r')`. On obtient alors un objet, avec notamment une méthode `readlines()` qui renvoie une liste contenant toutes les lignes du fichier comme autant de chaînes de caractères. Par ailleurs, si `s` est une chaîne de caractères, `s.split()` renvoie la liste de tous les mots non vides de `s`, dans l'ordre, les mots étant séparés par des caractères blancs (espaces et retours chariot). Ainsi, "`a bb c\n`".`split()` renvoie la liste `["a", "bb", "c"]`.

Complexité. Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. On considérera que toutes les opérations élémentaires de Python (affectation, comparaison avec `is`, accès à un élément de liste, etc.) s'effectuent en temps constant. La complexité sera exprimée sous la forme $O(f(n, m))$ où n et m sont les tailles des arguments de la fonction, et f une expression simple. Les calculs de complexité seront justifiés succinctement. Lorsqu'une question de programmation précise qu'une complexité est attendue, sauf demande explicite, il n'est alors pas nécessaire de justifier que la fonction écrite vérifie cette contrainte. Pour les calculs d'espace, on considérera qu'un pointeur occupe un espace constant.

Dépendances. Ce sujet contient plusieurs parties. Chaque partie utilise des définitions et des résultats des parties précédentes. Les questions restent néanmoins indépendantes, au sens où toute question peut être traitée en admettant les résultats énoncés dans les questions précédentes.

Attendus. Il est attendu des candidates et des candidats des réponses construites. Ils seront aussi évalués sur la précision, le soin et la clarté de la rédaction.

Partie I. Principe

Pour représenter de grands entiers, on exploite l'observation suivante : tout entier naturel s'écrit de manière unique

- soit comme l'entier 0 ;
- soit comme l'entier 1 ;
- soit comme $h \times 2^{2^p} + \ell$ avec $0 < h < 2^{2^p}$ et $0 \leq \ell < 2^{2^p}$.

Dans ce dernier cas, on note $\langle h, p, \ell \rangle$ ce triplet. Ainsi, l'entier 42 s'écrit $\langle 2, 2, 10 \rangle$ car $42 = 2 \times 2^{2^2} + 10 = 2 \times 16 + 10$. De même, l'entier 10 s'écrit $\langle 2, 1, 2 \rangle$ car $10 = 2 \times 2^{2^1} + 2 = 2 \times 4 + 2$ et l'entier 2 s'écrit $\langle 1, 0, 0 \rangle$ car $2 = 1 \times 2^{2^0} + 0 = 1 \times 2 + 0$.

À cette décomposition, on rajoute l'idée qu'un même triplet peut être construit de manière unique en mémoire. On a alors une représentation sous forme d'un graphe où les sommets sont des triplets $\langle h, p, \ell \rangle$, avec h , p et ℓ étant 0, 1 ou une référence à un autre sommet. Une telle représentation des entiers est baptisée IDD (pour *Integer Dichotomy Diagrams*). La figure 1 illustre l'IDD représentant l'entier 42.

Question 1. Dessiner l'IDD correspondant à l'entier 773.

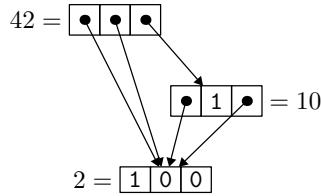


FIGURE 1 – Représentation de l'entier 42 par un IDD.

Nombres énormes. Pour $n \in \mathbb{N}$, on définit le *nombre énorme* $b(n)$ de la manière suivante :

$$\begin{aligned} b(0) &\stackrel{\text{def}}{=} 1 \\ b(n+1) &\stackrel{\text{def}}{=} \langle b(n), b(n), b(n) \rangle. \end{aligned} \tag{2}$$

Question 2. Donner la valeur de $b(1)$ en base 10. Donner la valeur de $b(2)$ en base 2.

Taille d'un entier. On définit la *taille* d'un entier n , notée $\mathbf{s}(n)$, comme le nombre de sommets distincts dans l'IDD qui le représente, les entiers 0 et 1 n'étant pas comptés comme des sommets. Ainsi, $\mathbf{s}(42) = 3$ au regard de la figure 1. En particulier, $\mathbf{s}(0) = \mathbf{s}(1) = 0$.

Question 3. Montrer que $b(n)$ est le plus grand entier i tel que $\mathbf{s}(i) \leq n$.

Partie II. Représentation en Python

On s'intéresse maintenant à la construction des IDD dans le langage Python. Un IDD est représenté par un objet de la classe `IDD`, dont le code est donné dans la figure 2. Cet objet possède trois champs, `hi`, `p` et `lo`, qui sont eux-mêmes trois objets de la classe `IDD`, et il représente l'entier $\langle \mathbf{hi}, \mathbf{p}, \mathbf{lo} \rangle$. Pour assurer l'unicité en mémoire d'un même entier, on utilise la technique du *hash-consing* : une table globale, `table` (ligne 4), contient tous les IDD déjà construits. Il s'agit d'un dictionnaire dont les clés sont des objets de la classe `Triple` (lignes 28–41) et dont les valeurs sont des objets de la classe `IDD`. Pour construire un IDD, on se sert de la fonction `IDD.create` (ligne 11). Elle consulte la table pour déterminer si l'objet a déjà été construit (ligne 15). Le cas échéant, elle le renvoie (ligne 16). Sinon, l'objet est construit, ajouté à la table et renvoyé (lignes 18–20).

La classe `Triple` (lignes 28–41) représente un triplet de trois IDD, avec trois composantes `hi`, `p` et `lo`. La classe `Triple` est munie d'une fonction de hachage (lignes 35–36) et d'une fonction d'égalité (lignes 38–41), qui sont appelées lorsque le dictionnaire `table` est utilisé.

Enfin, on construit deux objets particuliers, `zero` et `one`, pour représenter les entiers 0 et 1 (lignes 43–44). Pour faciliter le code de certaines fonctions, il est pratique de définir les trois composantes de `zero` et `one` en termes de ces mêmes objets (lignes 45–46).

```

1  class IDD:
2      """cet objet a 3 champs: hi,p,lo, trois autres IDD"""
3
4      table = {}
5
6      def __init__(self, hi, p, lo):
7          self.hi = hi
8          self.p = p
9          self.lo = lo
10
11     def create(hi, p, lo):
12         if hi is zero:
13             return lo
14         t = Triple(hi, p, lo)
15         if t in IDD.table:
16             return IDD.table[t]
17         else:
18             i = IDD(hi, p, lo)
19             IDD.table[t] = i
20         return i
21
22     def __hash__(self):
23         return id(self)
24
25     def __eq__(self, other):
26         return self is other
27
28 class Triple:
29     """cette classe sert uniquement de clé dans table"""
30     def __init__(self, hi, p, lo):
31         self.hi = hi
32         self.p = p
33         self.lo = lo
34
35     def __hash__(self):
36         return 31*(31*(id(self.hi) + id(self.p)) + id(self.lo))
37
38     def __eq__(self, other):
39         assert isinstance(other, Triple)
40         return self.hi is other.hi and self.p is other.p and \
41                 self.lo is other.lo
42
43 zero = IDD(None, None, None)
44 one = IDD(None, None, None)
45 zero.hi, zero.p, zero.lo = zero, zero, zero
46 one.hi, one.p, one.lo = zero, zero, one

```

FIGURE 2 – Construction des IDD en Python.

Par la suite, le constructeur de la classe `IDD` ne sera jamais utilisé. On utilisera exclusivement la fonction `IDD.create`. De plus, on suppose que seule la fonction `IDD.create` accède au dictionnaire stocké dans `table`. Ainsi, pour construire une valeur de type `IDD` représentant l'entier 42, on peut écrire le code suivant :

```
i2 = IDD.create(one, zero, zero)
i10 = IDD.create(i2, one, i2)
i42 = IDD.create(i2, i2, i10)
```

Dans tout le reste de ce sujet, on utilise la variable `n` pour désigner un entier Python de type `int` et les variables `i` et `j` pour désigner des entiers `IDD` de type `IDD`.

Question 4. Expliquer pourquoi il est légitime de considérer que la fonction `IDD.create` s'exécute en temps constant.

Question 5. Justifier la définition des méthodes `__hash__` (lignes 22–23) et `__eq__` (lignes 25–26) de la classe `IDD`. Expliquer pourquoi on peut considérer que les opérations d'ajout et de recherche dans un ensemble d'éléments de type `IDD` ou dans un dictionnaire dont les clés sont de type `IDD` s'exécutent en temps constant.

Question 6. Donner le code Python d'une fonction `big(n: int) -> IDD` correspondant à la fonction b (équation (2) page 3). La complexité en temps doit être en $O(n)$, mais il n'est pas demandé de la justifier.

Question 7. Donner le code Python d'une fonction `size(i: IDD) -> int` qui renvoie la taille de l'entier `i`, c'est-à-dire $s(i)$. On rappelle que les objets `zero` et `one` ne doivent pas être décomptés. La complexité en temps doit être $O(s(i))$. On justifiera la complexité.

Poids de Hamming. Le poids de Hamming d'un entier naturel n , noté $\text{pop}(n)$ (pour *population count*), est le nombre de chiffres 1 dans l'écriture de n en base 2. Ainsi, $\text{pop}(42) = 3$ car $42 = 101010_2$. La figure 3 contient une fonction `pop` qui renvoie le poids de Hamming d'un `IDD`.

Question 8. Montrer que la fonction `pop` est correcte. *Indication : Proposer un invariant pour le dictionnaire `memo` et montrer que la fonction `compute` le préserve.*

Question 9. Donner et justifier la complexité de la fonction `pop`, en fonction de $s(i)$. Peut-on calculer $\text{pop}(\text{big}(100))$, qui vaut 2^{100} , en un temps raisonnable ?

```

1 def pop(i: IDD) -> int:
2     """nombre de 1 dans la représentation en base 2 de i"""
3     memo = {}
4     memo[zero] = 0
5     memo[one] = 1
6     def compute(i):
7         if i in memo:
8             return memo[i]
9         else:
10            v = compute(i.hi) + compute(i.lo)
11            memo[i] = v
12            return v
13    return compute(i)

```

FIGURE 3 – Fonction pop.

```

1 def to_int(i: IDD) -> int:
2     """convertit un IDD en un entier Python"""
3     if i is zero:
4         return 0
5     elif i is one:
6         return 1
7     else:
8         return to_int(i.hi) << (1 << to_int(i.p)) | to_int(i.lo)

```

FIGURE 4 – Fonction to_int.

Partie III. Conversions

Dans cette partie, on étudie différentes opérations de conversions vers et depuis d'autres formats de représentation.

Avec le type int. La figure 4 contient le code d'une fonction `to_int` qui convertit un IDD vers le type `int` de Python, en supposant que la mémoire est suffisamment grande pour stocker le résultat.

Question 10. Montrer que la fonction `to_int` est correcte.

Question 11. La fonction récursive `to_int` est-elle susceptible de faire déborder la pile d'appels de Python (par défaut limitée à 1000 appels imbriqués) ?

Question 12. Donner le code d'une fonction Python `of_int(n: int) -> IDD` qui convertit un entier Python en IDD.

Sérialisation. Pour écrire un IDD $n \geq 2$ dans un fichier, on se propose d'utiliser le format texte suivant. Le fichier contient exactement $s(n)$ lignes et chaque ligne est de la forme

```
i j k l
```

où i, j, k et l sont quatre entiers, avec $2 \leq i \leq s(n) + 1$ et $0 \leq j, k, l < i$. L'entier i numérote la ligne, à partir de 2. Cette ligne définit un nouvel IDD, numéroté i , comme valant $\langle j, k, l \rangle$ où j, k et l sont les trois IDD respectivement numérotés j, k et l . Les numéros 0 et 1 font référence aux IDD 0 et 1. Le fichier représente l'IDD défini par la dernière ligne. Ainsi, l'IDD représentant l'entier 42 peut être sérialisé par les trois lignes suivantes :

```
2 1 0 0
3 2 1 2
4 2 2 3
```

Ces trois lignes correspondent aux trois IDD de la figure 1.

Question 13. Cette représentation est-elle unique ? Si oui, justifier. Sinon, donner deux fichiers définissant le même entier.

Question 14. Décrire un algorithme pour réaliser cette sérialisation, c'est-à-dire pour imprimer successivement les lignes du fichier qui représente un IDD n donné. Donner sa complexité. On ne demande pas d'écrire le code Python.

Question 15. Donner le code Python d'une fonction `parser(file: str) -> IDD` qui reconstruit l'IDD décrit par le contenu du fichier `file`. On suppose que ce fichier contient la sérialisation d'un IDD, *i.e.*, on ne demande pas de vérifier la bonne formation de ce fichier. La complexité en temps doit être proportionnelle à la taille du fichier, mais il n'est pas demandé de la justifier.

Partie IV. Arithmétique

Question 16. Donner le code Python d'une fonction `compare(i: IDD, j: IDD) -> int` qui compare deux IDD. Elle renvoie l'entier -1 si $i < j$, l'entier 0 si $i = j$ et l'entier 1 si $i > j$.

Incrémantation et décrémentation. La figure 5 contient le code de deux fonctions `xp` et `pred`. La fonction `xp` calcule $2^{2^i} - 1$ pour un argument $i \geq 0$. La fonction `pred` calcule $i - 1$ pour un argument $i > 0$.

Question 17. Montrer que le calcul de `xp(i)` ou de `pred(i)` termine toujours.

Question 18. En utilisant la fonction `xp`, donner le code Python d'une fonction `succ(i: IDD) -> IDD` qui calcule $i + 1$.

```

1 def xp(i: IDD) -> IDD:
2     """2^(2^i)-1"""
3     if i is zero:
4         return one
5     else:
6         p = pred(i)
7         j = xp(p)
8         return IDD.create(j, p, j)
9
10 def pred(i: IDD) -> IDD:
11     """prédecesseur i-1, pour i>0"""
12     assert i is not zero
13     if i is one:
14         return zero
15     elif i.lo is not zero:
16         return IDD.create(i.hi, i.p, pred(i.lo))
17     elif i.hi is one:
18         return xp(i.p)
19     else:
20         return IDD.create(pred(i.hi), i.p, xp(i.p))

```

FIGURE 5 – Fonctions `xp` et `pred`.

Ajout/retrait du bit de poids fort. Pour $n > 0$, on pose $R(n) = (m, i)$ avec $n = m + 2^i$ et $0 \leq m < 2^i$. Dit autrement, i est le bit de poids fort de l'entier n . Inversement, on pose $I(m, i) = m + 2^i$ pour $0 \leq m < 2^i$. Pour calculer $R(n)$ et $I(m, i)$ sur les IDD, on se donne les équations suivantes :

$$R(1) = (0, 0) \tag{3}$$

$$R(\langle h, p, \ell \rangle) = (\langle m, p, \ell \rangle, I(i, p)) \quad \text{si } m \neq 0 \tag{4}$$

$$= (\ell, I(i, p)) \quad \text{sinon} \tag{5}$$

avec $(m, i) = R(h)$

$$I(0, 0) = 1 \tag{6}$$

$$I(0, 1) = \langle 1, 0, 0 \rangle \tag{7}$$

$$I(1, 1) = \langle 1, 0, 1 \rangle \tag{8}$$

$$I(\langle h, p, \ell \rangle, i) = \langle I(0, e), j, \langle h, p, \ell \rangle \rangle \quad \text{si } j > p \tag{9}$$

$$= \langle I(h, e), j, \ell \rangle \quad \text{sinon} \tag{10}$$

avec $(e, j) = R(i)$

Question 19. Justifier les équations (3)–(10).

Dans la suite, on suppose avoir écrit deux fonctions Python réalisant ces calculs, sous la forme suivante :

```

def rmsb(i: IDD) -> Tuple[IDD, IDD]:
    """extrait le bit 1 de point fort de i>0, c'est-à-dire
    renvoie (m, j) avec i = m + 2^j et 0 <= m < 2^j"""
    ...
def imsb(i: IDD, j: IDD) -> IDD:
    """renvoie i + 2^j pour 0 <= i < 2^j"""
    ...

```

Question 20. Donner le code Python d'une fonction `power2(i: IDD) -> IDD` qui calcule 2^i .

Question 21. Donner le code Python d'une fonction `binary_length(i: IDD) -> IDD` qui calcule le nombre de chiffres dans l'écriture en base 2 de l'entier `i`.

Question 22. Donner le code d'une fonction Python `print2(i: IDD)` qui imprime l'entier `i` en base 2. Ainsi, `print2(of_int(42))` doit afficher

101010

On suppose que le nombre de chiffres est suffisamment raisonnable pour que l'impression ait une chance de terminer.

Autres opérations arithmétiques. Il est également possible de définir les opérations d'addition, de soustraction, de multiplication ou encore de division sur les IDD, mais cela dépasserait le cadre de ce sujet.

Partie V. Opérations logiques et applications

La structure des IDD est parfaitement adaptée à la définition d'opérations logiques (ET, OU, OU exclusif) sur l'écriture en binaire des entiers correspondants. Ainsi, on peut définir le ET logique de deux IDD `s` et `t`, noté $s \wedge t$, avec les cinq équations suivantes :

$$0 \wedge t = 0 \tag{11}$$

$$s \wedge s = s \tag{12}$$

$$s \wedge t = t \wedge s \quad \text{si } s > t \tag{13}$$

$$s \wedge t = s \wedge t.\text{lo} \quad \text{si } s.\text{p} < t.\text{p} \tag{14}$$

$$s \wedge t = \langle s.\text{hi} \wedge t.\text{hi}, s.\text{p}, s.\text{lo} \wedge t.\text{lo} \rangle \quad \text{sinon} \tag{15}$$

Question 23. Donner des équations comparables pour la définition de l'opération OU (notée $s \vee t$) et OU exclusif (notée $s \oplus t$).

Question 24. On pourrait écrire une fonction Python `log_and(i: IDD, j: IDD) -> IDD`, récursive, qui suive exactement les équations (11)–(15). Montrer que sa complexité en temps pourrait être exponentielle en les tailles `s(i)` et `s(j)` de ses arguments.

Question 25. En utilisant le principe de mémoïsation, donner le code Python d'une fonction `log_and` plus efficace dont la complexité en temps est $O(s(i) \times s(j))$. On justifiera la complexité.
Indication : On pourra introduire une classe pour des paires d'IDD servant de clés dans un dictionnaire.

Dans la suite, on suppose qu'on a écrit de la même façon des fonctions `log_or` et `log_xor`, également de complexité en temps $O(s(i) \times s(j))$.

Application : ensembles d'entiers naturels. Un ensemble fini d'entiers naturels $S \subseteq \mathbb{N}$ peut être représenté par un entier n en posant

$$n = \sum_{i \in S} 2^i. \quad (16)$$

Dit autrement, les chiffres 1 dans la représentation de n en base 2 indiquent les éléments de l'ensemble S . Ainsi, l'ensemble $S = \{1, 4, 5, 8, 9\}$ est représenté par l'entier 818, car $818 = 1100110010_2$.

Un ensemble S est donc naturellement représenté par l'IDD qui encode l'entier n défini par (16). En particulier, les fonctions `log_and`, `log_or` et `log_xor` introduites plus haut calculent respectivement l'intersection, l'union et la disjonction exclusive de deux ensembles.

Question 26. Donner le code Python d'une fonction `difference(s: IDD, t: IDD) -> IDD` qui calcule la différence ensembliste pour des ensembles représentés par des IDD.

Question 27. Donner le code Python d'une fonction `mem(n: int, s: IDD) -> bool` qui détermine si l'entier `n` appartient à l'ensemble représenté par `s`.

Question 28. Donner le code Python d'une fonction `subset(s: IDD, t: IDD) -> bool` qui détermine si $s \subseteq t$ pour deux ensembles représentés par des IDD.

Question 29. Justifier, informellement, que l'espace mémoire occupé par la représentation physique d'un IDD n est proportionnel à sa taille $s(n)$.

Question 30. On admet l'inégalité $s(n) \leq \text{pop}(n) \times (n.p + 1)$ pour tout IDD n . Soit S un ensemble de K entiers, qui s'écrivent tous sur au plus W bits. Majorer l'espace utilisé par un IDD qui représente S selon (16), en fonction de K et W . Comparer avec l'espace utilisé par une liste d'entiers Python pour représenter ce même ensemble.

Question 31. Montrer que l'espace *total* occupé par les IDD représentant tous les entiers $2, 3, \dots, n$ est en $O(n)$.

Question 32. Soit $S = \{0, 1, \dots, K - 1\}$. Comparer l'espace occupé simultanément par tous les sous-ensembles de S , dans les deux cas suivants :

1. les ensembles sont représentés par des IDD ;
2. les ensembles sont représentés par des listes d'entiers.

Partie VI. Pour aller plus loin

Question 33. La structure des IDD ne permet pas de tirer parti des opérations natives fournies par la machine, comme par exemple des opérations arithmétiques ou logiques sur 64 bits fournies par le processeur. Proposer une adaptation de la structure des IDD à même de tirer le meilleur parti d'une arithmétique native sur W bits.

Question 34. Dans ce sujet, nous nous sommes limités à des entiers naturels. Proposer une extension de cette bibliothèque à des entiers relatifs. On ne demande pas d'écrire le code mais on demande d'être précis quant à la représentation choisie et à l'adaptation des différentes opérations.

* *
*



EAE INF 3

SESSION 2022

AGREGATION CONCOURS EXTERNE

Section : INFORMATIQUE

ÉPREUVE SPÉCIFIQUE SELON L'OPTION CHOISIE :

- ÉTUDE DE CAS INFORMATIQUE
- FONDEMENTS DE L'INFORMATIQUE

Durée : 6 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique (y compris la calculatrice) est rigoureusement interdit.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

A

Épreuve spécifique

Étude de cas informatique	1
Fondements de l'informatique	18

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

► Etude de cas informatique :

Concours	Section/option	Epreuve	Matière
EAE	6200A	103	9424

► Fondement de l'informatique :

Concours	Section/option	Epreuve	Matière
EAE	6200A	103	9425

Étude de cas informatique

Préliminaires

Le projet proposé s'inspire d'un projet réel. La procédure de conception utilisée dans le sujet s'inspire librement de l'approche agile avec une organisation en plusieurs parties. Chaque partie comprend la définition d'un certain nombre d'objectifs concrets, la réflexion sur les moyens de les atteindre, la préparation d'une procédure de validation permettant de vérifier si les objectifs sont atteints et une discussion critique de l'ensemble du processus.

Attendus. Il est attendu des candidates et des candidats des réponses construites. Ils seront aussi évalués sur la précision, le soin et la clarté de la rédaction.

Dépendances. Ce sujet contient quatre parties. La première partie est guidée de manière assez précise car elle sert de point de départ à l'ensemble des autres parties. Les autres parties sont largement indépendantes. Il est possible de les aborder dans l'ordre qui vous conviendra le mieux.

Partie I. Problème d'échecs

Le but de ce sujet est de développer la plate-forme **ChessMate** permettant de spécifier et résoudre des problèmes d'échecs tels que le problème des huit dames mais aussi d'autres problèmes tels que les problèmes des mats en N coups. **ChessMate** permettra également de pouvoir jouer une partie d'échecs.

Fonction de résolution du problème des huit dames

Nous allons commencer par le problème des huit dames : “Le but du **problème des huit dames** est de placer huit dames d'un jeu d'échecs sur un échiquier de 8 x 8 cases sans que les dames ne puissent se menacer mutuellement, conformément aux règles du jeu d'échecs (la couleur des pièces étant ignorée). Par conséquent, deux dames ne devraient jamais partager la même rangée, colonne, ou diagonale.” (*Problème des huit dames, d'après Wikipedia*)

Le problème des huit dames a 92 solutions distinctes. La figure 1 en présente une. On propose la fonction *check_solution8queens* en Python pour vérifier si une solution au problème des huit dames est correcte. Cette fonction trie les cases où sont positionnées les dames sur l'échiquier et vérifie que les dames ne sont pas sur la même rangée ou colonne. Du moins c'est l'intention du programmeur.

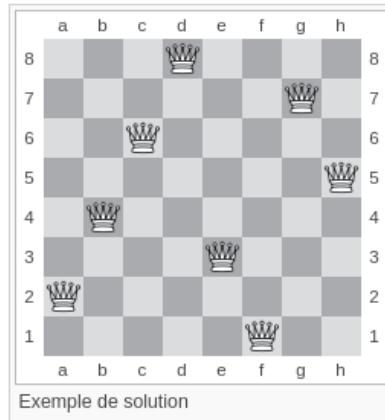


FIGURE 1 – Solution du problème des huit dames

Listing 1 – fonction check_solution8queens

```

1   from math import floor
2
3   def check_solution8queens(sol):
4       """
5           sol est une solution au probl\`eme des 8 dames
6           sol est un ensemble de 8 valeurs
7           exemple: [8, 5, 20, 25, 42, 59, 54, 39]
8           correspondant \`a une solution correcte (cf \`echiquier ci-dessus)
9           retourne True si la solution est correcte, False sinon
10
11          /\ l'\`echiquier est repr\`esent\'e par une liste de 64 cases
12          la case 0 est le coin bas gauche
13          la case 63 est le coin haut droit
14
15          56 57 58 59 60 61 62 63
16          48 49 50 51 52 53 54 55
17          40 41 42 43 44 45 46 47
18          32 33 34 35 36 37 38 39
19          24 25 26 27 28 29 30 31
20          16 17 18 19 20 21 22 23
21          8 9 10 11 12 13 14 15
22          0 1 2 3 4 5 6 7
23
24
25      sortedSol = sorted(sol)
26      for i in range(len(sortedSol)):
27          square = sortedSol[i]
28          otherSquares = sortedSol[i+1:]
29          for otherSquare in otherSquares:
30              if square % 8 == otherSquare % 8:
31                  return False
32              if (floor(otherSquare/8) - floor(square/8)) == 0:
33                  return False
34
35      return True

```

Question 1. La fonction du listing 1 est en fait fausse car incomplète. Elle retourne True pour certaines solutions qui ne sont en fait pas correctes. Programmer trois tests qui montrent trois défauts différents de la fonction `check_solution8queens(sol)`. Le code de vos tests doit préciser

les données en entrée et le résultat attendu. Commenter en quoi vos trois tests vérifient des propriétés différentes.

L'échiquier avec une classe

Maintenant que nous avons testé la fonction `check_solution8queens(sol)`, on s'intéresse à l'encodage de l'échiquier. L'encodage proposé dans la fonction `check_solution8queens(sol)` n'est pas idéal. En principe, une case est identifiée par une lettre pour la colonne (de 'a' à 'h') et un chiffre pour la ligne (de 1 à 8). Ainsi la case 'a1' correspond à la case 0 de l'encodage proposé à la question 1, la case 'b1' correspond à la case 1, etc.

Pour réaliser cet encodage, nous proposons la classe **Board** (échiquier) qui propose les méthodes suivantes :

- `putQueen(self, square)` qui place une dame sur la case `square`, avec `square` une chaîne de caractères de la forme 'a1'.
- `hasQueen(self, square)` qui retourne True si la case `square` est occupée par une reine, False sinon.
- `removeQueen(self, square)` qui retire la dame de la case `square`.

Listing 2 – code d'utilisation de la classe Board

```

1 board = Board() # l'échiquier est vide
2 board.hasQueen('a1') # False
3 board.putQueen('a1')
4 board.hasQueen('a1') # True
5 board.removeQueen('a1')
6 board.hasQueen('a1') # False

```

Listing 3 – code de manipulation des chaînes de caractères

```

1 b8 = 'b8'
2 row = int(b8[1]) # 8
3 column = int(ord(b8[0]) - 96) # 2
4 a1 = chr(97) + str(1) # 'a1'

```

Question 2. Programmer la classe **Board** et faire en sorte que le code du listing 2 puisse être exécuté. Le code du listing 3 vous aide à effectuer les manipulations de chaînes de caractères.

Question 3. Programmer la fonction `queen_legal_moves(square)` qui retourne l'ensemble des cases sur lesquelles une reine peut bouger si elle est placée sur la case **square** et en considérant un échiquier vide. La fonction doit retourner toutes les cases de sa ligne, de sa colonne, ainsi que les deux diagonales qui passent par sa case. Enfin, la case d'origine ne doit pas appartenir à l'ensemble retourné.

Question 4. Modifier la fonction `check_solution8queens(sol)` pour qu'elle prenne en entrée un échiquier (un objet instance de la classe **Board**). La fonction devient alors : `check_solution8queens(board)`. Vous vous appuierez sur la fonction `queen_legal_moves(square)` pour savoir si l'échiquier passé contient une solution correcte au problème des huit dames.

La dame et les autres pièces

On considère la classe **Queen** qui représente une dame. Dans un premier temps cette classe ne contient qu'une seule méthode : *legal_moves(self, square)*. Cette méthode retourne la liste des cases sur lesquelles une dame peut se déplacer si elle est placée sur la case **square** et si on considère un échiquier vide. Le code de cette méthode est donc le même que la fonction *queen_legal_moves(square)*.

Question 5. La conception de la classe **Queen** fait le choix que la position des pièces est connue par classe **Board** (et non par **Queen**). Discuter de l'intérêt de ce choix de conception (avantages et inconvénients).

Déplacement des pièces

Pour rendre le jeu plus intéressant, on va ajouter d'autres pièces : le roi, la tour et le fou. Quelques explications sur le déplacement de ces pièces aux échecs sont données ci-dessous. Notez que des explications sur les autres pièces du jeu d'échecs (le cavalier ou le pion), ne sont pas données car nous ne les considérerons pas dans la suite du sujet. Nous ne traiterons également pas des règles comme le pat ou la prise en passant.

Question 6. Programmer en Python les classes **Queen** (Dame), **King** (Roi), **Rook** (Tour) et **Bishop** (Fou). On ne considère pas le cavalier et le pion. On se contentera d'implémenter les mouvements possibles des pièces, en s'appuyant sur les explications données préalablement. Ces classes proposent la même méthode *legal_moves(self, square)* qui donne la liste des cases sur lesquelles une pièce peut bouger à partir d'une case de départ et en considérant un échiquier vide. Il est attendu à ce que vous utilisiez l'héritage pour définir ces classes et faire en sorte qu'il y ait peu de redondance de code.

Question 7. Modifier la classe **Board** et remplacez les méthodes *putQueen*, *hasQueen* et *removeQueen* par des méthodes *put*, *has* et *remove* pour toutes les pièces considérées (**Queen**,

Le fou, la tour et la dame sont des pièces à longue portée (ou pièces de lignes) : elles peuvent se déplacer le long de lignes. Les fous se déplacent toujours sur les cases d'une même couleur, en diagonale. La tour est capable de se mouvoir en ligne droite, verticalement, horizontalement, sur un nombre quelconque de cases inoccupées (voir figure 3). La dame est capable de se mouvoir en ligne droite, verticalement, horizontalement, et diagonalement, sur un nombre quelconque de cases inoccupées (voir figure 3). Le roi se déplace d'une seule case à la fois dans toutes les directions. Les pièces (excepté le pion) capturent une pièce adverse qui se trouve sur leur trajectoire, sans pouvoir aller au-delà. La pièce qui capture prend la place de la pièce capturée, cette dernière étant définitivement retirée de l'échiquier.

FIGURE 2 – Explications sur les déplacements des pièces

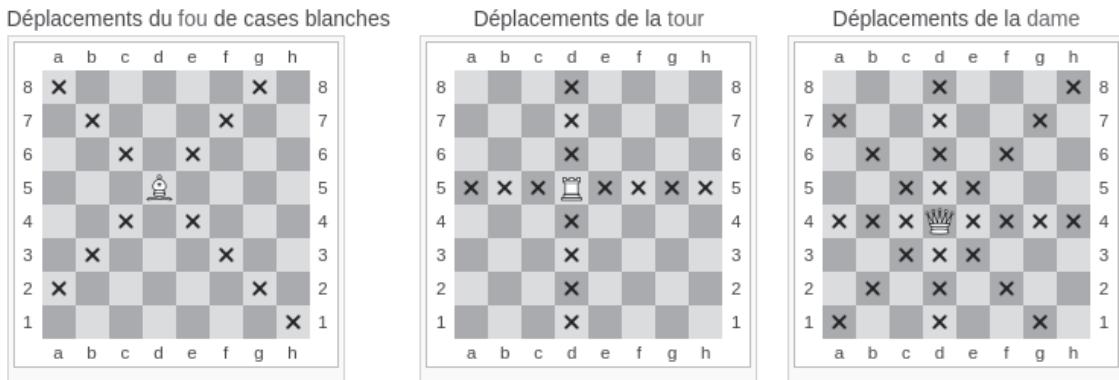


FIGURE 3 – Déplacements Fou, Tour et Dame

King, Rook et Bishop). Vous ajouterez également la méthode `get(self,square)` qui retourne la pièce positionnée à la case square, ou `None` si aucune pièce n'y est positionnée. Il est attendu que votre code soit le plus générique possible et qu'il y ait le moins de redondance et le plus de réutilisation. Charge à vous de proposer une conception Orientée Objet adaptée et de bien définir les arguments de ces méthodes. Une fois votre classe programmée, le code du listing 4 devra pouvoir être exécuté.

Listing 4 – code d'utilisation de la classe Board

```

1 board1 = Board()
2 board1.put('a1', Queen())
3 board1.put('b3', Rook())
4 board1.has('a2') # False
5 queen = board1.get('a1') # Queen

```

Les problèmes similaires au problèmes des huit dames

On considère la classe **Problem8Queens** qui représente le problème des 8 dames. Cette classe a les méthodes suivantes :

- `check_solution8queens(self, board)` qui permet de vérifier que la solution board est valide.
- `propose_solution(self, board)` qui permet à un joueur de proposer une solution au problème. Cette méthode fait appel à la méthode `check_solution8queens(self, board)` pour vérifier la solution proposée. De plus, elle mémorise dans un tableau toutes les solutions proposées par les joueurs, que celles-ci soient valides ou non.
- `statistiques(self)` qui écrit un rapport sur la sortie standard. Ce rapport précise combien de solutions ont été proposées et combien sont valides.

Question 8. Programmer la classe **Problem8Queens**. Vous prendrez soin d'exploiter les méthodes de la classe **Board** et les méthodes des classes représentant les pièces. Une fois votre classe programmée, le code du listing 5 devra pouvoir être exécuté.

Listing 5 – code d'utilisation de la classe Problem8Queens

```

1 board1 = Board()
2 board1.put('a1', Queen())
3 board1.put('b3', Queen())
4 problem = Problem8Queens()
5 problem.propose_solution(board1)
6 problem.statistiques() # affiche sur la sortie standard "1 solution, 0 valide"

```

On souhaite maintenant proposer différents types de problèmes : le problème des huit dames, le problème des huit tours, le problème des 4 fous et des 4 tours, etc.

La classe **ProblemFactory** propose une méthode *create_problem* permettant de créer des problèmes de différents types (8 dames, 8 fous, etc.). A noter qu'il est également possible créer des problèmes avec une combinaison de différents types de pièces (exemple : 4 fous et 4 tours).

Question 9. Décrire la classe **ProblemFactory** (sans la programmer en Python) en prenant soin de bien définir les paramètres de la méthode *create_problem* permettant de créer des problèmes de différents types. Vous expliquerez comment cette méthode permet de créer à minima le problème des 8 dames, le problème des 8 tours et le problème des 4 fous et des 4 tours. Vous décrirez également le lien avec la classe **Problem8Queens** et les modifications qu'il faut y apporter. Enfin vous décrirez les qualités de votre conception en termes de réutilisation de code.

Résolution de problème

On considère qu'un algorithme résout un problème du type problème des 8 dames s'il retourne toutes les solutions à ce problème. De manière générique, on appelle *solve* les fonctions qui implémentent un tel algorithme.

Question 10. Programmer une fonction Python qui : (1) génère une instance de la classe **Board** contenant 8 dames placées aléatoirement, (2) vérifie que cette instance est une solution au problème des 8 dames en appelant la méthode *check_solution8queens(self,board)* d'un objet instance de la classe **Problem8Queens**, et (3) s'arrête lorsqu'une solution valide a été trouvée en imprimant sur la sortie écran le résultat trouvé. Vous discuterez de la qualité de la solution consistant à exploiter cette fonction pour proposer une fonction *solve_random* qui résout le problème des 8 dames.

Question 11. Programmer une fonction Python *solve_walker* qui parcourt l'espace des **board** contenant 8 dames et qui s'arrête dès qu'ont été trouvées x **board** solutions au problème des 8 dames ($0 < x \leq 92$) en imprimant sur la sortie écran chaque solution trouvée. Sachant qu'il existe 92 solutions au problème des 8 dames, on peut dire que cette fonction propose un algorithme qui résout le problème des 8 dames. Est-ce que la connaissance du nombre de solutions (92) est nécessaire ?

Question 12. Programmer une fonction *solve_matrix* qui résout le problème des 8 dames en exploitant les matrices de *permutation*. En effet, comme les dames attaquent le long des lignes, des colonnes et des diagonales, cela équivaut à une matrice de permutation d'ordre 8 dans laquelle la somme de chaque diagonale est au plus égale à 1. On rappelle qu'une matrice de *permutation*

est une matrice carrée qui vérifie les propriétés suivantes : les coefficients sont 0 ou 1 ; il y a un et un seul 1 par ligne ; il y a un et un seul 1 par colonne.

Question 13. À ce stade, nous avons au moins trois implémentations candidates pour calculer l'ensemble des solutions du problème des huit dames : *solve_walker*, *solve_random*, et *solve_matrix*. Ces implémentations ont des qualités différentes mais en toute rigueur, l'ensemble des solutions retourné par les trois implémentations devrait donner le même résultat. Montrer en Python comment cette propriété peut être automatiquement vérifiée avec un programme.

Restructuration et difficulté des problèmes

On souhaite que n'importe quel développeur puisse proposer sa fonction *solve*. L'idée étant de pouvoir disposer de plusieurs fonction *solve* proposées par des développeurs différents.

Question 14. Décrire la restructuration de votre application permettant à un développeur d'intégrer sa fonction *solve* dans votre conception. Vous devez prendre soin de bien décrire toutes les modifications à effectuer sur le code des classes existantes. Vous expliquerez aussi ce que doit faire le développeur pour intégrer sa fonction. Enfin, vous montrerez comment choisir une fonction *solve* qui a été proposée, et comment l'exécuter pour trouver toutes les solutions à un problème.

Question 15. On souhaite générer des problèmes d'échecs difficiles. La notion de difficulté est définie ainsi : un problème d'échecs est difficile si (1) les trois implémentations précédentes en charge de trouver toutes les solutions mettent plus de 10 minutes ; et si (2) le nombre de solutions est inférieur à 10. Le problème des huit dames, par exemple, n'est pas difficile (car le nombre de solutions, 92, est supérieur à 10). Proposer un algorithme pour générer et trouver de tels problèmes difficiles. Expliquer comment cet algorithme peut s'intégrer dans la conception des questions précédentes (classe **ProblemFactory**, fonction *solve*, etc.)

Pour mesurer le temps d'exécution d'un programme en Python, vous appuierez sur le module *time*. Le listing 6 illustre le fonctionnement de *time*.

Listing 6 – code d'utilisation du module time

```

1 import time
2 start = time.time() # avant l'appel d'un programme
3 # execution du programme
4 # ...
5 end = time.time() # fin du programme
6 duration = end - start # la durée de l'exécution en seconde (de type float)

```

Partie d'échecs

On souhaite maintenant étendre notre conception et implémentation précédente pour permettre de jouer une vraie partie d'échecs avec notre application.

Question 16. Modifier votre code pour prendre en compte la couleur des pièces. Vous ne recopierez pas tout votre code et vous décrirez uniquement le code modifié.

Question 17. Dans la classe **Board**, programmer la méthode *move(self, from, to)* qui déplace la pièce de la case **from** vers la case **to**. Cette méthode vérifiera que le déplacement est légal (via la méthode *legal_move* des pièces) et qu'il n'y a pas de pièces entre la case **from** et la case **to** (pour rappel on ne considère pas le cavalier et le pion). La méthode vérifiera aussi que la case d'arrivée ne doit pas contenir une pièce de la même couleur que la pièce déplacée. Enfin, il ne devra pas être possible de déplacer deux fois de suite des pièces de la même couleur. Vous veillerez à ce que le code de la méthode *move* réutilise un maximum de code existant de la classe **Board** et des classes des pièces.

Nous considérons maintenant la classe **Game** qui représente une partie d'échecs. A l'initialisation, une partie contient un **Board** avec les pièces en position initiale. Il sera alors possible de jouer le premier coup des blancs. Puis, alternativement, il sera possible de jouer les coups des noirs et des blancs jusqu'au mat.

Question 18. Proposer une conception générale de la classe **Game** permettant aux joueurs de déplacer les pièces jusqu'au mat. Certaines particularités du jeu d'échecs (prise en passant, pat, fin de partie, etc.) non décrites dans l'extrait (cf figure 2, page 4) ne doivent pas être considérées ici. On ne détectera donc pas le mat. Donner les signatures des méthodes que vous jugez nécessaires (pas le code) pour que les joueurs puissent jouer une partie. Préciser par du texte les propriétés de cette classe (sans donner le code de la méthode *__init__*).

Question 19. Illustrer les 4 premiers coups échangés entre deux joueurs (2 coups chacun) à l'aide d'un diagramme décrivant la séquence des échanges de messages entre les objets de l'application. L'objectif est d'illustrer graphiquement combien d'objets participent à ces échanges, comment ces objets sont créés et reliés entre eux, et dans quel ordre se font les échanges. Votre diagramme devra alors faire apparaître graphiquement : tous les objets impliqués, tous les messages échangés, et l'ordre de ces messages

Le roque

On veut maintenant supporter et implémenter le *roque*. Des explications, issues et adaptées de Wikipedia, sont données dans l'encadré ci-dessous.

Question 20. Implémenter une fonction en Python qui implémente le mouvement du roque en vérifiant préalablement si le roque est possible. On fera l'hypothèse que le joueur indiquera la case de départ du roi et la case d'arrivée (le déplacement de la tour étant déduit). Expliquer comment vous avez organisé votre code notamment pour vérifier la possibilité du roque.

Le roque est un déplacement spécial du roi et d'une des tours au jeu d'échecs. Le roque permet, en un seul coup, de mettre le roi à l'abri tout en centralisant une tour, ce qui permet par la même occasion de mobiliser rapidement cette dernière. Il s'agit du seul coup légal permettant de déplacer deux pièces, sans respecter le déplacement classique du roi et de la tour de surcroît. On distingue le petit roque et le grand roque. Dans le cas du petit roque, le roi blanc se déplace de deux cases en direction de la tour, et la tour se place immédiatement à gauche du roi. La figure 5 donne une illustration.

Le grand roque s'effectue de la même manière que le petit roque : le roi avance de deux cases en direction de la tour (en a1 ou en a8), et cette dernière se place de l'autre côté du roi. La figure 6 illustre le principe. Une particularité importante du roque est que plusieurs conditions doivent être respectées :

- toutes les cases qui séparent le roi de la tour doivent être vides. Par conséquent, il n'est pas permis de prendre une pièce adverse en roquant ;
- ni le roi, ni la tour ne doivent avoir quitté leur position initiale. Par conséquent, le roi et la tour doivent être dans la première rangée du joueur ;
- chaque camp ne peut roquer qu'une seule fois par partie ; en revanche, le roi peut déjà avoir subi des échecs, s'il s'est soustrait à ceux-ci sans se déplacer.
- aucune des cases (de départ, de passage ou d'arrivée) par lesquelles transite le roi lors du roque ne doit être contrôlée par une pièce adverse. Par conséquent, le roque n'est pas possible si le roi est en échec.

FIGURE 4 – Explications du roque



FIGURE 5 – Petit roque (avec les pièces blanches)



FIGURE 6 – Grand roque (avec les pièces blanches)

Problème des mats en N coups

L'application **ChessMate** ne porte pas bien son nom puisque, en l'état, les problèmes d'échecs et mat ne sont pas supportés. On se propose d'y remédier.

La classe **MatInNMoves** représente un problème de type mat en N coups. Un tel problème est défini par une position initiale des pièces sur l'échiquier, une couleur (blanc ou noir) pour laquelle le problème est posé, et un nombre de coups maximum pour que cette couleur mette l'autre couleur en position de mat (c'est-à-dire dans l'impossibilité d'empêcher la prise de son roi).

Question 21. Proposer les méthodes de cette classe (pas le code) permettant à un utilisateur d'essayer de résoudre un tel problème. A noter que dans le cadre spécifique des mats en N coups, c'est la même entité qui manipule à la fois le joueur avec les pièces blanches et avec les pièces noires. Vous ferez en sorte que la classe **MatInMoves** réutilise au maximum la classe **Game**, et vous expliquerez la façon dont la vérification du problème se fait.

Réflexion sur la conception

Question 22. Il existe différentes façons de représenter l'échiquier dans la classe **Board**. Cela peut se faire notamment à l'aide d'un tableau comme c'est le cas dans le code donné du listing 1. Proposer une autre façon de représenter l'échiquier et discuter de ses avantages et de ses inconvénients (en espace et en temps).

Question 23. On souhaite généraliser le problème des huit dames (et uniquement ce problème) et faire en sorte que l'échiquier soit de $N \times N$ cases (N étant un nombre positif donné lors de la création du Board). Discuter de l'impact de cette généralisation sur la conception réalisée jusqu'à présent. Vous préciserez quels sont les impacts sur les classes et sur leurs méthodes.

Partie II. Web

On souhaite réaliser une interface graphique HTML, CSS et JavaScript permettant de jouer à des problèmes échiquéens.

L'application Web est composée de deux pages. La première page permet au joueur de sélectionner le problème. L'interface voulue est présentée par la figure 7.

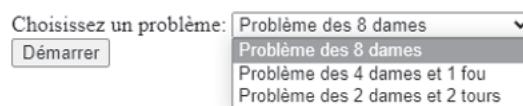


FIGURE 7 – Première page de l'application Web

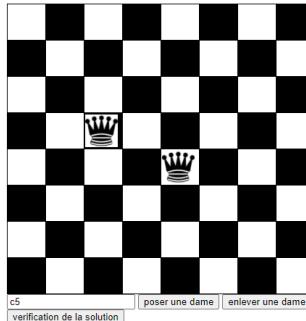


FIGURE 8 – Deuxième page de l'application Web pour le problème des 8 dames

La deuxième page permet de jouer (résoudre le problème proposé). La figure 8 présente l'écran pour tenter de résoudre le problème des 8 dames. On y voit l'échiquier avec ses 64 cases. Une première rangée de boutons permet de saisir les coordonnées d'une case (sur l'exemple : 'c5' et 'e4'), de déposer ou d'enlever une dame. Enfin, un dernier bouton permet de demander la vérification de la solution qui sera réalisée au niveau du serveur web. Le code HTML de la deuxième page web vous est partiellement donné pour le problème des 8 dames (voir listing 7).

Listing 7 – page HTML du problème des 8 dames

```

1 <html>
2 <head>
3   <title>Chess Board</title>
4   <style>
5     table {
6       border-collapse: collapse;
7       border: 1px solid black;
8     }
9     td {
10       width: 50px;
11       height: 50px;
12       border: 1px solid black;
13     }
14     img {
15       width: 45px;
16       height: 45px;
17     }
18     .black {
19       background-color: black;
20     }
21     .white {
22       background-color: white;
23     }
24   </style>
25 </head>
26 <body>
27   <table id="board">
28   </table>
29   <div>
30     <input type="text" id="square" value="e4" />
31     <input type="button" id="putQueen" value="poser une dame" />
32     <input type="button" id="removeQueen" value="enlever une dame" />
33   </div>
34   <div>
35     <input type="button" id="solve" value="verification de la solution" />
36   </div>
37 </body>
38 </html>
```

Question 24. Programmer la première page. Celle-ci contient un formulaire permettant au joueur de choisir le type de problème qu'il veut résoudre. Ce formulaire soumet une requête HTTP GET vers "/choose_problem". Il contient un seul champ nommé "problem". Vous proposerez une programmation en HTML pur de l'ensemble de cette page (i.e., sans JavaScript ou CSS).

Question 25. Dans la deuxième page, programmer un script JavaScript permettant d'afficher un échiquier à l'ouverture de la page. L'échiquier est un tableau composé de 8 lignes (balises TR) et de 64 cellules en tout (balises TD). Vous ferez en sorte que chaque cellule utilise le style proposé pour la couleur des cases. De plus, chaque cellule doit avoir un identifiant unique. Votre code sera inséré dans la page HTML de la deuxième page web à l'aide de la balise proposée dans le listing 8. Les premières lignes du code JavaScript sont déjà écrites, il vous reste à écrire le code pour afficher l'échiquier.

Listing 8 – script pour afficher l'échiquier

```

1 <script>
2   let board = document.getElementById("board");
3   for (let i = 8; i > 0; i--) {
4     //TODO
5   }
6 </script>
```

Question 26. Dans la deuxième page, programmer un script JavaScript permettant de déposer une dame sur la case choisie. Il faudra alors ajouter une image (balise IMG) à la cellule choisie avec l'attribut src de la balise IMG étant égal à "queen.jpg". La première ligne du code JavaScript est déjà écrite (voir listing 9), il vous reste à compléter ce code.

Listing 9 – script pour déposer une dame

```

1 <script>
2   let putQueenButton = document.getElementById("putQueen");
3   //TODO
4 </script>
```

Question 27. Programmer un script JavaScript permettant de retirer une dame de la case choisie. Vous ferez en sorte de minimiser la redondance avec le code de la question précédente. La première ligne du code JavaScript est déjà écrite dans le listing 10, il vous reste à compléter ce code.

Listing 10 – script pour supprimer une dame

```

1 <script>
2   let removeQueenButton = document.getElementById("removeQueen");
3   //TODO
4 </script>
```

La vérification d'une solution s'effectue en envoyant une requête HTTP vers un serveur web (note : vous n'avez pas à proposer d'implémentation du serveur web). La requête de vérification est une requête GET vers /solve8queens et qui a un paramètre dont le nom est **solution** et dont la valeur est une chaîne de caractères décrivant les positions des dames en séparant chaque position par le caractère '-'. Par exemple, si l'adresse du serveur est http://www.serveur-chess.fr, envoyer la requête GET http://www.serveur-chess.fr/solve8queens ?solution=e1-b2-c3-a4-h5-d7-f8-h6 permet de demander la vérification de la solution : ('e1', 'b2', 'c3', 'a4', 'h5', 'd7', 'f8', 'h6').

Si la réponse à la requête est positive (code de retour : 200), le script doit ouvrir une alerte précisant que la solution est valide. Sinon, une alerte précisera que la solution n'est pas valide. On rappelle que la fonction JavaScript "alert(message)" permet de lancer une alerte à l'utilisateur.

Question 28. Programmer ce script JavaScript permettant de soumettre une solution au problème des 8 dames. La première ligne du code JavaScript est déjà écrite dans le listing 11, il vous reste à compléter ce code.

Listing 11 – script pour envoyer une solution

```

1 <script>
2   let solveButton = document.getElementById("solve");
3   //TODO
4 </script>
```

Question 29. Discuter des avantages et inconvénients de réaliser certaines validations de la solution proposée avant d'envoyer une demande de validation au serveur (on considère que le calcul de la vérification est trop long pour se faire intégralement sur le navigateur).

Partie III. Base de données

Pour améliorer l'application **ChessMate** il a été convenu de mettre à disposition une base de données qui contient différents problèmes d'échecs.

Chaque problème a un titre et une description. De plus, certains problèmes sont accompagnés d'une solution déjà trouvée et établie que les utilisateurs pourront consulter. D'autres problèmes, ouverts ('open' en anglais), n'ont pas de solution associée. Enfin, pour certains problèmes, un programme de vérification ('checking_procedure' en anglais) est éventuellement associé et permet de vérifier si la solution proposée au problème est correcte.

Pour commencer la construction de cette base de données, il a été convenu d'utiliser le format **YAML**, qui est un format de représentation de données clé/valeur et arborescent. Un exemple de données en cours de construction est donné ci-dessous :

Listing 12 – exemple de la base

```

1 chessmate:
2   - description: "Le but du problème des huit dames est de placer..."
3     version: "1.0.0"
4     title: "8 queens problem"
5     type: "all_solutions"
6     contact:
7       name: "Elisabeth Harmon"
8       email: "XXX@agreg-info.org"
9     problem: "QQQQQQQQ"
10    checking_procedure: "https://chessmate.org/8queens"
11    solutions: [[8, 5, 20, 25, 42, 59, 54, 39], [0, 14, 20, 31, 33, 43, 53, 58]]
12
13
14  - description: "Votre objectif est de placer 4 dames et 1 fou..."
15    title: "4 Queens + Bishop problem"
16    type: "one_solution"
17    open: true
18    contact:
19      name: "Judit Polkarov"
```

```

20      email: "YYYY@agreg-info.org"
21      problem: "QQQQB"
22      checking_procedure: "https://chessmate.org/4QueensBishop"
23
24      - description: "Votre objectif est de placer 2 dames et 2 tours sur un plateau 6×
25          par 6..."
26          version: "0.1"
27          title: "2 Queens + 2 Rooks problem"
28          type: "one_solution"
29          open: true
30          contact:
31              name: "Judit Polkarov"
32              email: "YYYY@agreg-info.org"
33              problem: "QQRR"
34              chessboard: 36
35              checking_procedure: "https://chessmate.org/2QueensRooks36"
36
37      - description: "Mate in 2"
38          version: "0.1"
39          title: "Mate in 2"
40          type: "check_mate"
41          moves: 3
42          turn: "white"
43          contact:
44              name: "Garry Agreg"
45              email: "XXYY@agreg-info.org"
46              problem: "r1b2r1k/ppp3p1/2n2q1N/7Q/2BPp3/2P5/P4PPP/R1B3K1 w -- 1 16"
47              solutions: "Nf7+ Kg8#"
48

```

L'exemple YAML fourni spécifie 4 problèmes. Les trois premiers problèmes sont des "puzzles", avec respectivement le problème des huit dames, le problème des 4 dames et du fou, et le problème des 2 dames et des deux tours sur un échiquier 6*6.

Le quatrième est un problème d'échecs et mat : il faut trouver une série de 3 coups ('moves' en anglais) en commençant par un coup des pièces blanches (cf `turn` et `white`).

La description des puzzles ou les problèmes de mat partagent un certain nombre de caractéristiques, mais exhibent également des informations spécifiques. On peut catégoriser ainsi les informations caractérisant les problèmes :

- *ProblemDescription* : description du problème, version du problème, titre du problème, type de problème (échecs et mat, puzzle), si le problème est ouvert, etc. Un problème a nécessairement une **description** qui est une chaîne de caractères, possiblement une **version** qui est une chaîne de caractères, un titre via **title**. La notion de **turn** n'a aucun sens pour les problèmes de type puzzle et donc n'est pas spécifiée pour les 3 problèmes de l'exemple. Quand **chessboard** n'est pas spécifié, on considère par défaut que l'échiquier est 8*8 et contient 64 cases. Pour les "puzzles", on distingue deux types de problèmes : le cas où il faut trouver une seule solution (**one_solution**), et le cas où il faut trouver toutes les solutions (**all_solutions**). Toujours pour les puzzles, un problème est considéré comme ouvert (**open**) si une ou des solutions ne sont pas connues et donc renseignées. Pour les "échecs et mat", la valeur de **type** est nécessairement égale à **check_mate** ;
- *Author* : des informations sur l'auteur du problème avec son nom (**name**) et son email (**email**) ;

- *Solution* : une ou des solutions si elles existent, spécifié dans ‘solutions‘ avec une chaîne de caractères et une syntaxe bien particulière selon que ce soit un "puzzle" ou un "échecs et mat".

On souhaite maintenant remplacer l'utilisation de **YAML** par un système de gestion de bases de données (SGBD) relationnel.

Autrement dit, nous souhaitons migrer le document YAML dans un SGBD relationnel.

Question 30. Proposer un modèle relationnel avec au moins trois tables : "ProblemDescription", "Author", "Solution". Pour chaque table, donner les noms de colonne ainsi que leurs types.

En plus de la description des tables, vous préciserez vos choix en répondant aux quatre questions ci-dessous :

Question 31. Comment avez-vous représenté l'attribut "type" ?

Question 32. Comment avez-vous représenté l'optionalité de la ‘version‘ d'un problème ?

Question 33. Comment avez-vous organisé les informations spécifiques à un "échecs et mat" et à un "puzzle" ?

Question 34. Dans ce schéma relationnel, quelles sont les clés primaires de chaque table ? Qu'existe-t-il comme clés étrangères ?

Pour illustrer le fonctionnement de votre base de données :

Question 35. On veut ajouter le problème échec et mat "Mate in 2" de l'exemple YAML ci-dessus. Donner le(s) ordre(s) d'ajout nécessaire(s) dans les tables.

En vous appuyant sur vos tables, écrire en SQL :

Question 36. une requête retournant tous les problèmes "ouverts" (correspondant à `open` dans le YAML).

Question 37. une requête retournant des problèmes, de type puzzle, avec uniquement 8 * 8 cases.

Question 38. une requête retournant le nombre de problèmes qui ont des solutions ou qui sont des problèmes de type "échecs et mat".

On souhaite identifier tous les problèmes de type "puzzle" qui mettent en jeu au moins un fou ('Bishop' en anglais).

Question 39. Est-ce possible de le faire uniquement avec une requête SQL ? Justifier.

Pour y parvenir on se propose de le faire en deux temps :

- via une requête SQL qui retourne tous les problèmes de type "puzzle" (excluant de fait les échecs et mat) et en particulier l'information **problem** qui décrit le problème avec une syntaxe spécifique. Par exemple, pour le problème numéro 2 de l'exemple YAML, **problem** : "QQQQB" spécifie qu'il y a 4 dames à positionner (Q pour Queen) et 1 fou à positionner (B pour Bishop) ;
- via un programme Python qui analysera le contenu de 'problem' pour ne garder que les problèmes avec au moins un fou.

Question 40. En vous appuyant sur l'API Python sqlite3 dont le principe de fonctionnement est décrit ci-dessous, programmer une telle solution en Python. Deux modifications de Listing 13 sont à réaliser : (1) adapter la ligne 4 pour écrire la requête SQL ; (2) adapter la ligne 6, possiblement en écrivant plusieurs lignes de code avant ou après, pour analyser le contenu et ne garder que les problèmes qui utilisent au moins un fou.

Les lignes 1, 2 et 3 permettent de se connecter à la base de donnée et n'ont pas d'intérêt majeur. La ligne 4 permet d'itérer sur chaque ligne (**row**) de la requête. La ligne 5 permet d'obtenir un dictionnaire avec clé/valeur de la ligne, la clé étant le nom de la colonne (dans cet exemple fictif : **date**, **type**, **price**, **city**). Ainsi, ligne 6, il est possible de traiter ce dictionnaire et d'analyser le contenu de la ligne (dans cet exemple fictif, c'est une simple impression écran avec **print**).

Listing 13 – API Python sqlite3

```

1 import sqlite3
2 con = sqlite3.connect('example.db') # nom de la base
3 cur = con.cursor()
4 for row in cur.execute('SELECT * FROM stocks ORDER BY price'): # requete SQL
5     r = dict(row) # dictionnaire cle/valeur
6     print(r)

```

Le résultat obtenu est :

Listing 14 – Résultat de la requête SQL

```

1 {'date': '2006-01-05', 'type': 'BUY', 'price': 35.14, 'city': 'Rennes'}
2 {'date': '2006-01-08', 'type': 'SELL', 'price': 5.42, 'city': 'Marseille'}
3 {'date': '2016-11-05', 'type': 'BUY', 'price': 18.94, 'city': 'Bordeaux'}

```

Question 41. Discuter des avantages et inconvénients de votre solution, basée sur une requête SQL puis un traitement programmatique, en termes de (1) maintenance et de (2) performance.

Partie IV. Architecture

Question 42. L'analyse des requêtes et de l'audience de **ChessMate** montre que le problème des 8 dames est très populaire et que de nombreux joueurs essayent de résoudre ce problème et proposent régulièrement des solutions. L'unique machine serveur qui vérifie ces solutions est

donc trop petite et ne supporte plus la charge. Proposer une infrastructure réseau permettant de déployer plusieurs serveurs. Cette infrastructure doit répartir les requêtes des joueurs (requêtes HTTP) sur les différents serveurs. Vous expliquerez de façon algorithmique et pratique comment ces requêtes sont réparties (comment orienter les requêtes vers les serveurs) et comment la charge est distribuée entre les serveurs (de manière équitable ? avec des effets de seuil ?).

Question 43. La solution proposée par la question précédente est intéressante mais elle cible essentiellement les pics de charge. En effet les serveurs déployés sont très souvent inexploités, la nuit par exemple lorsqu'aucun joueur ne propose de solutions. Proposer une infrastructure avec des serveurs virtuels. Vous expliquerez de façon algorithmique et pratique comment les serveurs virtuels se partagent la charge et comment ils sont démarrés et éteints. Vous préciserez combien de serveurs virtuels sont déployés (1) au déploiement de l'application et (2) lors d'un pic de charge.

Question 44. Dans l'application **ChessMate**, il est possible vérifier si une solution est bonne. Il est également possible de vérifier si *toutes* les solutions proposées sont bonnes. Plutôt que d'effectuer la vérification en séquentiel, sur une seule machine, solution après solution, on propose de distribuer la vérification des solutions sur plusieurs machines. Si une solution parmi toutes celles proposées n'est pas valide, alors la vérification s'arrête et retourne la solution erronée. Proposer une architecture permettant de distribuer une telle vérification de plusieurs solutions.

Question 45. L'approche distribuée de la question précédente peut potentiellement fournir des bénéfices en termes d'accélération du calcul et ainsi réaliser une vérification plus rapide. Cependant, cette approche a aussi des défauts car elle induit également des coûts computationnels et réseaux. Identifier tous les facteurs qui peuvent soit influer sur les bénéfices soit sur les coûts définis ci-dessus soit sur les deux à la fois. Discuter des compromis à trouver.

Fondements de l'informatique

Les questions de programmation doivent être traitées en langage OCaml. On pourra utiliser toutes les fonctions des modules `Array` et `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max`, `incr` ainsi que les opérateurs comme `@`). L'utilisation d'autres modules est interdite.

Dans l'écriture d'une fonction, on pourra faire appel à des fonctions définies dans les questions précédentes, même si ces dernières n'ont pas été traitées. On pourra également définir des fonctions auxiliaires, mais il faudra alors préciser leurs rôles ainsi que le type et les significations de leurs arguments. Si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction de tester si les hypothèses sont bien satisfaites. Lorsque les choix d'implémentation ne découlent pas directement des spécifications de l'énoncé, il est conseillé de les expliquer.

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et en Computer Modern à chasse fixe du point de vue informatique (par exemple `n`).

Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n))$ où n est la taille de l'argument de l'algorithme, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement sauf mention spécifique dans la question.

Il est attendu des candidates et des candidats des réponses construites. Ils seront aussi évalués sur la précision, le soin et la clarté de la rédaction.

1 Compilation d'expressions vers une machine à pile

1.1 Une machine à pile

On s'intéresse à une machine à pile de type «calculatrice en notation polonaise inverse». Cette machine dispose de 4 instructions qui opèrent sur une pile contenant des valeurs entières :

<code>CONST(<i>n</i>)</code>	empile l'entier <i>n</i>
<code>VAR(<i>X</i>)</code>	empile la valeur de la variable <i>X</i>
<code>ADD</code>	dépile deux entiers n_1 et n_2 , empile leur somme $n_1 + n_2$
<code>MUL</code>	dépile deux entiers n_1 et n_2 , empile leur produit $n_1 \times n_2$

Un *code* pour la machine à pile est une liste d'instructions. L'exécution d'un code consiste à exécuter chaque instruction de la liste en séquence. Un *environnement*, donné en paramètre à la machine, associe une valeur à chaque variable. La machine s'arrête lorsque la dernière instruction a été exécutée. Le résultat de l'exécution est l'entier au sommet de la pile.

Exemple : l'exécution du code `CONST(1); VAR(X); ADD` dans un environnement où *X* vaut 2, et à partir d'une pile vide, effectue les étapes suivantes :

	Instruction exécutée	État de la pile après exécution de l'instruction
Étape 1	<code>CONST(1)</code>	1
Étape 2	<code>VAR(<i>X</i>)</code>	$2 \cdot 1$ (2 au sommet, 1 en dessous)
Étape 3	<code>ADD</code>	3

Le résultat de l'exécution est l'entier 3.

Pour simplifier, on supposera tout au long du sujet que l'exécution d'un code machine ne produit jamais d'erreurs : ni débordements arithmétiques, ni tentative de dépiler un entier depuis une pile vide.

Définition : On définit la *consommation en espace* d'un code comme la hauteur maximale atteinte par la pile pendant l'exécution du code, en partant de la pile vide.

Exemple : le code de l'exemple ci-dessus a une consommation en espace égale à 2 puisque juste avant l'exécution de l'instruction `ADD`, la pile contient deux valeurs, et en tout autre point la pile est vide ou contient une seule valeur.



Question 1 Dans un environnement où X vaut 2 et Y vaut 5, déterminer le résultat de l'exécution du code `VAR(X); CONST(1); ADD; VAR(Y); MUL`. Préciser la consommation en espace de ce code.

Une instruction est représentée en OCaml par le type `instr` suivant :

```
type instr = CONST of int | VAR of string | ADD | MUL
```

Un code est alors représenté par une liste d'éléments de type `instr`. L'environnement est représenté par un objet de type `string -> int`, c'est-à-dire une fonction qui prend en argument une chaîne de caractère correspondant à une variable et renvoie un entier.

Question 2 Écrire une fonction `exec : instr list -> (string -> int) -> int` qui prend en argument un code et un environnement et renvoie le résultat de l'exécution du code à partir d'une pile vide.

Question 3 Écrire une fonction `conso_espace : instr list -> int` qui calcule la consommation en espace d'un code donné en argument.

1.2 Compilation simple des expressions arithmétiques

On s'intéresse aux expressions arithmétiques formées à partir de constantes entières $0, 1, 2, \dots, n, \dots$ et de variables symboliques X, Y, Z, \dots en utilisant les opérateurs $+$ (somme) et \times (produit).

Ces expressions sont représentées par leurs arbres de syntaxe abstraite : des arbres binaires avec, à chaque feuille, une constante ou une variable et, à chaque nœud, un opérateur $+$ ou \times . Par exemple, l'arbre A représenté en figure 1 correspond au produit de X plus 1 et de Y :

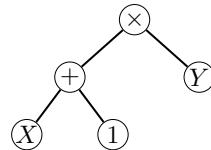


FIGURE 1 – L'expression A

La sémantique de ce langage d'expressions arithmétiques est donnée par les équations suivantes, qui définissent la valeur entière $\llbracket e \rrbracket_\rho$ d'une expression e dans un environnement ρ .

$$\begin{array}{ll} \llbracket \textcircled{n} \rrbracket_\rho = n & \llbracket \textcircled{X} \rrbracket_\rho = \rho(X) \\ \llbracket e_1 \textcircled{+} e_2 \rrbracket_\rho = \llbracket e_1 \rrbracket_\rho + \llbracket e_2 \rrbracket_\rho & \llbracket e_1 \textcircled{\times} e_2 \rrbracket_\rho = \llbracket e_1 \rrbracket_\rho \times \llbracket e_2 \rrbracket_\rho \end{array}$$

On rappelle l'algorithme de compilation classique \mathcal{C} qui, à une expression e représentée par son arbre, associe un code $\mathcal{C}(e)$ de la machine à pile :

$$\begin{array}{ll} \mathcal{C}(\textcircled{n}) = \text{CONST}(n) & \mathcal{C}(\textcircled{X}) = \text{VAR}(X) \\ \mathcal{C}(e_1 \textcircled{+} e_2) = \mathcal{C}(e_1); \mathcal{C}(e_2); \text{ADD} & \mathcal{C}(e_1 \textcircled{\times} e_2) = \mathcal{C}(e_1); \mathcal{C}(e_2); \text{MUL} \end{array}$$

Question 4 Quel code est produit par \mathcal{C} pour l'expression A de la figure 1 ?

Question 5 Soit e une expression, P la pile de la machine, et ρ un environnement. On fait exécuter par la machine le code compilé $\mathcal{C}(e)$ avec ρ comme environnement et P comme pile initiale. Démontrer par récurrence sur la structure de e que la pile finale, lorsque la machine s'arrête, est $\llbracket e \rrbracket_\rho \cdot P$, c'est à dire la pile contenant la valeur $\llbracket e \rrbracket_\rho$ au sommet et les mêmes valeurs que P en dessous.

Question 6 Dans cette question et la question 16 uniquement, on ajoute à notre langage d'expressions la soustraction $e_1 - e_2$, représentée par un arbre de la forme  , ainsi que le calcul de l'opposé $-e$, représenté par un arbre de la forme  . En parallèle, on ajoute à notre machine à pile une instruction OPP qui change le signe de l'entier au sommet de la pile. On veut étendre l'algorithme de compilation \mathcal{C} pour traiter la soustraction et l'opposé. Donner les équations qui définissent $\mathcal{C}(\ominus)$ et $\mathcal{C}(-)$, et les justifier informellement.

1.3 Consommation en espace du code compilé

Question 7 On considère les deux familles d'expressions G_n et D_n pour $n \geq 0$ définies par les récurrences suivantes :

$$\begin{aligned} G_0 &= \textcircled{0} & D_0 &= \textcircled{0} \\ G_n &= \begin{cases} \textcircled{+} & \text{si } n > 0 \\ G_{n-1} & \text{si } n = 0 \end{cases} & D_n &= \begin{cases} \textcircled{+} & \text{si } n > 0 \\ D_{n-1} & \text{si } n = 0 \end{cases} \end{aligned}$$

Montrer que pour tout $n \geq 1$, le code compilé $\mathcal{C}(G_n)$ a une consommation en espace égale à 2, et que pour tout $n \geq 1$ le code compilé $\mathcal{C}(D_n)$ a une consommation en espace égale à $n + 1$.

Question 8 Pour une expression e quelconque, on note $\mathcal{T}(e)$ la consommation en espace du code compilé $\mathcal{C}(e)$. Donner et justifier les relations de récurrence permettant de calculer $\mathcal{T}(\textcircled{n})$, $\mathcal{T}(\textcircled{X})$, $\mathcal{T}(\textcircled{+}_{e_1, e_2})$ et $\mathcal{T}(\textcircled{\times}_{e_1, e_2})$ en fonction de $\mathcal{T}(e_1)$ et $\mathcal{T}(e_2)$.

1.4 Compilation qui minimise la consommation en espace

L'informaticien russe Andreï Erchov propose un algorithme de compilation en 1957 : l'idée centrale est, dans le cas d'une somme ou d'un produit de deux expressions e_1 et e_2 , de traiter en premier la sous-expression e_i qui a le plus grand nombre de Strahler $\mathcal{S}(e_i)$, le nombre de Strahler d'une expression étant défini par la récurrence suivante :

$$\begin{aligned} \mathcal{S}(\textcircled{n}) &= \mathcal{S}(\textcircled{X}) = 1 \\ \mathcal{S}(\textcircled{+}_{e_1, e_2}) &= \mathcal{S}(\textcircled{\times}_{e_1, e_2}) = \begin{cases} 1 + \mathcal{S}(e_1) & \text{si } \mathcal{S}(e_1) = \mathcal{S}(e_2) \\ \max(\mathcal{S}(e_1), \mathcal{S}(e_2)) & \text{si } \mathcal{S}(e_1) \neq \mathcal{S}(e_2) \end{cases} \end{aligned}$$

Ces nombres ont été introduits par le géographe américain Arthur N. Strahler en 1952 pour l'étude des réseaux hydrographiques des cours d'eau.

L'algorithme de compilation d'Erchov \mathcal{E} considéré est alors :

$$\begin{aligned} \mathcal{E}(\textcircled{n}) &= \text{CONST}(n) & \mathcal{E}(\textcircled{X}) &= \text{VAR}(X) \\ \mathcal{E}(\textcircled{+}_{e_1, e_2}) &= \begin{cases} \mathcal{E}(e_1); \mathcal{E}(e_2); \text{ADD} & \text{si } \mathcal{S}(e_1) \geq \mathcal{S}(e_2) \\ \mathcal{E}(e_2); \mathcal{E}(e_1); \text{ADD} & \text{si } \mathcal{S}(e_1) < \mathcal{S}(e_2) \end{cases} & \mathcal{E}(\textcircled{\times}_{e_1, e_2}) &= \begin{cases} \mathcal{E}(e_1); \mathcal{E}(e_2); \text{MUL} & \text{si } \mathcal{S}(e_1) \geq \mathcal{S}(e_2) \\ \mathcal{E}(e_2); \mathcal{E}(e_1); \text{MUL} & \text{si } \mathcal{S}(e_1) < \mathcal{S}(e_2) \end{cases} \end{aligned}$$

Question 9 Montrer l'analogie de la question 5 pour l'algorithme de compilation d'Erchov : si on fait exécuter par la machine le code compilé $\mathcal{E}(e)$ avec ρ comme environnement et P comme pile initiale, la machine s'arrête avec la pile finale $\llbracket e \rrbracket_\rho \cdot P$. On ne demande pas de refaire une démonstration complète, mais juste d'expliquer ce qui change par rapport à la démonstration de la question 5.

Question 10 Calculer les nombres de Strahler $\mathcal{S}(G_n)$ et $\mathcal{S}(D_n)$ pour les expressions G_n et D_n définies à la question 7.

Question 11 En déduire la forme des codes $\mathcal{E}(G_n)$ et $\mathcal{E}(D_n)$ produits par l'algorithme d'Erchov. Quelle est la consommation en espace de ces codes ?

Question 12 Démontrer, pour toute expression e , que la consommation en espace du code compilé $\mathcal{E}(e)$ est égale à $\mathcal{S}(e)$. En d'autres termes, le nombre de Strahler est la consommation en espace du code produit par l'algorithme d'Erchov.

Question 13 On définit la taille $\|e\|$ d'une expression e comme le nombre de nœuds et de feuilles dans son arbre de syntaxe abstraite. Démontrer l'inégalité suivante :

$$\mathcal{S}(e) \leq \log_2(\|e\| + 1)$$

Question 14 En déduire qu'une machine disposant de 4 emplacements de pile peut évaluer n'importe quelle expression contenant au plus 7 additions ou multiplications.

Question 15 Expliquer comment construire une expression e de taille minimale dont le code compilé $\mathcal{E}(e)$ ne peut pas s'exécuter sur une machine disposant de 4 emplacements de pile. Quelle est la taille de cette expression ? Pourquoi cette taille est-elle minimale parmi toutes les expressions e satisfaisant cette propriété ?

Question 16 Comme dans la question 6, on ajoute à notre langage d'expressions l'opposé $\overset{\ominus}{e}$ et la différence $e_1 \overset{-}{e}_2$.

Toujours comme dans la question 6, on ajoute aussi l'instruction OPP à notre machine à pile.

Étendre la définition des nombres de Strahler et l'algorithme d'Erchov aux deux nouvelles formes d'expressions.

On donnera les équations qui définissent les nombres $\mathcal{S}(e_1 \overset{-}{e}_2)$ et $\mathcal{S}(\overset{\ominus}{e})$, ainsi que les codes $\mathcal{E}(e_1 \overset{-}{e}_2)$

et $\mathcal{E}(\overset{\ominus}{e})$.

On représente une expression en OCaml par le type `expr` suivant :

```
type expr =
  | Const of int
  | Var of string
  | Somme of expr * expr
  | Produit of expr * expr
```

Question 17 Écrire une fonction `strahler : expr -> int` qui calcule le nombre de Strahler d'une expression donnée en argument. Cette fonction devra avoir une complexité linéaire en la taille de son argument, et on demande de prouver cette complexité.

Question 18 Écrire une fonction `erchov : expr -> instr list` qui prend en argument une instruction et calcule le code associé selon l'algorithme d'Erchov.

Question 19 Déterminer la complexité temporelle de la fonction `erchov`.

Question 20 Comment modifier la fonction `erchov` pour que sa complexité temporelle soit linéaire en la taille de son argument (si ce n'est pas déjà le cas) ? On ne demande pas de réécrire le code mais de décrire les modifications en français.

2 Compilation d'expressions vers une machine à registres

2.1 Une machine à registres

À la place de la machine à pile de la partie 1, nous allons maintenant cibler une machine à registres. Cette machine dispose de K registres nommés R_1, \dots, R_K , chaque registre pouvant contenir une valeur entière. La machine dispose de 4 instructions :

$\text{MOV}(R_i, n)$	met l'entier n dans le registre R_i
$\text{LOAD}(R_i, X)$	charge la valeur de la variable X depuis la mémoire et la met dans le registre R_i
$\text{ADD}(R_i, R_j, R_k)$	met dans R_i la somme des valeurs des registres R_j et R_k
$\text{MUL}(R_i, R_j, R_k)$	met dans R_i le produit des valeurs des registres R_j et R_k

Un code pour la machine à registres est une liste d'instructions, qui sont exécutées en séquence.

Exemple : l'exécution du code $\text{LOAD}(R_1, X); \text{MOV}(R_2, 1); \text{ADD}(R_1, R_1, R_2)$ a pour effet de mettre la valeur de l'expression $X + 1$ dans le registre R_1 , et l'entier 1 dans le registre R_2 . Les autres registres ne sont pas modifiés.

2.2 Compilation des expressions arithmétiques

On peut adapter les algorithmes de la partie 1 en traitant les K registres de la machine comme une pile de hauteur au plus K . Pour représenter une pile de hauteur $k \leq K$, on stocke le sommet de la pile dans le registre R_k , l'élément suivant de la pile dans le registre R_{k-1} , jusqu'à la base de la pile qui est stockée dans R_1 .

En appliquant cette technique à l'algorithme de compilation simple de la section 1.2, on obtient l'algorithme de compilation \mathcal{C} suivant, qui prend en arguments une expression e et un numéro de registre d :

$$\begin{aligned}\mathcal{C}(\textcircled{n}, d) &= \text{MOV}(R_d, n) \\ \mathcal{C}(\textcircled{X}, d) &= \text{LOAD}(R_d, X) \\ \mathcal{C}(e_1 \xrightarrow{+} e_2, d) &= \mathcal{C}(e_1, d); \mathcal{C}(e_2, d+1); \text{ADD}(R_d, R_d, R_{d+1}) \\ \mathcal{C}(e_1 \xrightarrow{\times} e_2, d) &= \mathcal{C}(e_1, d); \mathcal{C}(e_2, d+1); \text{MUL}(R_d, R_d, R_{d+1})\end{aligned}$$

Question 21 Quel code est produit par $\mathcal{C}(A, 1)$ pour l'expression A représentée figure 1 ?

Question 22 Pour e une expression et d un entier, expliquer informellement l'effet du code produit par $\mathcal{C}(e, d)$ sur les registres de la machine : une fois exécuté ce code, quelle valeur contient le registre R_d ? et le registre R_i pour $i < d$?

Question 23 Donner un exemple d'expression e telle que le code compilé $\mathcal{C}(e, 1)$ est incorrect car utilisant trop de registres : le code utilise le registre R_{K+1} qui n'existe pas.

Question 24 En s'inspirant de l'algorithme d'Erchov, définir un algorithme de compilation \mathcal{E} prenant en entrée e et d et produisant du code de la machine à registre qui :

1. calcule la valeur de l'expression e et met cette valeur dans le registre R_d ;
2. utilise au plus $\mathcal{S}(e)$ registres différents.

En déduire qu'on peut compiler correctement toutes les expressions dont le nombre de Strahler est au plus K .

2.3 Optimalité en nombre de registres utilisés

Dans cette section, nous allons montrer la réciproque de la question précédente : tout code de la machine à registres qui évalue correctement une expression e a besoin d'au moins $\mathcal{S}(e)$ registres différents. Par conséquent, l'algorithme de la question précédente est optimal en nombre de registres utilisés. Ce résultat a été publié par Ravi Sethi et Jeffrey D. Ullman en 1970.

Définition : Soient c un code de la machine à registres (une liste d'instructions), e une expression, et R un registre. On dit que c calcule la valeur de e dans R si les conditions suivantes sont remplies :

- c est de la forme $c_1; I; c_2$ où c_1 et c_2 sont des listes d'instructions et I une seule instruction ;
- aucune instruction de la liste c_2 n'écrit dans le registre R ;
- si e est une constante n , l'instruction I est $\text{MOV}(R, n)$;
- si e est une variable X , l'instruction I est $\text{LOAD}(R, X)$;
- si e est une somme $e_1 \xrightarrow{+} e_2$, l'instruction I est de la forme $\text{ADD}(R, R', R'')$, et de plus le code c_1 calcule la valeur de e_1 dans R' et la valeur de e_2 dans R'' ;

— si e est un produit $e_1 \times e_2$, l'instruction I est de la forme $\text{MUL}(R, R', R'')$, et de plus le code c_1 calcule la valeur de e_1 dans R' et la valeur de e_2 dans R'' .

Exemple : le code $c = \text{MOV}(R_2, 1); \text{LOAD}(R_3, X); \text{ADD}(R_2, R_3, R_2)$ calcule la valeur de la variable X dans le registre R_3 et la valeur de la somme $X + 1$ dans le registre R_2 . Mais il ne calcule pas la valeur de la constante 1 dans R_2 .

Définition : On dit qu'un code c qui calcule la valeur d'une expression e utilise au moins k registres s'il existe un point dans le code c où k registres différents contiennent des résultats intermédiaires du calcul de e , c'est-à-dire les valeurs de k sous-expressions de e .

Exemple : le code c de l'exemple précédent utilise au moins deux registres, car au point qui suit immédiatement l'instruction LOAD , les registres R_2 et R_3 contiennent des résultats intermédiaires du calcul de $X + 1$, à savoir les valeurs de 1 et de X .

Définition : On dit qu'une expression e nécessite au moins k registres si tout code machine c qui calcule la valeur de e (dans un registre quelconque) utilise au moins k registres.

Question 25 En utilisant les résultats de la question 22, montrer que le code $\mathcal{C}(e, d)$ produit par l'algorithme simple de la section 2.2 calcule la valeur de e dans R_d (au sens de la définition ci-dessus).

Question 26 Soient e_1 et e_2 deux expressions telles que e_1 ou e_2 nécessite au moins k registres. Montrer que les expressions $e_1 + e_2$ et $e_1 \times e_2$ nécessitent au moins k registres.

Question 27 Soient e_1 et e_2 deux expressions telles que e_1 et e_2 nécessitent au moins k registres. On suppose que les valeurs de e_1 et de ses sous-expressions sont différentes des valeurs de e_2 et de ses sous-expressions, de sorte qu'aucun résultat intermédiaire ne peut être réutilisé. Montrer que les expressions $e_1 + e_2$ et $e_1 \times e_2$ nécessitent au moins $k + 1$ registres.

Question 28 Déduire des deux questions précédentes que toute expression e sans partage de sous-expressions nécessite au moins $\mathcal{S}(e)$ registres.

3 Compilation d'expressions avec partage

3.1 Introduction du problème

Lors de la compilation d'une expression, une même sous-expression peut apparaître plusieurs fois. Par exemple, dans l'expression e_0 suivante :

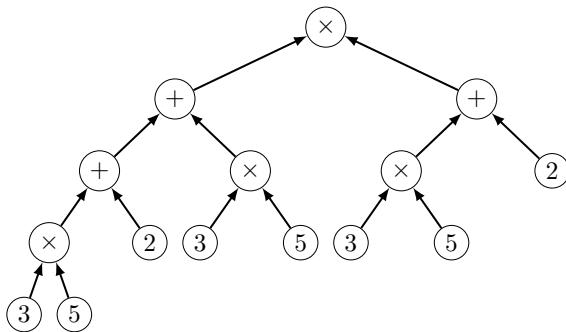
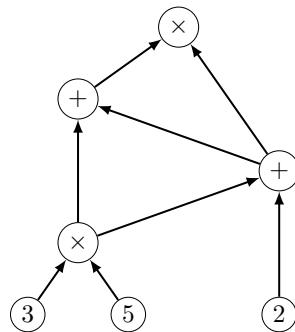
$$(((3 \times 5) + 2) + (3 \times 5)) \times ((3 \times 5) + 2)$$

la sous-expression 3×5 apparaît trois fois, et la sous-expression $(3 \times 5) + 2$ apparaît deux fois. En s'autorisant à ne calculer qu'une seule fois certaines de ces sous-expressions, on peut améliorer l'efficacité du calcul, en temps (nombre d'évaluations de sous-expressions) et en espace (nombre maximal de registres utilisés simultanément). On représente donc dorénavant une expression non plus par un arbre, mais par un graphe orienté sans cycle.

Question 29 En utilisant le graphe H_0 , montrer qu'il existe une suite d'instructions sur une machine à registres qui compile e_0 en utilisant strictement moins de registres et strictement moins d'instructions que la compilation de l'arbre \mathcal{A}_0 .

Définition : On définit un graphe orienté $G = (S, A)$ comme un couple composé d'un ensemble fini non vide de sommets S et un ensemble d'arêtes $A \subset S \times S$.

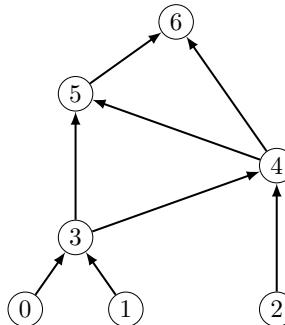
Pour $(s, t) \in A$, on dit que t est un successeur de s et que s est un prédécesseur de t . Pour $(s, t) \in S^2$, un chemin de s à t est une suite finie $c = (s_0, s_1, \dots, s_k)$ d'éléments de S , où $s_0 = s$, $s_k = t$, et pour $i \in \{0, 1, \dots, k-1\}$, s_{i+1} est un successeur de s_i . On dit alors que k est la longueur du chemin c . Si de plus $k \geq 1$ et $s = t$, on dit que c est un cycle.

FIGURE 2 – e_0 sous forme d'un arbre \mathcal{A}_0 FIGURE 3 – e_0 sous forme d'un graphe H_0

On appelle *degré sortant* d'un sommet $s \in S$ son nombre de successeurs et *degré entrant* de s son nombre de prédécesseurs. On dit que s est un *puits* s'il est de degré sortant nul, et une *source* s'il est de degré entrant nul.

Pour la suite de ce problème, on ne considérera que des graphes orientés sans cycle.

On cherche à établir une heuristique pour déterminer l'ordre dans lequel doivent être calculés les sommets d'un graphe correspondant à une expression. Dans la mesure où cet ordre ne dépend pas des opérations, on représente un graphe en OCaml sans se soucier de l'étiquette des sommets.

FIGURE 4 – Représentation de H_0 sans étiquette, en numérotant les sommets

Le type OCaml utilisé est le suivant :

```
type graphe = int list array
```

Dans le détail, pour $n \geq 0$, un graphe $G = (S, A)$ à n sommets sera représenté par un tableau g de taille n , où on assimilera S à $\{0, 1, \dots, n-1\}$, et tel que pour $s \in S$, $g.(s)$ est une liste qui contient tous les successeurs du sommet s (dans un ordre arbitraire et sans doublon). Par exemple, le graphe H_0 précédent peut être créé par la commande :

```
let h0 = [| [3]; [3]; [4]; [5; 4]; [5; 6]; [6]; [] |]
```

On remarque que lors de la compilation, il est intéressant de savoir quels sont les successeurs d'un sommet s donné (quels sont les sommets dont le calcul dépend de s ? À quel moment le registre qui stocke la valeur de s peut-il être libéré?), ainsi que les prédécesseurs d'un sommet s (quels sont les sommets dont le calcul est nécessaire pour calculer s ?). Pour un graphe $G = (S, A)$, on appelle *graphe transposé* de G , noté ${}^t G$, le graphe où on a « retourné » toutes les arêtes, c'est-à-dire le graphe ${}^t G = (S, A')$ où $A' = \{(s, t), (t, s) \in A\}$.

Question 30 Écrire une fonction `transpose : graphe -> graphe` qui prend en argument un graphe $G = (S, A)$ et renvoie le graphe ${}^t G$. La complexité de cette fonction devra être en $\mathcal{O}(|S| + |A|)$ et on demande de prouver cette complexité.

Définition : Pour un graphe $G = (S, A)$ à n sommets, un *ordre topologique* $(s_0, s_1, \dots, s_{n-1})$ est une suite de sommets de taille n et sans doublon telle que pour $i, j \in \{0, 1, \dots, n-1\}^2$, $(s_i, s_j) \in A \Rightarrow i < j$. Autrement dit, un sommet apparaît toujours dans l'ordre avant ses successeurs.

Pour un graphe G à n sommets, on représente en OCaml un ordre topologique $(s_0, s_1, \dots, s_{n-1})$ par un tableau de taille n contenant les valeurs s_0, s_1, \dots, s_{n-1} . Par exemple, le tableau suivant représente un ordre topologique de H_0 :

```
let topo = [|2; 1; 0; 3; 4; 5; 6|]
```

Question 31 On considère la fonction `est_ordre : graphe -> int array -> bool` suivante, qui prend en argument un graphe `g` correspondant à un graphe à n sommets et un tableau `topo` de taille n contenant tous les entiers de $\{0, 1, \dots, n-1\}$ sans doublon.

```
let est_ordre g topo =
  let n = Array.length topo in
  let rang = Array.make n (-1) in
  for i = 0 to n - 1 do
    rang.(topo.(i)) <- i
  done;
  let rec verif_aretes s lst =
    match lst with
    | []      -> s = n - 1 || verif_aretes (s + 1) g.(s + 1)
    | t :: q -> rang.(s) < rang.(t) && verif_aretes s q in
  verif_aretes 0 g.(0)
```

On cherche à montrer que cette fonction renvoie `true` si `topo` est un ordre topologique valide pour `g` et `false` sinon. Expliquer et caractériser par une assertion logique ce que contient le tableau `rang` à la fin de la boucle `for`. Donner une précondition nécessaire et suffisante pour que l'appel `verif_aretes s lst` renvoie `true`. Conclure que la fonction `est_ordre` a le comportement attendu.

Pour un ordre topologique donné, on compile un graphe de la manière suivante :

- les sommets du graphe sont calculés dans l'ordre topologique ;
- lorsqu'un sommet s_j est calculé :
 - pour chaque prédécesseur s_i de s_j ($i < j$), si s_j est le dernier successeur de s_i apparaissant dans l'ordre topologique, le registre stockant s_i peut être libéré au moment du calcul ;
 - on utilise alors un registre (libéré si possible, ou nouveau sinon) pour stocker le résultat de s_j ;
 - si s_j est un puits, on libère ensuite ce registre.

En particulier, chaque sommet ne sera calculé qu'une seule fois.

Question 32 Donner le nombre de registres utilisés pour calculer H_0 selon l'ordre $(2, 1, 0, 3, 4, 5, 6)$. Donner un ordre topologique utilisant strictement moins de registres.

On cherche à déterminer une heuristique permettant de calculer un ordre topologique utilisant peu de registres.

Définition : On redéfinit le *nombre de Strahler* d'un sommet de la manière suivante :

- si s est une source, alors $\mathcal{S}(s) = 1$;
- si s possède un unique prédécesseur t , alors $\mathcal{S}(s) = \mathcal{S}(t)$;
- si s possède deux prédécesseurs t et u , alors $\mathcal{S}(s) = \begin{cases} 1 + \mathcal{S}(t) & \text{si } \mathcal{S}(t) = \mathcal{S}(u) \\ \max(\mathcal{S}(t), \mathcal{S}(u)) & \text{si } \mathcal{S}(t) \neq \mathcal{S}(u) \end{cases}$

Pour les deux questions suivantes, on supposera que les graphes donnés en arguments correspondent à des expressions écrites avec des opérateurs binaires, c'est-à-dire possédant un unique puits et dont tous les sommets ont un degré entrant inférieur ou égal à 2.

Question 33 Écrire une fonction `strahler_graphe : graphe -> int array` qui prend en argument un graphe G et renvoie un tableau `st` tel que pour tout sommet s , `st.(s)` est égal à $\mathcal{S}(s)$. La complexité de cette fonction devra être en $\mathcal{O}(|S| + |A|)$ mais on ne demande pas de le prouver.

L'heuristique proposée pour trouver un ordre topologique particulier, appelé *ordre de Strahler*, pour un sommet s d'un graphe G est la suivante :

- si s est une source, l'ordre de Strahler est (s) ;
- si s possède un unique prédécesseur t , alors l'ordre de Strahler de s est l'ordre de Strahler de t suivi de s ;
- si s possède deux prédécesseurs t et u tels que $S(t) \geq S(u)$, alors l'ordre de Strahler de s est l'ordre de Strahler de t suivi de l'ordre de Strahler de u , auquel on a enlevé les sommets apparaissant dans l'ordre de Strahler de t , suivi de s .

L'ordre de Strahler d'un graphe G est alors l'ordre de Strahler de son unique puits.

Question 34 Écrire une fonction `ordre_strahler : graphe -> int array` qui prend en argument un graphe G et renvoie l'ordre de Strahler de G .

Question 35 En supposant qu'on commence toujours par le sommet de plus petit indice lorsqu'on a le choix entre deux sommets, donner l'ordre de Strahler déterminé par l'heuristique précédente sur le graphe de la figure 5 et le nombre de registres utilisés par cet ordre. Montrer qu'il existe un ordre topologique utilisant strictement moins de registres que l'ordre précédent.

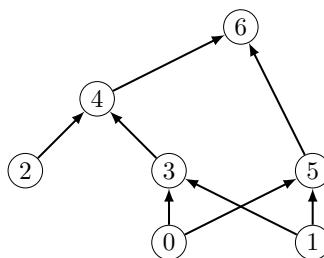


FIGURE 5

3.2 Jeu du marquage

Pour un graphe $G = (S, A)$ de \mathcal{G} , on définit le *jeu du marquage* sur G comme un jeu à un joueur. Les mouvements autorisés dans ce jeu sont les suivants :

- (i) enlever un marqueur d'un sommet ;
- (ii) si tous les prédécesseurs d'un sommet s ont un marqueur, marquer s ;
- (iii) si tous les prédécesseurs d'un sommet s ont un marqueur, déplacer un marqueur de l'un de ces prédécesseurs vers s .

Le jeu commence sans aucun sommet marqué, et l'objectif du jeu est que chaque puits G soit marqué au moins une fois (pas nécessairement simultanément). On remarque qu'un cas particulier du mouvement (ii) est qu'une source peut être marquée à tout moment.

L'intérêt de ce jeu est qu'il représente la compilation d'un graphe orienté sans cycle : la libération d'un registre correspond au mouvement (i), et le calcul d'une expression en fonction de ses sous-expressions directes au mouvement (ii) si un nouveau registre est utilisé pour stocker le résultat, ou au mouvement (iii) si on utilise le registre de l'une des sous-expressions.

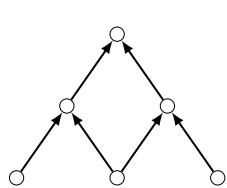
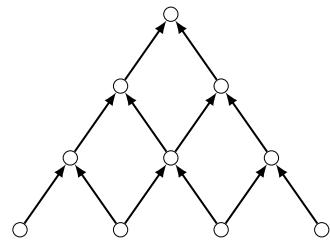
Définition : Une *stratégie* pour un marquage de G est une suite de mouvements permettant de gagner le jeu. Le *temps* de cette stratégie est le nombre de mouvements (ii) et (iii) utilisés, et l'*espace* de cette stratégie est le nombre maximal de marqueurs utilisés simultanément. On notera $M_{\min}(G)$ l'espace minimal d'une stratégie d'un marquage de G , c'est-à-dire le nombre minimal de marqueurs nécessaires pour marquer tous les puits de G .

Question 36 Déterminer $M_{\min}(H_0)$, H_0 étant le graphe représenté en figure 4.

Question 37 Montrer que tout graphe à n sommets possède une stratégie en temps n et en espace n .

Définition : Pour $k > 0$, on définit le *graphe pyramide* P_k par :

- P_1 est le graphe à un seul sommet, sans arête ;
- pour $k > 1$, si P_k est défini, et que $\{s_1, s_2, \dots, s_k\}$ sont les sources de P_k , alors on définit P_{k+1} en rajoutant $k + 1$ sommets $\{t_0, \dots, t_k\}$ et les $2k$ arêtes (t_{i-1}, s_i) et (t_i, s_i) , pour $i \in \{1, 2, \dots, k\}$.

FIGURE 6 – Le graphe P_3 FIGURE 7 – Le graphe P_4

Une représentation des graphes P_3 et P_4 est donnée en figures 6 et 7.

Question 38 Montrer que pour $k > 0$, il existe une stratégie en espace k pour le graphe P_k . Quel est le temps minimal pour une telle stratégie ?

On cherche à montrer dans les deux questions suivantes qu'une telle stratégie est optimale.

On appelle p le puits de P_k . On considère une stratégie en temps T , c'est-à-dire utilisant T mouvements (m_1, \dots, m_T) , et en espace M . On définit l'ensemble suivant :

$$J = \{j \in \{1, 2, \dots, T\} \mid \text{après } m_j, \text{ tous les chemins d'une source de } P_k \text{ à } p \text{ contiennent un sommet marqué}\}$$

Question 39 Montrer que J est non vide. On note j_0 son minimum. Montrer que le mouvement m_{j_0} a marqué une source s_0 de P_k .

Question 40 Montrer que M est supérieur ou égal à la longueur du chemin de s_0 à p . En déduire que $M_{\min}(P_k) = k$ pour $k > 0$.

3.3 Une borne supérieure en espace

L'étude des graphes pyramides a permis de montrer que pour $n > 0$, il existe des graphes à n sommets qui ne possèdent pas de stratégie en espace inférieur à $\sqrt{2n}$. Il s'agit d'une borne inférieure de l'espace minimal requis pour une stratégie de certains graphes à n sommets. En 1976, W. Paul, R. Tarjan et J. Celoni ont amélioré cette borne inférieure en construisant, pour tout $n > 0$, un graphe à n sommets dont toute stratégie nécessite un espace de l'ordre de $\frac{n}{\log n}$.

Dans cette partie, on cherche à montrer que sous certaines hypothèses de degrés entrants, cette borne est optimale. On considère pour la suite uniquement des graphes orientés sans cycles dont les degrés entrants de tous les sommets sont inférieurs ou égaux à 2, en d'autres termes, tels que chaque sommet a zéro, un ou deux prédécesseurs.

On veut montrer que tout graphe à n sommets possède une stratégie en espace $\mathcal{O}\left(\frac{n}{\log n}\right)$.

On note, pour $m > 0$, $a(m)$ le nombre d'arêtes minimal d'un graphe G tel que $M_{\min}(G) = m$. Dans un souci de simplification de la preuve, jusqu'à la question 48 inclusive, on supposera que m est un multiple de 4.

Question 41 Montrer que a est une fonction croissante.

On considère un graphe $G = (S, A)$ tel que $M_{\min}(G) = m$, on note T_1 l'ensemble des sommets qui peuvent être marqués en utilisant $\frac{m}{2}$ marqueurs ou moins, et $T_2 = S \setminus T_1$. Par définition, tous les sommets de T_2 nécessitent strictement plus que $\frac{m}{2}$ marqueurs. On note de plus $B_1 = A \cap (T_1 \times T_1)$, $B_2 = A \cap (T_2 \times T_2)$, et $B = A \setminus (B_1 \cup B_2)$. Enfin, on note $H_1 = (T_1, B_1)$ et $H_2 = (T_2, B_2)$.

Question 42 Montrer que pour toute arête $(s, t) \in B$, $s \in T_1$ et $t \in T_2$.

Question 43 Montrer que pour toute stratégie pour un jeu restreint au graphe H_1 , il existe un sommet de T_1 qui nécessite $\frac{m}{2} - 1$ marqueurs ou plus pour être marqué.

On pourra raisonner par l'absurde en considérant un sommet $s \in T_2$ dont tous les prédecesseurs dans le graphe G sont dans T_1 .

Question 44 On cherche à montrer que pour une stratégie pour un jeu restreint au graphe H_2 , il existe un sommet qui nécessite $\frac{m}{2} - 1$ marqueurs ou plus. Pour ce faire, on raisonne par l'absurde, et on suppose pour cette question que tous les sommets peuvent être marqués dans H_2 en utilisant $\frac{m}{2} - 2$ marqueurs ou moins. Montrer que dans un jeu dans G , tous les sommets de S peuvent être marqués en utilisant $m - 1$ marqueurs ou moins et conclure.

On suppose pour les trois questions suivantes que $|B| < \frac{m}{4}$.

Question 45 Montrer que dans un jeu dans le graphe H_1 , tous les prédecesseurs dans T_1 d'un sommet de T_2 peuvent être marqués simultanément en utilisant $\frac{m}{2} + \frac{m}{4} - 1 \leq \frac{3m}{4}$ marqueurs ou moins.

Question 46 En déduire qu'une stratégie dans le graphe H_2 nécessite $\frac{3m}{4}$ marqueurs ou plus.

Question 47 Montrer qu'il existe un ensemble $C \subset B_2$, de cardinal $\frac{m}{4}$ ou plus, tel qu'une stratégie sur le graphe $(T_2, B_2 \setminus C)$ nécessite $\frac{m}{2}$ marqueurs ou plus.

On revient dans le cas général sans hypothèse sur $|B|$.

Question 48 Montrer que dans tous les cas, $a(m) \geq 2a\left(\frac{m}{2} - 1\right) + \frac{m}{4}$.

On suppose que le résultat de la question 48 est vrai même lorsque m n'est pas un multiple de 4.

Question 49 Montrer que pour tout $m > 0$, $a(m) \geq \frac{m}{8} \log_2 m$. En déduire qu'un graphe qui nécessite m marqueurs ou plus pour être marqué possède $\frac{m}{16} \log_2 m$ sommets ou plus. Conclure que, pour n assez grand, tout graphe à n sommets possède une stratégie en espace au plus $\frac{32n}{\log_2 n}$.

**ECOLE POLYTECHNIQUE - ESPCI
ECOLES NORMALES SUPERIEURES**

CONCOURS D'ADMISSION 2022

**JEUDI 28 AVRIL 2022
16h30 - 18h30**

FILIERES MP-PC-PSI

Epreuve n° 8

INFORMATIQUE B (XELSR)

Durée : 2 heures

*L'utilisation des calculatrices n'est pas
autorisée pour cette épreuve*

Spéléo-logique

*Une phrase courte et claire prend moins de temps à écrire que des pensées confuses...
Utilisez du brouillon !*

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve. Le langage de programmation sera **obligatoirement Python**.

Complexité. La complexité, ou le temps d'exécution, d'une fonction P est le nombre d'opérations élémentaires (addition, multiplication, affectation, test, etc...) nécessaires à l'exécution de P . Lorsque cette complexité dépend de plusieurs paramètres n et m , on dira que P a une complexité en $\mathcal{O}(\phi(n, m))$ lorsqu'il existe trois constantes A , n_0 et m_0 telles que la complexité de P est inférieure ou égale à $A \cdot \phi(n, m)$, pour tout $n \geq n_0$ et $m \geq m_0$. Une fonction prenant une liste en argument sera dite de *complexité linéaire* si elle est de complexité $\mathcal{O}(n)$ où n désigne la longueur de la liste passée en argument.

Lorsqu'il est demandé de donner la complexité d'un programme, vous devrez justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Rappels concernant le langage Python. *L'utilisation de toute fonction Python sur les listes autre que celles mentionnées dans ce paragraphe est interdite.*

Si a désigne une liste en Python de longueur n :

- `len(a)` renvoie la longueur de cette liste, c'est-à-dire le nombre d'éléments qu'elle contient ; la complexité de `len` est en $\mathcal{O}(1)$.
- $a[i]$ désigne le i -ème élément de la liste, où l'indice i est compris entre 0 et `len(a)` - 1 ; la complexité de cette opération est en $\mathcal{O}(1)$.
- `a.append(e)` ajoute l'élément e à la fin de la liste a ; la complexité de cette opération est en $\mathcal{O}(1)$.
- `a.pop()` renvoie la valeur du dernier élément de la liste a et l'élimine de la liste a ; la complexité de cette opération est en $\mathcal{O}(1)$.
- `a.copy()` fait une copie de la liste a ; la complexité de cette opération est en $\mathcal{O}(n)$.

- Si f est une fonction (que l'on supposera de complexité $\mathcal{O}(1)$), la syntaxe `[f(x) for x in a]` permet de créer une nouvelle liste, similaire à la liste b résultant de l'exécution du code suivant :

```
b = []
for x in a:
    b.append(f(x))
```

La complexité de cette création de liste est en $\mathcal{O}(n)$.

L'usage des structures de données d'ensemble `set` ou de dictionnaire `dict` **n'est pas autorisé**.

On fera attention à éviter les effets de bord : sauf lorsque cela est explicitement demandé dans l'énoncé de la question, les fonctions proposées ne devront pas modifier les paramètres qui lui sont passés en argument.

Aucune justification d'un algorithme ou de sa complexité ne devrait excéder 10 lignes.

Le problème. Nous allons déterminer le remplissage d'une grotte lors d'une inondation alimentée par une source d'eau localisée quelque part dans la grotte. La grotte considérée sera bi-dimensionnelle et décrite par le profil de son fond. On supposera définies quatre constantes globales $H = "H"$, $B = "B"$, $G = "G"$ et $D = "D"$, représentant les quatre directions verticales (Haut et Bas) et horizontales (Gauche et Droite). Le profil de la grotte sera donné sous la forme d'une suite de pas horizontaux ou verticaux de longueur 1, encodée sous la forme d'une liste composée des constantes H , B , G et D , pour les quatre directions Haut, Bas, Gauche et Droite. L'origine du profil sera toujours le point $(0,0)$. On considérera toujours que le profil de la grotte se prolonge à gauche et à droite par deux murs verticaux infinis (B^∞ et H^∞). Dans tout le sujet, on supposera également que le profil contient toujours au moins un pas D vers la Droite. La figure 1 donne l'exemple d'une grotte et de son encodage par une liste :

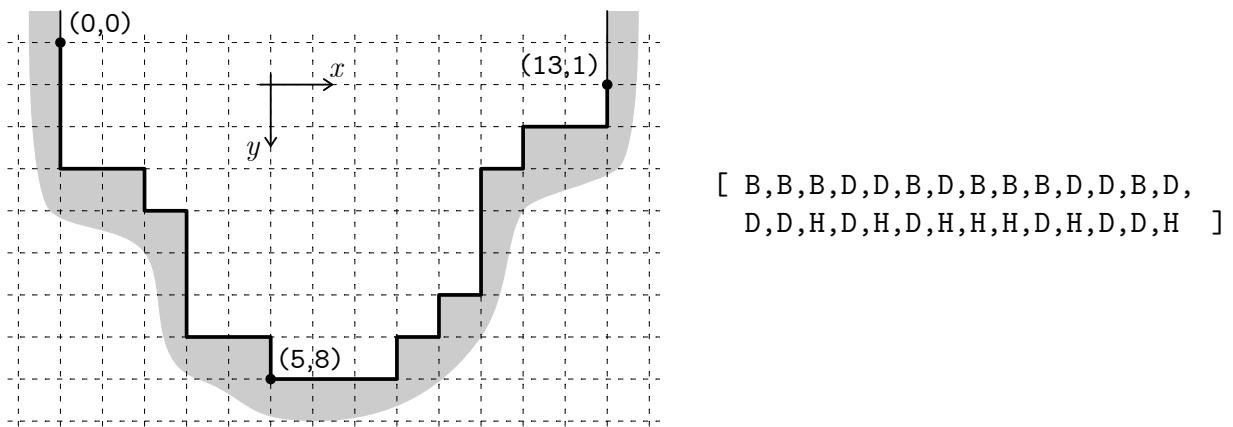


FIGURE 1 – Une grotte et son profil.

Partie I: Validité d'un profil

On dira qu'un profil est *sans rebroussement* s'il ne contient pas de pas qui revienne immédiatement sur le pas précédent, par exemple pas de ..., G, D, Étant donné les conditions aux bords, le profil d'une grotte sans rebroussement ne commence pas par H et ne finit pas par B.

Question 1. Écrire une fonction `est_sans_rebroussement(g)` qui prend en argument la liste `g` décrivant un profil et renvoie `True` si et seulement si le profil est sans rebroussement, et `False` sinon.

Une *vallée* est une grotte dont le profil est sans rebroussement et commence par descendre en ne faisant que des pas vers le Bas ou la Droite, puis remonte en ne faisant que des pas vers le Haut ou la Gauche jusqu'à son point d'arrivée (la direction Droite est en particulier interdite). La grotte de la figure 1 est une vallée.

Question 2. Écrire une fonction `est_une_vallée(g)` qui prend en argument la liste `g` décrivant un profil et renvoie `True` si et seulement si le profil est une vallée, et `False` sinon.

On considère désormais que les axes des x et des y pointent respectivement vers la Droite et le Bas. On rappelle que le profil d'une grotte a pour origine la position $(0,0)$.

Question 3. Écrire une fonction `voisin(x,y,d)` qui prend en arguments deux entiers `x`, `y` et une direction `d` $\in \{H, B, G, D\}$, et renvoie le couple de coordonnées du voisin du point (x,y) dans la direction `d`.

Question 4. Écrire une fonction `liste_des_points(g)` qui prend en argument la liste `g` décrivant un profil et renvoie la liste des coordonnées $[(x_0, y_0), \dots, (x_n, y_n)]$ des points de l'origine à l'arrivée du profil.

On dira qu'un profil est *simple* s'il ne repasse pas par le même point.

Question 5. Écrire une fonction `est_simple(g)` qui prend en argument la liste `g` décrivant un profil et renvoie `True` si et seulement si le profil est simple. Expliciter sa complexité.

Partie II: Vallée

Dans cette partie, nous considérerons que les profils sont toujours de type “vallée”.

Le *fond* d'une vallée est son point le plus à gauche parmi ses points les plus bas. Le fond de la vallée de la figure 2 page suivante a pour coordonnées $(5,8)$.

Question 6. Écrire une fonction `fond(v)` qui renvoie les coordonnées (x, y) du fond de la vallée encodée par la liste des directions `v`.

On considère à présent qu'au temps $t = 0$, une source d'eau située au fond de la vallée, commence à couler avec un *débit constant* et à remplir la vallée. L'objectif de cette partie est de calculer quelle sera la hauteur de l'eau dans la vallée à chaque instant t . On considérera que le *débit de la source est unitaire*, c'est-à-dire d'une unité de surface (un carreau) par unité de temps. La figure 2 indique le niveau de l'eau à différentes dates t dans la vallée de la figure 1.

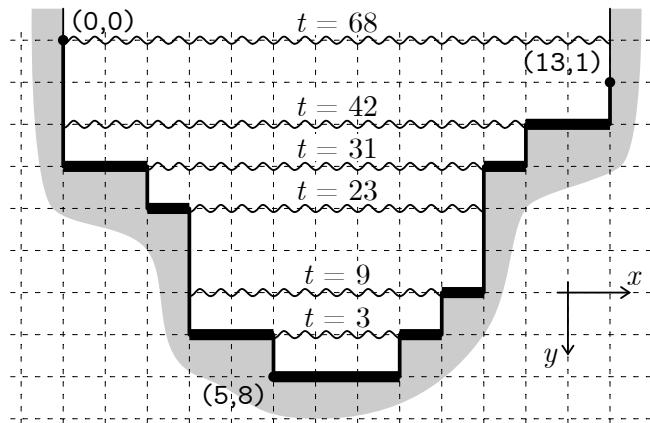


FIGURE 2 – Remplissage d'une vallée.

On appelle *plateau* tout segment horizontal *maximal* du profil de la vallée. Un plateau est défini par le triplet (x_0, x_1, y) où $x_0 < x_1$ sont les abscisses de ses deux extrémités et y est leur ordonnée. La vallée de la figure 2 possède exactement 8 plateaux, indiqués en gras sur la figure : $(0, 2, 3)$, $(2, 3, 4)$, $(3, 5, 7)$, $(5, 8, 8)$, $(8, 9, 7)$, $(9, 10, 6)$, $(10, 11, 3)$, $(11, 13, 2)$.

Question 7. Écrire une fonction `plateaux(v)` de complexité linéaire qui renvoie la liste des triplets correspondant aux plateaux de la vallée encodée par la liste `v`.

Remarquons que si l'on trie les plateaux d'une vallée du plus profond au moins profond (par y décroissants), on obtient une décomposition du volume intérieur de la vallée en rectangles. Ces rectangles sont délimités verticalement par les ordonnées consécutives des plateaux et horizontalement par les abscisses des extrémités des plateaux. La vitesse de montée des eaux est constante à l'intérieur de chaque rectangle et vaut exactement $1/w$ où w est la largeur du rectangle. L'eau met donc un temps hw à remplir chaque rectangle de taille $w \times h$. Dans le cas de la vallée illustrée ci-dessus la liste des tailles (w, h) des rectangles ainsi obtenus est, de bas en haut : $[(3, 1), (6, 1), (7, 2), (8, 1), (11, 1), (13, -1)]$ où la valeur -1 de la dernière hauteur signifie que ce dernier rectangle est de hauteur infinie.

Question 8. Écrire une fonction `decomposition_en_rectangles(v)` de complexité linéaire qui renvoie la liste des tailles des rectangles, triés de bas en haut, décomposant le volume intérieur d'une vallée encodée par la liste `v`. Justifier le bon fonctionnement de votre algorithme.

Question 9. Écrire une fonction `hauteur_de_l_eau(t, v)` qui pour tout nombre flottant $t \geq 0.0$, renvoie la hauteur de l'eau (mesurée depuis le fond) dans une vallée encodée par la liste `v`.

Partie III : Grottes à ciel ouvert

Une grotte est dite *à ciel ouvert* si son profil est simple et ne contient aucun pas vers la Gauche.

Nous dirons que le profil d'une grotte à ciel ouvert est *normalisé* si le point à la fin du profil est situé à la même profondeur que l'origine, 0, et si tous les autres points du profil sont à une profondeur au moins égale à 1.

La figure 3 présente deux profils d'une même grotte à ciel ouvert : l'un normalisé (à droite) et l'autre non (à gauche).

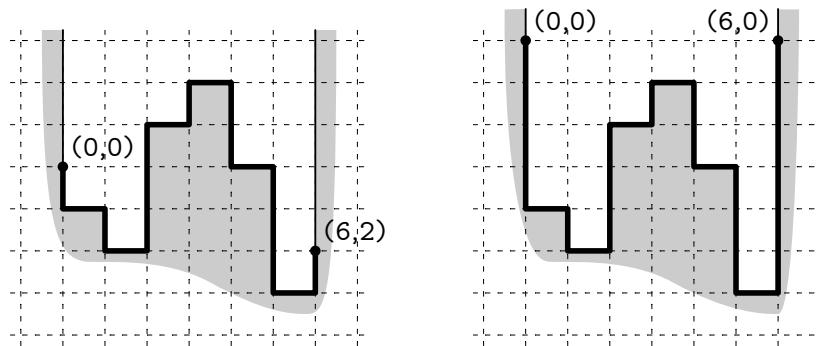


FIGURE 3 – Deux profils, non-normalisé (à gauche) et normalisé (à droite), d'une même grotte à ciel ouvert.

On suppose désormais que les profils seront tous à ciel ouvert et normalisés jusqu'à la fin de cette partie. Remarquons qu'un profil normalisé contient exactement le même nombre de pas B que de pas H. Cette propriété sera utile pour le bon déroulement des algorithmes ci-dessous.

Pour déterminer l'ordre de remplissage de la grotte, nous allons procéder comme précédemment en la découplant en rectangles, sauf que cette fois-ci, pour simplifier, *tous les rectangles de la décomposition seront de hauteur 1* (sauf le dernier qui est de hauteur infinie).

Dans le cas d'une grotte à ciel ouvert, les rectangles qui se remplissent les uns après les autres ne sont plus les uns au-dessus des autres mais organisés hiérarchiquement : chaque rectangle qui n'est pas au fond de la grotte est le “parent” d'un ou plusieurs rectangles “enfants” au-dessous de lui que l'on liste de gauche à droite. La figure 4 page suivante montre la structure hiérarchique parent-enfant pour les 12 rectangles composant la grotte à ciel ouvert décrite par un profil normalisé.

Cette structure hiérarchique sera encodée par 4 listes `origine`, `largeur`, `parent`, `enfants`, de la façon suivante :

- les n rectangles seront numérotés de 0 à $n - 1$;
- `origine[i]` contiendra le couple d'entiers correspondant aux coordonnées du coin inférieur gauche du rectangle n° i ;

[B,B,B,D,H,D,B,B,D,H,H,H,D,B,B,D,H,D,B,B,D,H,D,B,D,H,H,D,B,B,D,H,H,H]

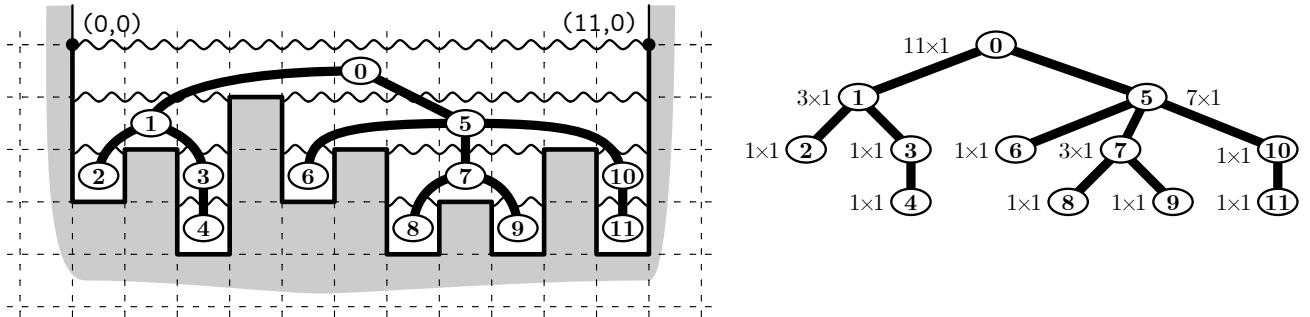


FIGURE 4 – La structure hiérarchique des rectangles.

- `largeur[i]` contiendra la largeur du rectangle n°*i* ;
- `parent[i]` contiendra le numéro du rectangle parent du rectangle n°*i* (ou -1 si c'est le rectangle au sommet de la hiérarchie) ;
- `enfants[i]` contiendra la liste des numéros *de gauche à droite* des rectangles enfants du rectangle n°*i* (cette liste sera vide, [], si ce rectangle n'a pas d'enfants).

Voici les valeurs de ces quatre listes pour la grotte de la figure 4 :

```
origine = [(0,1),(0,2),(0,3),(2,3),(2,4),(4,2),(4,3),(6,3),(6,4),(8,4),(10,3),(10,4)]
largeur = [11, 3, 1, 1, 1, 7, 1, 3, 1, 1, 1, 1]
parent = [-1, 0, 1, 1, 3, 0, 5, 5, 7, 7, 5, 10]
enfants = [[1,5], [2,3], [], [4], [], [6,7,10], [], [8,9], [], [], [11], []]
```

Nous allons dans un premier temps construire cette structure hiérarchique, puis nous l'utiliserons pour calculer le niveau de l'eau dans les différentes parties en fonction de la position de la source et du temps.

Algorithme de décomposition en rectangles. L'algorithme procède en parcourant le profil (normalisé !) de la grotte une seule fois en partant de l'origine. Tout au long de l'algorithme, on maintient une liste `pile` qui contient les numéros des rectangles *ouverts* dont on connaît l'origine mais pas encore la largeur et qui peuvent donc avoir des enfants :

- Au départ : toutes les listes `pile`, `origine`, `largeur`, `parent`, `enfants` sont vides.
- Tout au long de l'algorithme, on maintient les coordonnées (*x,y*) du point où nous en sommes sur le profil.
- À chaque fois que le pas du profil est B : on crée un nouveau rectangle dont on stocke l'origine dans `origine`, dont on met la largeur temporairement à -1 (car on ne la connaît pas encore), dont on initialise la liste des enfants à vide [], et dont le parent est le numéro

[B,D,B,B,D,H,H,D,B,B,D,H,D,B,D,H,H,D,B,D,D,H,D,H]

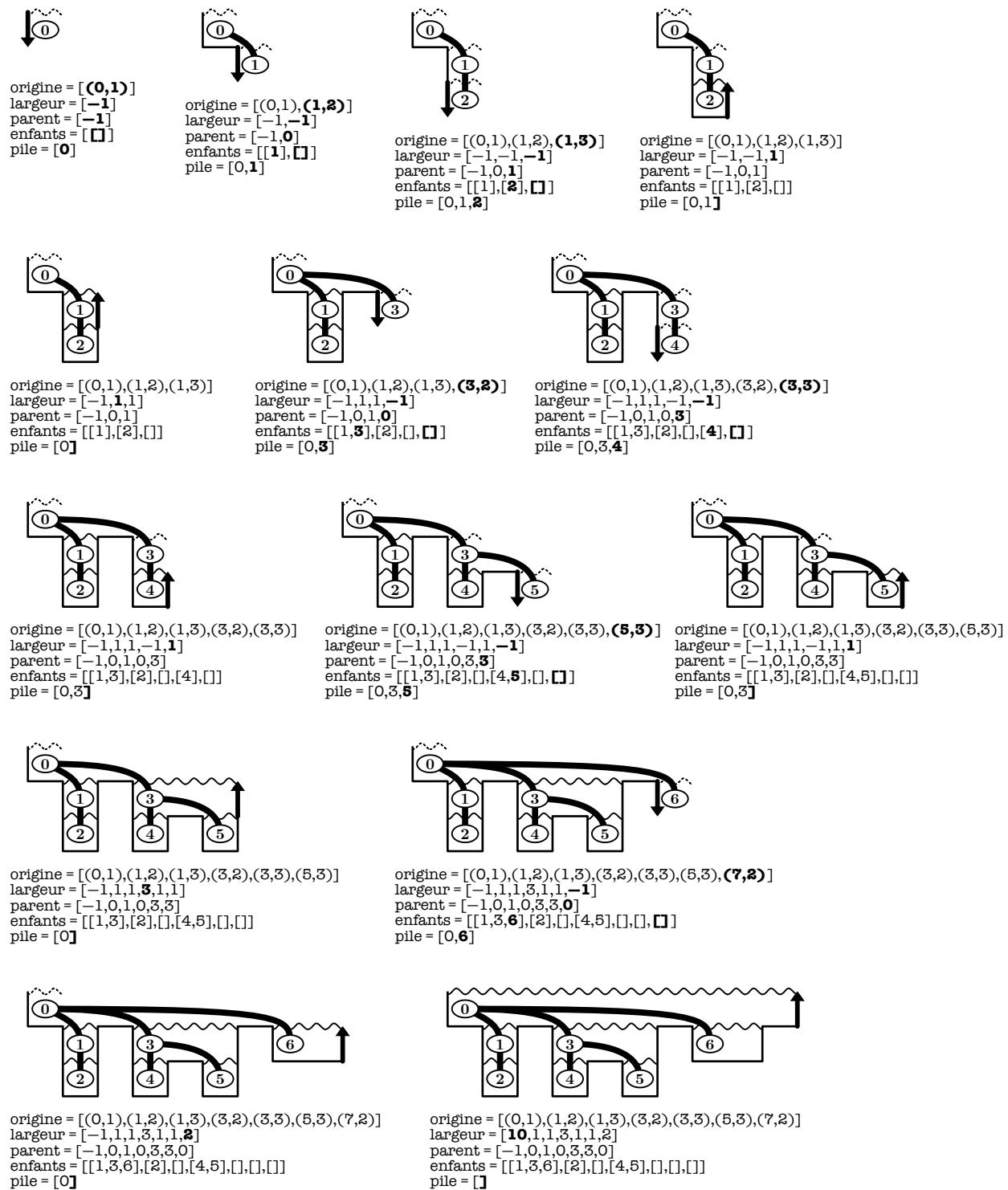


FIGURE 5 – Une exécution de l’algorithme de décomposition en 7 rectangles de la grotte à ciel ouvert en haut : on peut suivre les modifications **en gras** de chacune des listes pile, origine, largeur, parent et enfants à chaque étape du parcours du profil de la grotte.

du rectangle au bout de la liste `pile` (ou `-1` si `pile` est vide) ; on l'ajoute à la liste des enfants de son père, puis on rajoute le numéro de ce rectangle nouvellement “ouvert” au bout de la liste `pile` des rectangles ouverts.

- À chaque fois que le pas du profil est `H` : on “ferme” le rectangle qui se trouve au bout de liste `pile` (qui contient les rectangles actuellement ouverts). Pour cela, on met à jour sa largeur en se basant sur la position actuelle et sur son origine stockée dans `origine` ; puis on retire son numéro de la liste `pile`.

La figure 5 page précédente exécute cet algorithme pas à pas sur un exemple, en montrant l'évolution des listes `pile`, `origine`, `largeur`, `parent` et `enfants` à chaque étape.

Le bon fonctionnement de cet algorithme est garanti par le fait que le profil est normalisé : à chaque ouverture d'un rectangle en suivant un pas `B` (correspondant à son bord gauche), correspond un pas `H` (son bord droit) pour sa fermeture.

Question 10. Écrire une fonction `hierarchie_rectangles(g)` qui renvoie le quadruplet des quatre listes (`origine`, `largeur`, `parent`, `enfants`) décrivant la hiérarchie de rectangles correspondant au profil *normalisé* `g`. Donner sa complexité.

Nous allons désormais exploiter cette structure hiérarchique pour calculer l'ordre de remplissage des rectangles de la grotte. Commençons par observer que cet ordre dépend de la position de la source. Sur la figure 6, la source (symbolisée par \blacktriangle) est placée soit à l'origine (figure de gauche), soit à l'origine du rectangle au milieu (figure de droite).

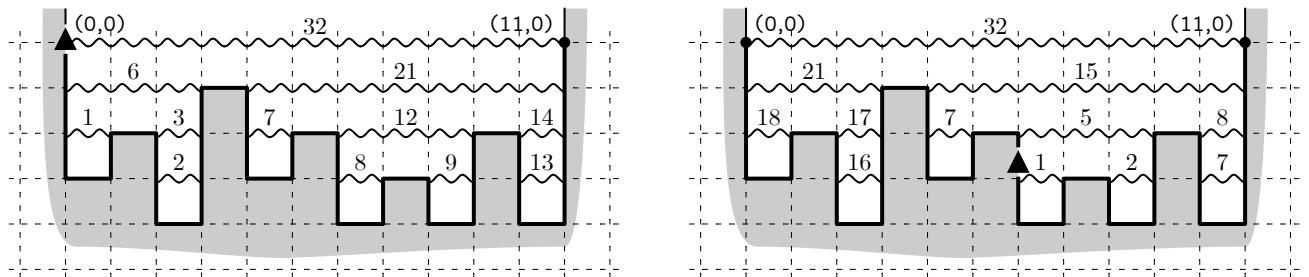


FIGURE 6 – Les dates et ordre de remplissage dépendent de la position de la source (symbolisée par \blacktriangle).

Les dates de remplissage des différents rectangles sont marquées au-dessus de leur bord supérieur. On constate que ces dates sont non seulement différentes, mais aussi que, lorsque la source est “au milieu” de la grotte, alors, plusieurs rectangles peuvent se remplir simultanément, comme c'est le cas des rectangles remplis entre $t = 5$ et $t = 7$ dans la figure de droite. Cette situation n'est cependant pas possible quand la source est située tout à gauche de la grotte, à la position $(0, 0)$ (admis).

Le cas de la source située à l'origine. On supposera que l'eau s'écoule instantanément verticalement et prend donc un temps nul à dévaler les pentes (comme cela a été supposé dans les deux chronologies de la figure 6). On se place dans le cas où la source est située à l'origine.

On admet alors que l'eau remplit les rectangles de gauche à droite, un seul à la fois. On admettra également que chaque rectangle commencera à se remplir une fois que l'ensemble de ses rectangle-enfants seront remplis et que ceux-ci se remplissent l'un après l'autre de gauche à droite.

Question 11. Écrire une fonction `ordre_remplissage_depuis_origine(parent, enfants)` qui prend en entrée les deux listes `parent` et `enfants` décrivant la hiérarchie des rectangles et renvoie la liste des numéros des rectangles dans l'ordre dans lequel ils se remplissent. Donner sa complexité.

Question 12. Écrire une fonction `hauteurs_eau_depuis_origine(t, largeur, parent, enfants)` qui prend en entrée un flottant `t`, et les trois listes `largeur`, `parent` et `enfants`, décrivant la hiérarchie des rectangles d'une grotte à ciel ouvert, et renvoie une liste `hauteur` où `hauteur[i]` est la hauteur d'eau dans le rectangle n° i à l'instant `t` (la hauteur sera donc un flottant entre 0 et 1 sauf pour le dernier rectangle qui est infini et peut donc être rempli à une hauteur arbitrairement grande). Expliciter sa complexité.

Le cas d'une source à une position arbitraire. Comme nous l'avons vu précédemment, lorsque la source est à une position arbitraire, il est possible que plusieurs rectangles se remplissent simultanément : quand un bassin est plein, l'eau s'écoule alors équitablement des deux côtés, comme illustré sur la figure 6 à droite entre les dates $t = 5$ et $t = 7$.

Question 13. Expliquez pourquoi jamais plus de deux rectangles ne se rempliront simultanément. Votre réponse ne devrait pas excéder 5 lignes.

Question 14. Écrire une fonction `volumes_totaux(largeur, parent, enfants)` qui prend en entrée les trois listes `largeur`, `parent` et `enfants` décrivant la hiérarchie des rectangles, et qui renvoie une liste `volume` telle que `volume[i]` est la somme des volumes des rectangles descendants du rectangle n° i , i inclus.

Question 15. Décrire un *algorithme* qui prend en entrée le numéro `source` du rectangle à l'origine duquel est située la source, les trois listes `largeur`, `parent` et `enfants` décrivant la hiérarchie des rectangles, et qui renvoie une liste `hauteur` telle que `hauteur[i]` est la hauteur d'eau présente dans le rectangle n° i à l'instant `t`. On pourra utiliser les procédures définies ci-dessus. On ne demande pas l'écriture d'un programme mais la présentation argumentée d'une solution algorithmique à ce problème.

Justifier le fonctionnement de votre algorithme. Expliciter sa complexité.



A2022 – INFO

**ÉCOLE DES PONTS PARISTECH,
ISAE-SUPAERO, ENSTA PARIS,
TÉLÉCOM PARIS, MINES PARIS,
MINES SAINT-ÉTIENNE, MINES NANCY,
IMT ATLANTIQUE, ENSAE PARIS,
CHIMIE PARISTECH - PSL.**

Concours Mines-Télécom,
Concours Centrale-Supélec (Cycle International).

CONCOURS 2022**ÉPREUVE D'INFORMATIQUE COMMUNE**

Durée de l'épreuve : 2 heures

L'usage de la calculatrice et de tout dispositif électronique est interdit.

Cette épreuve est commune aux candidats des filières MP, PC et PSI.

*Les candidats sont priés de mentionner de façon apparente
sur la première page de la copie :*

INFORMATIQUE COMMUNE

L'énoncé de cette épreuve comporte 12 pages de texte.

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Les sujets sont la propriété du GIP CCMP. Ils sont publiés sous les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 3.0 France. Tout autre usage est soumis à une autorisation préalable du Concours commun Mines Ponts.



Modélisation numérique d'un matériau magnétique

Certains matériaux particuliers peuvent acquérir des états magnétiques qualifiés de paramagnétique et ferromagnétique. Le matériau est dit **paramagnétique** lorsqu'il ne possède pas d'aimantation spontanée, mais acquiert une aimantation sous l'effet d'un champ magnétique extérieur. Il est dit **ferromagnétique** lorsqu'il possède une aimantation même en l'absence de champ magnétique extérieur. Dans ces matériaux, la température T joue un rôle crucial : si T est supérieure à une température particulière T_C , nommée température de Curie, le matériau adopte un état paramagnétique. Dans le cas contraire ($T < T_C$), il adopte un état ferromagnétique. C'est par exemple le cas du fer, pour lequel la transition entre les deux états se produit à $T_C = 1043$ kelvin.

Dans un matériau magnétique, les divers éléments magnétiques (électrons, atomes) possédant un moment magnétique créent une aimantation moyenne à l'intérieur du matériau. Nous admettrons les principaux résultats de la théorie du paramagnétisme.

Ce sujet est constitué de 4 parties. Dans la première, on cherche à obtenir l'aimantation moyenne du matériau en fonction de la température à partir d'une formule théorique connue. Dans la seconde, on recherche dans une base de données les propriétés de matériaux magnétiques. Dans la troisième, on cherche à développer une modélisation microscopique d'un matériau magnétique à deux dimensions pour retrouver ce comportement (modèle d'Ising). Dans la quatrième, on s'intéresse aux domaines magnétiques du matériau (nommés domaines de Weiss).

Une courte documentation de quelques fonctions utiles est disponible à la fin du sujet.

Les candidats sont fortement incités à expliciter brièvement leurs programmes à l'aide de quelques commentaires bien placés.

L'utilisation du module numpy n'est pas autorisée.



Partie I : Transition paramagnétique/ferromagnétique sans champ magnétique extérieur

La théorie des matériaux indique que, dans un matériau ferromagnétique, l'aimantation volumique moyenne du matériau est donnée par :

$$M = N\mu \tanh\left(\frac{\mu B}{k_B T}\right) \quad (1)$$

où N est le nombre d'atomes par unité de volume, B est la valeur du champ magnétique à l'intérieur du matériau, μ le moment magnétique des atomes ou des ions, k_B la constante de Boltzmann, T la température et $\tanh : x \mapsto \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$ la fonction tangente hyperbolique.

On considère une situation sans champ magnétique extérieur. Le champ magnétique local à l'intérieur du matériau ferromagnétique est donc celui créé par le matériau lui-même. On admet que ce champ magnétique est proportionnel à l'aimantation moyenne dans le matériau ($B = \lambda M$), et on obtient alors : $M = N\mu \tanh\left(\frac{\mu\lambda M}{k_B T}\right)$.

En introduisant l'aimantation réduite $m = \frac{M}{N\mu}$ et la température réduite $t = \frac{k_B T}{N\mu^2 \lambda} = \frac{T}{T_C}$, l'équation 1 devient :

$$\boxed{m = \tanh(m/t)} \quad (2)$$

Cette équation d'inconnue m ne possède pas de solution analytique : si on veut connaître une approximation de l'aimantation moyenne dans le matériau, il est donc nécessaire de la résoudre numériquement par une méthode de recherche de zéro.

1. Écrire les instructions nécessaires pour importer exclusivement les fonctions exponentielle (`exp`) et tangente hyperbolique (`tanh`) du module `math`, ainsi que les fonctions `randrange` et `random` du module `random`. Ces fonctions seront ainsi utilisables dans tous les programmes que vous écrirez ultérieurement.
2. A partir de l'équation 2, indiquer une équation `f(x, t) = 0`, d'inconnue `x` que l'on doit résoudre et écrire en Python la définition de la fonction `f` correspondante (paramètres `x` et `t`, valeur renvoyée `f(x, t)`).
3. Écrire une fonction `dicho(f, t, a, b, eps)` qui calcule une valeur approchée à `eps` près du zéro d'une fonction `f(m, t)` de variable `x` et de paramètre `t` fixé sur un intervalle $[a, b]$. On supposera pour simplifier que la fonction dont on recherche le zéro est continue et s'annule une fois et une seule sur l'intervalle $[a, b]$.
4. Établir l'expression de la complexité temporelle asymptotique de la fonction `dicho` en fonction de `a`, `b` et `eps`.

Une étude mathématique simple permet de prouver que l'équation $m = \tanh(m/t)$ n'a de solution $m > 0$ que pour $0 < t < 1$, ce qui revient à dire que le matériau ne possède une aimantation non nulle que pour une température inférieure à la température de Curie. On admet ainsi que si $t \geq 1$ alors $m = 0$. De plus, pour $t < 1$, on admet que la solution $m = 0$ ne doit pas être prise en compte car elle correspond à une solution instable.

5. En utilisant la fonction `dicho`, écrire une fonction `construction_liste_m(t1, t2)` qui construit et retourne une liste de 500 solutions de l'équation (1), pour t variant linéairement de t_1 à t_2 (bornes incluses). On cherchera les valeurs de m à 10^{-6} près avec un intervalle de recherche initial $m \in [0.001, 1]$.

En traçant l'aimantation m en fonction de la température t , on obtient le graphe de la figure 1 permettant de visualiser les domaines ferromagnétique ($t < 1$) et paramagnétique ($t \geq 1$).

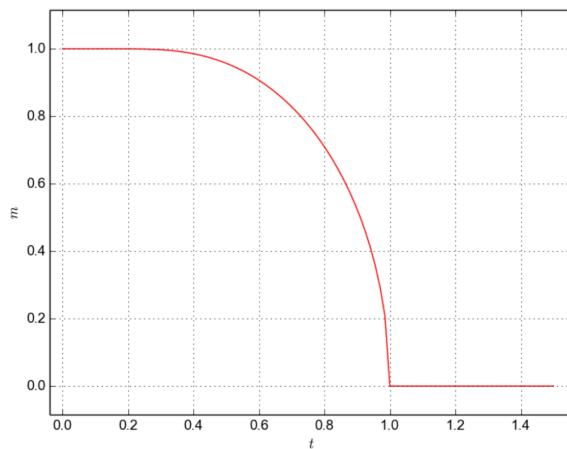


FIGURE 1 – aimantation réduite m en fonction de la température réduite t

Partie II : Recherche dans une base de données de matériaux magnétiques

Il existe des bases de données contenant les propriétés de nombreux matériaux, dont des propriétés magnétiques. Dans cette partie, on donne un modèle simplifié d'une telle base, et on souhaite effectuer quelques requêtes sur celle-ci.

La base de données possède la structure suivante :

- La table **materiaux** contient un champ **id_materiau**, clé primaire de la table de valeur entière, un champ **nom** de type chaîne de caractères pour le nom du matériau et un champ **t_curié** de valeur entière pour la température de Curie du matériau en kelvin.

id_materiau	nom	t_curié
4534	cobalt	1 388
1254	dioxyde de chrome	386
8713	nickel	627
8284	YIG	560
...

- La table **fournisseurs**, contenant un champ **id_fournisseur**, clé primaire de type entier qui précise le code de chaque fournisseur, et un champ **nom_fournisseur** de type chaîne de caractères pour le nom du fournisseur.

id_fournisseur	nom_fournisseur
145	Worldwide Materials
13	Materials Company
...	...

- La table **prix** qui contient un champ **id_prix**, clef primaire de type entier, un champ **id_mat** dont les valeurs sont incluses dans l'ensemble des valeurs de la clé **id_materiau** de la table **materiaux**, un champ **id_four** dont les valeurs sont incluses dans l'ensemble des valeurs de la clé **id_fournisseur** de la table **fournisseurs**, et un champ **prix_kg** de type flottant qui précise le prix au kg que ce fournisseur propose pour ce matériau, en euros. Un fournisseur qui ne propose pas un matériau donné n'a pas d'entrée correspondante dans cette table.

id_prix	id_mat	id_four	prix_kg
1	4567	145	50.40
2	8671	13	1357.30
3	1763	145	52.75
...

Les requêtes demandées dans cette partie sont à écrire en langage SQL.

6. Écrire une requête permettant d'obtenir le nom de tous les matériaux qui ont une température de Curie strictement inférieure à 500 kelvins.

Un client potentiel souhaite acheter 4,5 kilogrammes de nickel (d'identifiant 8713, que l'on pourra utiliser directement dans les requêtes) et sélectionner le fournisseur le moins cher.

7. Écrire une requête permettant d'obtenir les noms de tous les fournisseurs proposant du nickel et le prix proposé par chacun pour 4,5 kilogrammes de nickel.
8. Modifier ou compléter la requête précédente afin d'obtenir le nom du fournisseur de nickel le moins cher ainsi que le prix à payer chez ce fournisseur pour ces 4,5 kilogrammes de nickel. En cas d'égalité du prix optimal entre plusieurs fournisseurs, on obtiendra les noms de tous les fournisseurs possibles.
9. Écrire une requête permettant d'obtenir le nom de tous les matériaux et le prix moyen pour un kilogramme de chacun de ces matériaux (la moyenne étant calculée pour tous les fournisseurs proposant ce matériau), en se limitant aux prix moyens strictement inférieurs à 50 euros par kilogramme.

Partie III : Modèle microscopique d'un matériau magnétique

Pour étudier l'effet du champ magnétique sur un matériau magnétique, on adopte une modélisation microscopique. On modélise les atomes par des sites portant chacun une grandeur physique, nommée *spin*, dont il n'est pas nécessaire de connaître les propriétés.

L'échantillon modélisé est une zone carrée à deux dimensions possédant h *spins* régulièrement répartis dans chaque direction, donc formant une grille carrée de $n = h^2$ *spins*. Chaque *spin* ne possède que deux états *down* ou *up*, ce que l'on modélise par une variable $s_i \in \{-1, +1\}$.

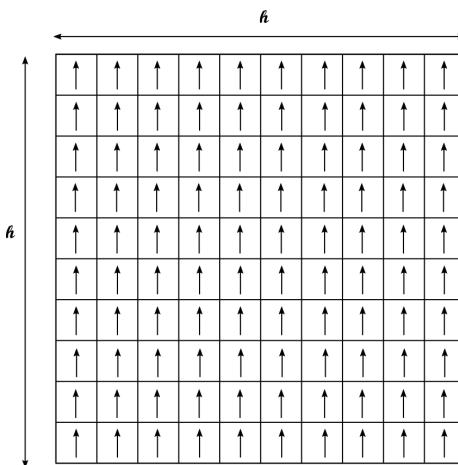


FIGURE 2 – Modèle des *spins* dans un matériau ferromagnétique

Pour implémenter cette configuration de *spins* décrivant l'état microscopique du matériau, on choisit de travailler sur une liste **s**, contenant n entiers, chacun valant -1 ou 1 . On notera le choix d'implémentation adoptée, qui impose de travailler sur **une simple liste de n éléments** pour modéliser une grille de taille $n = h \times h$, dans l'ordre suivant : première ligne puis deuxième ligne , etc.

Un domaine d'aimantation uniforme (cf. figure 2) sera donc représenté par une liste contenant n fois 1 ([1, 1, 1, ..., 1]).

Le début du programme, outre les imports de module Python déjà réalisés à la question 1, est défini par :

```
1 | h = 100
2 | n = h**2
```

ce qui définit deux variables globales utilisables dans tout le programme.

10. Écrire une fonction `initialisation()` renvoyant une liste d'initialisation des domaines contenant n *spins* de valeur 1 comme sur la figure 2.

L'antiferromagnétisme est une propriété de certains milieux magnétiques. Contrairement aux matériaux ferromagnétiques, dans les matériaux antiferromagnétiques, l'interaction d'échange entre les atomes voisins conduit à un alignement antiparallèle des moments magnétiques atomiques (cf. figure 3). L'aimantation totale du matériau est alors nulle (on se limite au cas où h est pair).

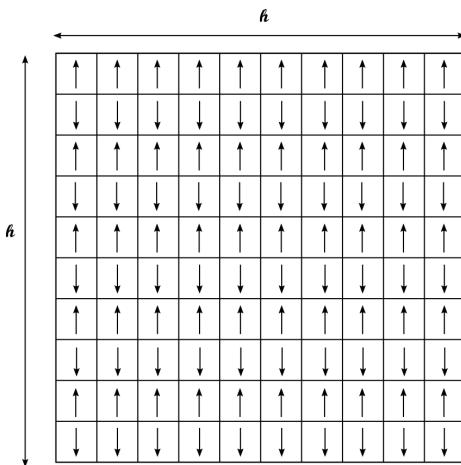


FIGURE 3 – Modèle des *spins* d'un matériau antiferromagnétique

11. Écrire une fonction `initialisation_anti()` renvoyant une liste s d'initialisation des domaines contenant h *spins* en largeur et h en hauteur en alternant les 1 et -1 comme sur la figure 3.
12. Pour afficher l'état global du matériau, il est nécessaire de convertir la liste s utilisée en un tableau de taille $h \times h$ représenté par une liste de listes. Écrire une fonction `repliement(s)` qui prend en argument la liste de *spins* s et qui renvoie une liste de h listes de taille h représentant le domaine.

Attention : dans la suite, l'utilisation de la fonction `repliement` n'est pas autorisée : on travaille exclusivement sur une liste unidimensionnelle s .

Dans la modélisation adoptée (sans champ magnétique extérieur), l'énergie d'une configuration, définie par l'ensemble des valeurs de tous les *spins*, est donnée par :

$$E = -\frac{J}{2} \sum_i \sum_{j \in V_i} s_i s_j \quad (3)$$

avec V_i l'ensemble des voisins du *spin* i .

On suppose que seuls les quatre *spins* situés juste au dessus, en dessous, à gauche et à droite de s_i sont capables d'interagir avec lui.

J est nommée **intégrale d'échange** et modélise l'interaction entre deux *spins* voisins. Pour simplifier, on considérera dans les programmes que $J = 1$.

Malgré le caractère fini de l'échantillon, on peut utiliser une modélisation très utile pour faire comme s'il était infini en utilisant les *conditions aux limites périodiques*. Lorsque l'on considère un *spin* dans la colonne située la plus à droite (*resp.* gauche), il ne possède pas de plus proche voisin à droite (*resp.* gauche) : on convient de lui en affecter un, qui sera situé sur la même ligne complètement à gauche (*resp.* droite) de l'échantillon. De même, le plus proche voisin manquant d'un *spin* situé sur la première (*resp.* dernière) ligne sera situé sur la dernière (*resp.* première) ligne de l'échantillon (voir la figure 4).

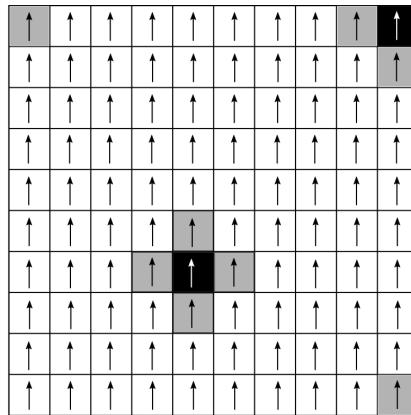


FIGURE 4 – Voisinage d'un *spin* (les voisins des *spins* sur les cases noires sont indiqués en gris)

13. Définir une fonction `liste_voisins(i)` qui renvoie la liste des indices des plus proches voisins du *spin* s_i d'indice i dans la liste `s` (dans l'ordre gauche, droite, dessous, dessus). On pourra utilement utiliser les opérations `%` et `//` de Python, qui renvoient le reste et le quotient de la division euclidienne.
14. Définir la fonction `énergie(s)` qui calcule l'énergie d'une configuration `s` donnée (cf. équation 3).

Pour trouver une configuration stable pour les spins, il faut faire évoluer la liste vers une situation d'équilibre conformément aux principes de la physique statistique. On adopte une méthode probabiliste connue sous le nom de méthode de Monte-Carlo, dont le principe de fonctionnement est le suivant. À chaque étape :

- on choisit un *spin* au hasard dans l'échantillon,
- on calcule la variation d'énergie ΔE qui résulterait d'un changement d'orientation de ce *spin*,
- si $\Delta E \leq 0$, ce *spin* change de signe,
- si $\Delta E > 0$, ce *spin* change de signe avec la probabilité donnée par la loi de Boltzmann :

$$p = \exp\left(\frac{-\Delta E}{k_B T}\right).$$

Dans la suite, ΔE et $k_B T$ seront désignées par les variables `delta_e` et `T` correspondantes dans le programme.

15. Définir une fonction `test_boltzmann(delta_e, T)` qui renvoie `True` si le *spin* change de signe, et `False` sinon.
16. Juste après avoir sélectionné au hasard l'indice `i` d'un *spin* de la liste `s` à basculer éventuellement, pour évaluer l'écart d'énergie `delta_e` entre les deux configurations avant/après, on propose deux solutions sous forme des fonctions `calcul_delta_e1` et `calcul_delta_e2` :

```

1 def calcul_delta_e1(s, i):
2     s2 = s[:]
3     s2[i] = -s[i]
4     delta_e = energie(s2)-energie(s)
5     return delta_e
6
7 def calcul_delta_e2(s, i):
8     delta_e = 0
9     for j in liste_voisins(i):
10         delta_e = delta_e + 2*s[i]*s[j]
11     return delta_e

```

où `s[i]` est le *spin* choisi pour être éventuellement retourné. Indiquer la solution qui vous paraît la plus efficace pour minimiser le temps de calcul en justifiant votre réponse.

17. En utilisant la fonction `test_boltzmann`, définir une fonction `monte_carlo(s, T, n_tests)` qui applique la méthode de Monte-Carlo et qui modifie la liste `s` où l'on a choisi successivement `n_test spins` au hasard, que l'on modifie éventuellement suivant les règles indiquées dans les explications.
18. Écrire la fonction `aimantation_moyenne(n_tests, T)` qui :
 - initialise une liste des *spins* (avec la fonction `initialisation` par exemple),
 - la fait évoluer en effectuant `n_tests` tests de Boltzmann et les inversions éventuelles qui en découlent,
 - calcule et renvoie l'aimantation moyenne de la configuration à la température `T` (définie ici comme la somme des valeurs des *spins* divisée par le nombre total de *spins*).
19. Évaluer la complexité asymptotique de la fonction `aimantation_moyenne(n_tests, T)` en fonction de `n`, nombre de *spins* dans le système, et de `n_tests`.

20. Cette complexité asymptotique serait-elle modifiée si on avait voulu prendre en compte toutes les interactions entre deux *spins* quelconques dans le système, et plus seulement entre les plus proches voisins ? Justifier.

On réalise plusieurs simulations en faisant varier la température T autour de la température de Curie (ici $T_C=2,269$ compte-tenu du choix des valeurs de J et k_B). On représente alors les *spins* orientés vers le haut par une case foncée et les *spins* orientés vers le bas par une case claire. On obtient les résultats de la figure 5.

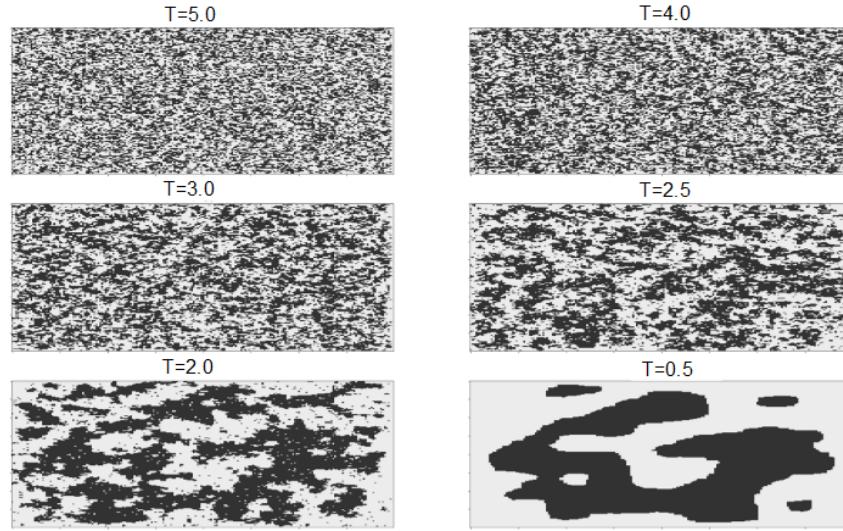


FIGURE 5 – Évolution du domaine en fonction de la température T

21. Indiquer l'influence de l'augmentation de la température sur le comportement du matériau ferromagnétique.

Partie IV : Exploration des domaines de Weiss

Une observation microscopique des matériaux magnétiques nous apprend que les zones magnétiques du matériau sont organisées en domaines, nommés domaines de Weiss. Dans un domaine de Weiss donné, tous les *spins* ont la même valeur.

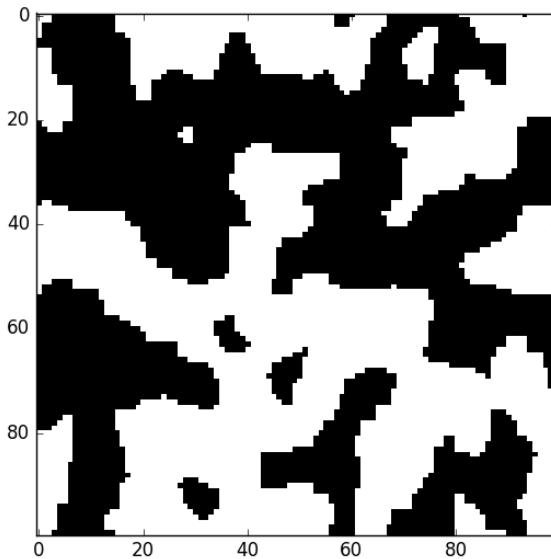


FIGURE 6 – Représentation des domaines de Weiss

On souhaite dans la suite décrire les différents domaines de Weiss afin, par exemple, de les colorier d'une manière différente. Pour cela, on va construire une liste, nommée `weiss`, possédant exactement la même taille que `s` (soit `n`) contenant initialement des `-1` (cette valeur signifiant que le *spin* correspondant n'a encore été affecté à aucun domaine de Weiss).

On souhaite alors écrire une fonction récursive `explorer_voisinage(s, i, weiss, num)` qui, à partir d'une configuration donnée `s`, d'un indice de départ `i` (repérant le *spin* dans `s`), ainsi qu'un entier `num` qui précise le numéro du domaine auquel appartient s_i , construit récursivement la liste `weiss` par effet de bord. Cette fonction doit réaliser les opérations suivantes :

- Pour chaque *spin* voisin du *spin* s_i , elle doit vérifier si les *spins* sont identiques, et s'il n'a pas déjà été affecté à un domaine de Weiss précédemment.
 - Dès qu'un tel *spin* est ajouté, on inscrit son numéro de domaine dans la liste `weiss`, et on explore récursivement son voisinage.
22. Écrire le code de la fonction récursive `explorer_voisinage(s, i, weiss, num)` conforme à la description ci-dessus.

Avec la fonction précédente, la pile de récursion peut devenir de taille très importante dans le cas d'un domaine de grande taille. Afin de mieux contrôler ce parcours, on choisit de l'effectuer avec une structure de pile explicite. Cette pile sera représentée par une liste nommée `pile` sur laquelle on peut ajouter un élément (avec `append`) ou récupérer l'élément du dessus (via `pile.pop()` qui renvoie l'élément sur le dessus de la pile et le retire de la pile). Il s'agit donc, tant qu'il reste des *spins* à explorer, de :

- récupérer l'indice d'un *spin* à explorer dans la pile et le marquer dans la liste `weiss`,
- regarder dans son voisinage si des *spins* possèdent la même valeur et n'ont pas encore été affectés à un domaine, puis ajouter leurs indices dans la pile si c'est le cas.

23. Écrire le code de la fonction itérative `explorer_voisinage_pile(s, i, weiss, num, pile)` conforme à la description ci-dessus.

Enfin, la fonction précédente va permettre de construire la liste `weiss` contenant les numéros des domaines auxquels appartiennent tous les *spins* (le numéro du domaine auquel appartient le *spin* d'indice 0 sera pris à 0, le domaine suivant à 1, etc.).

24. Écrire le code de la fonction `weiss=construire_domaines_weiss(s)` qui construit et renvoie la liste `weiss` contenant le numéro des domaines de Weiss de chaque *spin* du domaine.

L'intérêt de la construction précédente est de disposer d'un marqueur différent pour chaque domaine de Weiss, permettant par exemple de les visualiser dans deux dimensions avec des couleurs différentes, comme dans l'image de la figure 7 :

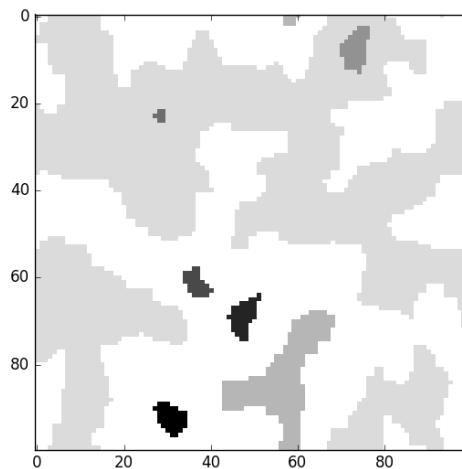


FIGURE 7 – Domaines de Weiss après marquage en nuances de gris

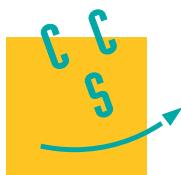
Fin de l'épreuve

Annexe : Documentation sommaire

Les fonctions présentées ci-dessous pourront être utilisées sous réserve de l'import du module Python auquel elles appartiennent.

- Module `math` : les fonctions `exp` et `tanh` permettent de calculer l'exponentielle et la tangente hyperbolique d'un entier ou d'un flottant.
- Module `random` :
 - `randrange(n)` permet de renvoyer un entier aléatoirement choisi parmi $0, 1, 2, \dots, n - 1$
 - `random()` permet de renvoyer un flottant aléatoire entre 0 et 1 suivant une densité de probabilité uniforme.

On admet que ces deux fonctions sont de complexité constante.



Informatique

MP, PC, PSI, TSI

CONCOURS CENTRALE-SUPÉLEC

3 heures

Calculatrice autorisée

2022

Modélisations autour de la Formule 1

Ce sujet s'intéresse à la modélisation de voitures de courses évoluant sur un circuit.

Les deux premières parties consistent à représenter un circuit suivant deux modélisations différentes, à vérifier la cohérence de sa représentation puis à le tracer à l'écran. La troisième partie évalue, en fonction des caractéristiques du circuit, le temps idéal pour effectuer un tour puis une course entière. La dernière partie est dédiée à la gestion des résultats du championnat de formule 1 et à la réalisation de statistiques sur plusieurs années.

Le championnat du monde de formule 1 a été créé en 1950. Chaque année, une vingtaine de courses (appelées grand prix), auxquelles prennent part une vingtaine de pilotes, se courront sur différents circuits et comptent pour ce championnat. Ces nombres ont cependant légèrement varié depuis sa création. Les courses proprement dites sont précédées de séances de « qualifications » qui permettent, en particulier, de définir l'ordre des voitures sur la grille de départ. Chaque course fait environ 300 km, ce qui représente entre 40 et 80 tours de circuit suivant la longueur de celui-ci.

Les seuls langages de programmation autorisés dans cette épreuve sont Python et SQL. Pour répondre à une question, il est possible de faire appel aux fonctions définies dans les questions précédentes. Dans tout le sujet on suppose que les bibliothèques `math`, `numpy` et `turtle` sont rendues accessibles grâce à l'instruction

```
import math, numpy as np, turtle
```

Si les candidats font appel à des fonctions d'autres bibliothèques, ils doivent préciser les instructions d'importation correspondantes.

Ce sujet utilise la syntaxe des annotations pour préciser le type des paramètres et du résultat des fonctions à écrire. Ainsi

```
def maFonction(n:int, X:[float], c:str, u) -> (int, np.ndarray):
```

signifie que la fonction `maFonction` prend quatre paramètres, le premier (`n`) est un entier, le deuxième (`X`) une liste de nombres à virgule flottante, le troisième (`c`) une chaîne de caractères et le type du dernier (`u`) n'est pas précisé. Cette fonction renvoie un couple dont le premier élément est un entier et le deuxième un tableau `numpy`. Il n'est pas demandé aux candidats de recopier les entêtes avec annotations telles qu'elles sont fournies dans ce sujet, ils peuvent utiliser des entêtes classiques. Ils veilleront cependant à décrire précisément le rôle des fonctions qu'ils définiraient eux-mêmes.

Les candidats peuvent à tout moment supposer qu'une fonction définie dans une question précédente est disponible, même s'ils n'ont pas traité la question correspondante. Pour les questions de programmation, cela revient à disposer d'une fonction respectant exactement la spécification de l'énoncé, sans propriété supplémentaire.

Dans ce sujet, le terme « liste » appliquée à un objet Python signifie qu'il s'agit d'une variable de type `list`. Les termes « vecteur » et « tableau » désignent des objets `numpy` de type `np.ndarray`, respectivement à une dimension ou de dimension quelconque. Enfin le terme « séquence » représente une suite itérable et indicable, indépendamment de son type Python, ainsi un tuple d'entiers, une liste d'entiers et un vecteur d'entiers sont tous trois des « séquences d'entiers ».

Une attention particulière sera portée à la lisibilité, la simplicité et l'efficacité du code proposé. En particulier, l'utilisation d'identifiants significatifs, l'emploi judicieux de commentaires et la description du principe de chaque programme seront appréciés.

Une liste de fonctions utiles est fournie à la fin du sujet.

I Modélisation sommaire d'un circuit

Dans un premier temps, nous considérons un circuit constitué uniquement de segments de droite et de virages à angle droit. Un tel circuit est représenté par une liste de chaînes de caractères dont les éléments sont "A", "G" et "D" où

- "A" représente une portion de ligne droite de longueur déterminée,
- "G" correspond à un virage à 90° à gauche et
- "D" à un virage à 90° à droite.

La figure 1 donne un exemple de circuit et de sa représentation.

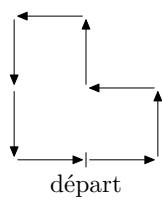


Figure 1 Circuit correspondant à la liste
["A", "G", "A", "G", "A", "D", "A", "G", "A", "G", "A", "G", "A"]

I.A – Validité de la représentation d'un circuit

Q 1. Écrire une fonction d'entête

```
def longueur1(c:[str], d:int) -> int:
```

qui prend en paramètres **c**, une liste représentant un circuit, et **d** la longueur, en mètres, des portions de ligne droite utilisées et renvoie la longueur totale du circuit en mètres.

Q 2. Donner la valeur de l'expression `représentation_minimale(["A", "A", "G", "D", "G", "G", "G", "A"])` où `représentation_minimale` est la fonction définie en figure 2. Que représente la variable **nbg** ?

```
1 def représentation_minimale(c:[str]) -> [str]:
2     virages = [[], ["G"], ["G", "G"], ["D"]]
3     nbg = 0
4     res = []
5     for e in c:
6         if e == "A":
7             res.extend(virages[nbg])
8             nbg = 0
9             res.append("A")
10        elif e == "G":
11            nbg = (nbg + 1) % 4
12        else:
13            nbg = (nbg - 1) % 4
14    res.extend(virages[nbg])
15    return res
```

Figure 2

Q 3. Expliquer en quelques lignes le but de la fonction `représentation_minimale`.

Q 4. Toutes les voitures sur un circuit automobile roulent dans le même sens. Ainsi, un demi-tour (deux virages à droite ou deux virages à gauche) n'est pas envisageable. Écrire une fonction d'entête

```
def contient_demi_tour1(c:[str]) -> bool:
```

qui prend en paramètre une représentation d'un circuit et renvoie `True` si le circuit **c** comporte un demi-tour et `False` s'il n'en contient pas.

Q 5. Écrire une fonction d'entête

```
def est_fermé1(c:[str]) -> bool:
```

qui détermine si le circuit **c** est fermé, autrement dit, si une voiture qui le parcourt revient à la fin du circuit à son point de départ dans la même orientation qu'au début.

Q 6. Il faut éviter qu'une partie de la trajectoire en croise ou se superpose à une autre, même en imaginant un tunnel ou un pont, cela rendrait la sécurité des pilotes très difficile à assurer. Écrire une fonction d'entête

```
def circuit_convenable1(c:[str]) -> bool:
```

qui détermine si le circuit **c**, fourni dans une représentation de longueur minimale, respecte les critères attendus : il est fermé et ne comporte pas de demi-tour, ni de sections qui se superposent ou se croisent. On pourra remarquer, en le justifiant, que, dans la modélisation utilisée, les croisements éventuels ont forcément lieu à une extrémité d'un élément de ligne droite.

I.B – Tracé d'un circuit

Q 7. En utilisant la bibliothèque `turtle`, dont une description figure en fin de sujet, écrire une fonction d'entête

```
def dessine_circuit1(c:[str], d:int) -> None:
```

qui prend en paramètre la représentation d'un circuit convenable et la longueur, en pixels, d'un segment de ligne droite et dessine le circuit correspondant à l'écran.

II Modélisation plus réaliste d'un circuit

Afin de représenter un circuit de manière plus réaliste, la modélisation précédente est enrichie en considérant que les virages sont des arcs de cercle. Un virage est désormais modélisé par un tuple de deux entiers (r, α) dans lequel le premier élément désigne le rayon du virage en mètres (entier strictement positif) et le second son angle en degrés (dans l'intervalle $]-360, 360[$). L'angle est orienté dans le sens trigonométrique, un angle positif désigne un virage à gauche et un angle négatif un virage à droite (figure 3). De plus, une ligne droite est désormais représentée par un entier correspondant à sa longueur en mètres (entier strictement positif). Un exemple de circuit de ce type est donné figure 4.

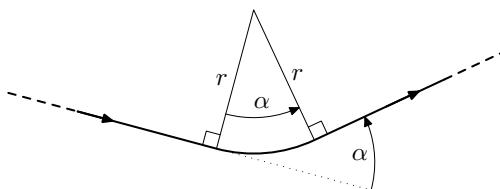


Figure 3 Représentation d'un virage

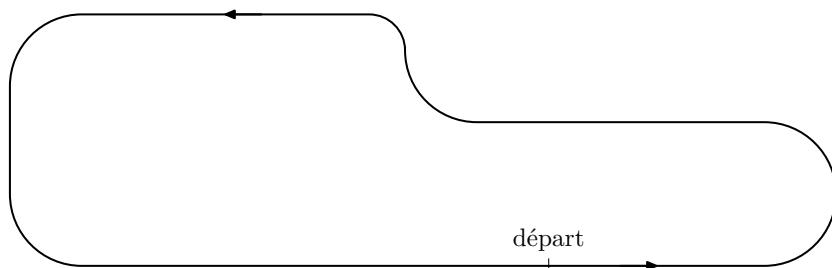


Figure 4 Circuit correspondant à la liste
[30, (10, 180), 40, (10, -90), (5, 90), 40, (10, 90), 15, (10, 90), 65]

Dans toute la suite du sujet, nous utiliserons cette nouvelle modélisation pour représenter un circuit. Cette modélisation reste toutefois imparfaite. Dans la réalité, les virages ne sont pas des arcs de cercle, leur rayon de courbure varie tout au long du virage. De plus un circuit n'est pas forcément plan et certains circuits proposent des virages relevés qui autorisent des vitesses plus élevées.

II.A – Validation

Q 8. Écrire une fonction d'entête

```
def élément_valide2(e) -> bool:
```

qui prend en paramètre un objet quelconque et détermine s'il peut figurer dans une liste représentant un circuit, autrement dit s'il s'agit d'un entier strictement positif ou d'un tuple de deux entiers de valeurs compatibles avec les spécifications de la modélisation.

Dans toute la suite du sujet, on considère que les représentations des circuits utilisées respectent la forme attendue.

II.B – Première méthode de tracé à l'écran

Q 9. En utilisant le module `turtle`, écrire une fonction d'entête

```
def dessine_circuit2(c:list, échelle:float) -> None:
```

qui trace à l'écran le circuit représenté par la liste `c` à l'échelle `échelle` exprimée en pixels par mètre.

II.C – Tracé pixel par pixel

Dans cette sous-partie, on réalise le tracé d'un circuit sous forme matricielle, pixel par pixel.

On représente l'écran par un tableau d'entiers à deux dimensions. Chaque élément du tableau représente un pixel, 0 désigne un pixel noir et 1 un pixel blanc. L'élément d'indice $(0, 0)$ correspond au pixel situé en bas à gauche. Le premier indice correspond à la colonne du pixel, le second à sa ligne.

On dispose des deux fonctions d'entête

```
def ligne(s:np.ndarray, début:np.ndarray, fin:np.ndarray) -> None:  
def arc(s:np.ndarray, début:np.ndarray, centre:np.ndarray, angle:int) -> None:
```

qui prennent en paramètre un tableau à deux dimensions `s` représentant l'écran et le modifient pour tracer, en noir, respectivement une ligne droite entre les pixels dont les coordonnées (colonne, ligne) sont données par les vecteurs `début` et `fin` et un arc de cercle d'angle `angle` degrés, commençant au pixel `début` et dont le centre est situé au point de coordonnées `centre` dans le système de coordonnées de l'écran d'une unité de pixel. Si l'angle est

positif, l'arc est tracé dans le sens trigonométrique, sinon, il est tracé dans le sens horaire. Si les tracés demandés sortent de l'écran, seules les parties visibles sont dessinées sans produire d'erreur.

Q 10. Écrire une fonction d'entête

```
def matrot(t:int) -> np.ndarray:
```

qui prend en paramètre un angle exprimé en degrés et renvoie un tableau numpy correspondant à la matrice de la rotation plane de centre O et d'angle t .

Q 11. La figure 5 présente un extrait de programme Python. En s'appuyant sur la figure 3, préciser la signification et la nature des paramètres de la fonction `cc` et de son résultat. Un schéma explicitant le principe de la fonction sera apprécié.

```
1 def signe(x):
2     if x > 0: return 1
3     elif x < 0: return -1
4     return 0
5
6 def cc(pos, dir, r, alpha):
7     return pos + matrot(dir) @ np.array([0., signe(alpha) * r])
```

Figure 5 Extrait de programme

Q 12. Écrire une fonction d'entête

```
def dessine_circuit3(s:np.ndarray, c:list, échelle:float) -> None:
```

qui prend en paramètres un tableau `s` représentant l'écran de l'ordinateur et le modifie pour tracer, en partant du centre de l'écran, le circuit représenté par la liste `c` à raison de `échelle` pixels par mètre.

III Le parcours d'une voiture

Cette partie s'intéresse au parcours par une voiture de course d'un circuit, tel qu'il a été modélisé dans la partie précédente. La voiture considérée est capable d'une accélération maximale notée a_{\max} , d'un freinage maximal (accélération négative) noté f_{\max} et d'une vitesse maximale v_{\max} . Nous supposons que la voiture peut produire son accélération maximale et son freinage maximal instantanément et les maintenir de manière continue tant que sa vitesse reste dans l'intervalle $[0, v_{\max}]$.

Dans le modèle utilisé où les virages sont des arcs de cercle, le rayon du virage détermine une vitesse recommandée pour prendre ce virage. Par ailleurs, indépendamment de la vitesse d'entrée dans un virage, on suppose que les virages sont toujours parcourus à vitesse constante.

Pour toutes les applications numériques, on prendra les valeurs $a_{\max} = 10 \text{ m}\cdot\text{s}^{-2}$, $f_{\max} = -20 \text{ m}\cdot\text{s}^{-2}$ et $v_{\max} = 100 \text{ m}\cdot\text{s}^{-1} = 360 \text{ km}\cdot\text{h}^{-1}$.

III.A – Formules de calcul des vitesses et temps de parcours

III.A.1)

Q 13. La voiture est à l'arrêt au début d'une ligne droite. Le pilote démarre et accélère au maximum, exprimer le temps nécessaire pour atteindre la vitesse maximale. Faire l'application numérique.

Q 14. Exprimer le temps minimal nécessaire à une voiture qui roule en ligne droite à la vitesse v_1 pour passer à la vitesse v_2 . Distinguer les cas $v_1 > v_2$ et $v_1 < v_2$. Faire l'application numérique pour $v_1 = 300 \text{ km}\cdot\text{h}^{-1}$ et $v_2 = 120 \text{ km}\cdot\text{h}^{-1}$.

Q 15. Montrer que la distance minimale nécessaire à une voiture qui roule en ligne droite à la vitesse v_1 pour passer à la vitesse $v_2 > v_1$ s'écrit

$$d = \frac{v_2^2 - v_1^2}{2 a_{\max}}$$

Q 16. Exprimer la distance minimale nécessaire à une voiture qui roule en ligne droite à la vitesse v_1 pour passer à la vitesse $v_2 < v_1$. Faire l'application numérique pour $v_1 = 300 \text{ km}\cdot\text{h}^{-1}$ et $v_2 = 120 \text{ km}\cdot\text{h}^{-1}$.

III.A.2)

On s'intéresse maintenant au temps de parcours d'une ligne droite de longueur d entre un virage d'où l'on sort à la vitesse v_1 et un virage qu'on doit prendre à la vitesse v_2 . Afin de parcourir la ligne droite le plus rapidement possible, le pilote accélère au maximum jusqu'au moment où il lui faut freiner pour arriver à l'entrée du virage à la vitesse v_2 .

Q 17. Exprimer d_{\min} , la longueur minimale de la ligne droite pour que la vitesse v_{\max} soit atteinte.

Q 18. Si la longueur de la ligne droite est supérieure au seuil calculé à la question précédente ($d \geq d_{\min}$), exprimer le temps minimal nécessaire pour parcourir la ligne droite.

Si $d < d_{\min}$, on note alors v_3 la vitesse maximale atteinte dans la ligne droite. On ne demande pas de calculer cette vitesse.

Q 19. Exprimer, en fonction de v_3 et des autres données du problème, le temps minimal nécessaire pour traverser une telle ligne droite.

III.B – *Implantation en Python*

Les instructions suivantes sont exécutées au début du programme Python, ainsi toutes les fonctions ont accès aux trois constantes **AMAX**, **FMAX** et **VMAX** correspondant respectivement aux valeurs maximales de l'accélération, du freinage et de la vitesse de la voiture considérée.

```
AMAX = 10 # accélération maximale en m/s2
FMAX = -20 # freinage maximal en m/s2
VMAX = 100 # vitesse maximale en m/s
```

On dispose par ailleurs des fonctions d'entête

```
def vr(r:int) -> float:
    def vmax_droite(d:int, v1:float, v2:float) -> float:
```

La fonction **vr** calcule la vitesse recommandée, en mètres par seconde, pour un virage de rayon **r** exprimé en mètres. La fonction **vmax_droite** détermine la vitesse maximale, en mètres par seconde, que l'on peut atteindre sur une ligne droite de longueur **d** (exprimée en mètres) dans laquelle on entre à la vitesse **v1** et dont on sort à la vitesse **v2** (exprimées en mètres par seconde) ; le résultat de cette fonction est toujours inférieur ou égal à la valeur de **VMAX**. Ces deux fonction s'exécutent en temps constant.

III.B.1) Temps pour un tour

Pour un circuit donné, on cherche d'abord à déterminer la vitesse maximale possible à l'entrée de chaque virage. Cette vitesse peut être inférieure à la vitesse recommandée pour un virage car il faut tenir compte d'un éventuel virage ultérieur plus serré, sans ligne droite de longueur suffisante pour freiner. Dans ce cas, il faut réduire la vitesse en amont afin de ne pas dépasser la vitesse recommandée dans le virage serré.

Q 20. Écrire une fonction d'entête

```
def vitesses_entree_max(c:list, vf:float) -> [float]:
```

qui prend en paramètre la représentation d'un circuit et la vitesse maximale (en mètres par seconde) à respecter à la fin du circuit et qui calcule, pour chacun de ses éléments (lignes droites et virages), la vitesse maximale à l'entrée de l'élément de façon à ne jamais dépasser la vitesse recommandée dans les virages qui suivent ni la vitesse **vf** en fin de circuit.

Q 21. Écrire une fonction d'entête

```
def temps_droite(d:int, v1:float, v2:float) -> (float, float):
```

qui prend en paramètre la longueur d'une ligne droite, la vitesse de la voiture à l'entrée de cette ligne droite et la vitesse maximale souhaitée à sa sortie et qui calcule le temps minimal nécessaire pour parcourir cette ligne droite et la vitesse effectivement atteinte à l'extrémité de la ligne droite. S'il n'est pas possible de sortir de la ligne droite avec une vitesse inférieure ou égale à **v2**, cette fonction lève l'exception **ValueError**.

Q 22. Écrire une fonction d'entête

```
def temps_tour(c:list, v0:float, vf:float) -> float:
```

qui prend en paramètre un circuit, la vitesse à l'entrée du circuit et la vitesse maximale autorisée à la fin du tour et qui calcule le temps minimal, en secondes, pour effectuer un tour de ce circuit, sans jamais dépasser la vitesse recommandée de chaque virage ni la vitesse finale maximale passée en paramètre.

III.B.2) Temps de course

Un grand prix de formule 1 est une course d'environ 300 km, ce qui représente entre 40 et 80 tours de circuit suivant la longueur de celui-ci. Au moment du départ les voitures sont à l'arrêt sur la grille de départ. Comme les tours de circuit s'enchainent, il convient d'anticiper à la fin d'un tour la vitesse pour débuter le tour suivant, afin d'être sûr de ne pas dépasser la vitesse d'entrée maximale du premier virage. Cependant, à la fin du dernier tour, les pilotes n'ont pas à négocier de prochain virage et disposent après la ligne d'arrivée d'un dégagement suffisant pour ralentir quelle que soit leur vitesse. La vitesse à la fin du dernier tour n'est donc pas limitée.

Q 23. Écrire une fonction d'entête

```
def temps_course(c:list, n:int) -> float:
```

qui calcule le temps minimal (en secondes) nécessaire pour effectuer une course de **n** tours du circuit **c** sans jamais dépasser la vitesse recommandée de chaque virage. Le départ est donné à l'arrêt, les lignes de départ et d'arrivée sont confondues et la vitesse sur la ligne d'arrivée à l'issue du dernier tour n'est pas limitée.

Q 24. Déterminer la complexité temporelle asymptotique dans le pire des cas de la fonction **temps_course** en fonction de **n** et du nombre de segments du circuit **c** (longueur de la liste).

IV Gestion des résultats

Depuis la création du championnat du monde de formule 1 en 1950, la fédération internationale de l'automobile (FIA) conserve l'ensemble des résultats des différents grands prix. Ces différentes informations sont stockées dans une base de données relationnelle dont un modèle physique simplifié est schématisé figure 6.

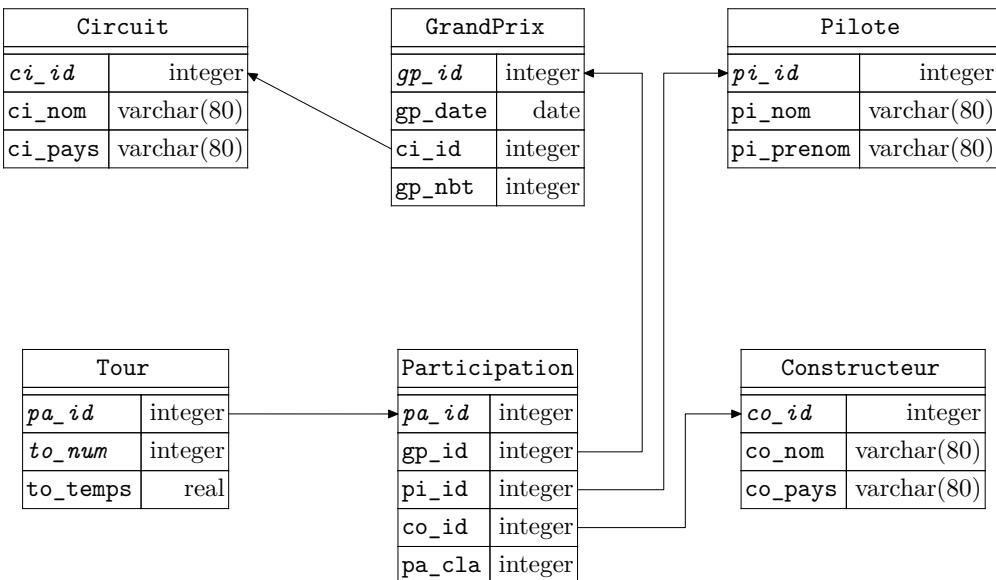


Figure 6 Structure physique simplifiée de la base de données du championnat du monde de formule 1

Cette base comporte les six tables listées ci-dessous avec la description de leurs colonnes :

- la table **Circuit** de circuits utilisés pour le championnat du monde de formule 1
 - **ci_id** identifiant (entier arbitraire) du circuit (clef primaire)
 - **ci_nom** nom du circuit (généralement le nom de la ville ou de la région dans laquelle il se trouve)
 - **ci_pays** pays dans lequel se trouve le circuit
- la table **GrandPrix** des courses comptant pour le championnat du monde
 - **gp_id** identifiant (entier arbitraire) du grand prix (clef primaire)
 - **gp_date** date de la course
 - **ci_id** circuit sur lequel s'est déroulé la course
 - **gp_nbt** nombre de tours de circuit
- la table **Pilote** des pilotes de formule 1
 - **pi_id** identifiant (entier arbitraire) du pilote (clef primaire)
 - **pi_nom** nom de famille du pilote
 - **pi_prenom** prénom du pilote
- la table **Constructeur** des écuries de formule 1
 - **co_id** identifiant (entier arbitraire) de l'écurie (clef primaire)
 - **co_nom** nom de l'écurie
 - **co_pays** pays d'origine de l'écurie
- la table **Participation** des participants (couple pilote, constructeur) pour une course donnée
 - **pa_id** identifiant (entier arbitraire) de la participation (clef primaire)
 - **gp_id** la course
 - **pi_id** le pilote
 - **co_id** le constructeur
 - **pa_cla** le classement à l'arrivée de ce participant ou NULL s'il n'a pas terminé la course ou a été disqualifié
- la table **Tour** des temps mis par un participant pour chaque tour de course, de clef primaire (**pa_id**, **to_num**)
 - **pa_id** identifiant (entier arbitraire) de la participation
 - **to_num** numéro du tour (de 1 à **gp_nbt**)
 - **to_temps** le temps (en secondes) pour le tour correspondant

Q 25. Pourquoi avoir utilisé la colonne **pi_id** comme clef primaire de la table pilote et pas la colonne **pi_nom** ?

Q 26. Estimer le nombre de lignes de la table **Tour**.

Q 27. Écrire une requête SQL qui liste, par ordre chronologique, la date et le nom du circuit de toutes les courses qui se sont déroulées en France (`ci_pays = 'France'`).

Q 28. Écrire une requête SQL qui liste, pour chaque course de l'année 2021, le nom du circuit, le nom du pilote gagnant et son temps de course.

Q 29. Expliquer le résultat de la requête suivante

```

1   SELECT ci_nom, pi_nom, gp_date, to_temps
2     FROM (SELECT ci_id, ci_nom, MIN(to_temps) AS mtt
3            FROM Circuit
4              JOIN GrandPrix ON GrandPrix.ci_id = Circuit.ci_id
5              JOIN Participation ON Participation_gp_id = GrandPrix_gp_id
6              JOIN Tour ON Tour_pa_id = Participation_pa_id
7            GROUP BY
8                  ci_id, ci_nom
9            ) AS rdc
10    JOIN GrandPrix ON GrandPrix.ci_id = rdc.ci_id
11    JOIN Participation ON Participation_gp_id = GrandPrix_gp_id
12    JOIN Pilote ON Pilote.pi_id = Participation_pi_id
13    JOIN Tour ON Tour_pa_id = Participation_pa_id AND to_tmmps = mtt
14 ORDER BY
15      ci_nom

```

Opérations et fonctions disponibles

Fonctions

- `a % b` calcule le reste de la division euclidienne de `a` par `b`, son signe est toujours celui de `b`
 $5 \% 3 \rightarrow 2, -5 \% 3 \rightarrow 1$.
- `isinstance(o, t)` teste si l'objet `o` est de type `t`
`isinstance(-13, int) \rightarrow True, isinstance((1, 2, 3), tuple) \rightarrow True.`
- `range(n:int)` renvoie la séquence des `n` premiers entiers ($0 \rightarrow n - 1$)
`list(range(5)) \rightarrow [0, 1, 2, 3, 4].`
- `range(d:int, f:int, s:int)` construit une séquence d'entiers espacés de `s` débutant à `d` et finissant avant `f`
`list(range(0, -10, -2)) \rightarrow [0, -2, -4, -6, -8].`
- `enumerate(s)` itère sur la séquence `s` en renvoyant, pour chaque élément de `s`, un couple formé de son indice et de l'élément considéré
`list(enumerate([3, 1, 4, 1, 5])) \rightarrow [(0, 3), (1, 1), (2, 4), (3, 1), (4, 5)].`
- `min(a, b, ...), max(a, b, ...)` renvoie son plus petit (respectivement plus grand) argument
`min(2, 4, 1) \rightarrow 1, max("Un", "Deux") \rightarrow "Un".`
- `math.sqrt(x)` calcule la racine carrée du nombre `x`.
- `round(n)` arrondit le nombre `n` à l'entier le plus proche.
- `math.pi` valeur approchée de la constante π .

Opérations sur les listes

- `len(u)` donne le nombre d'éléments de la liste `u`
`len([1, 2, 3]) \rightarrow 3, len([[1,2], [3,4]]) \rightarrow 2.`
- `u.count(e)` renvoie le nombre d'éléments de la liste `u` égaux à `e`
`[1, 2, 3, 1, 5].count(1) \rightarrow 2.`
- `u + v` construit une liste constituée de la concaténation des listes `u` et `v`
`[1, 2] + [3, 4, 5] \rightarrow [1, 2, 3, 4, 5].`
- `n * u` construit une liste constituée de la liste `u` concaténée `n` fois avec elle-même
`3 * [1, 2] \rightarrow [1, 2, 1, 2, 1, 2].`
- `u[i] = e` remplace l'élément d'indice `i` de la liste `u` par `e`.
- `u[i:j] = v` remplace les éléments de la liste `u` dont les indices sont compris dans l'intervalle $\llbracket i, j \rrbracket$ par ceux de la séquence `v` (peut modifier la longueur de la liste `u`).
- `u.append(e)` ajoute l'élément `e` à la fin de la liste `u` (identique à `u[len(u):] = [e]`). On suppose que cette opération a une complexité temporelle en $O(1)$.

- `u.extend(v)` ajoute les éléments de la séquence `v` à la fin de la liste `u` (identique à `u[len(u):] = v`). On suppose que cette opération a une complexité temporelle en $O(\text{len}(v))$.
- `u.insert(i, e)` insère l'élément `e` à la position d'indice `i` dans la liste `u` (en décalant les éléments suivants) ; si `i >= len(u)`, `e` est ajouté en fin de liste (identique à `u[i:i] = [e]`).
- `del u[i]` supprime de la liste `u` son élément d'indice `i` (identique à `u[i:i+1] = []`).
- `del u[i:j]` supprime de la liste `u` tous ses éléments dont les indices sont compris dans l'intervalle $\llbracket i, j \rrbracket$ (identique à `u[i:j] = []`).
- `u.reverse()` modifie la liste `u` en inversant l'ordre de ses éléments. On suppose que cette opération a une complexité temporelle en $O(\text{len}(u))$.
- `e in u` teste si l'élément `e` apparaît dans la liste `u`.

Opérations sur les tableaux

- `np.array(s)` crée un nouveau tableau contenant les éléments de la séquence `s`. La forme de ce tableau est déduite du contenu de `s`

$$\text{np.array}([[1, 2, 3], [4, 5, 6]]) \rightarrow \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$
- `a.ndim` nombre de dimensions du tableau `a`.
- `a.shape` tuple donnant la taille du tableau `a` pour chacune de ses dimensions.
- `a.size` nombre total d'éléments du tableau `a`.
- `np.around(a)` produit un tableau dont les éléments sont ceux du tableau `a` arrondis à l'entier le plus proche.
- `a @ b` calcule le produit matriciel des deux tableaux `a` et `b`. Les dimensions de `a` et `b` doivent être compatibles. On suppose que la multiplication d'une matrice $n \times n$ et d'un vecteur a une complexité temporelle en $O(n^2)$.

Module graphique turtle

Le module `turtle`, inspiré du langage Logo, simule une « tortue » robot qui se déplace sur l'écran. Cette tortue est munie d'un « crayon » qui, quand il est baissé, trace à l'écran la trajectoire de la tortue. Au début du programme, la tortue est située au centre de la fenêtre de visualisation, crayon baissé et orientée vers la droite de l'écran.

- `turtle.pendown()`, `turtle.penup()` baisse ou lève le crayon.
- `turtle.right(a)`, `turtle.left(a)` fait pivoter la tortue sur place à droite ou à gauche de `a` degrés.
- `turtle.forward(n)`, `turtle.back(n)` fait avancer ou reculer la tortue de `n` pixels (si `n` est négatif, le mouvement est inversé) ; `n` peut être un entier ou un nombre à virgule flottante.
- `turtle.circle(r, a)` fait parcourir par la tortue un arc de cercle de `a` degrés dont le centre est situé `r` pixels à gauche de la tortue. Si `r` est positif, le centre est effectivement à gauche de la tortue et l'arc de cercle est parcouru dans le sens trigonométrique ; si `r` est négatif, le centre est situé à droite de la tortue et l'arc de cercle est parcouru dans le sens horaire. Si `a` est positif, l'arc de cercle est parcouru en marche avant, la tortue tourne donc à gauche si `r` est positif et à droite si `r` est négatif. Si `a` est négatif, l'arc de cercle est parcouru en marche arrière, la tortue recule sur sa gauche si `r` est positif et sur sa droite si `r` est négatif.
- `turtle.home()` ramène la tortue au centre de la fenêtre, orientée vers la droite de l'écran.
- `turtle.reset()` efface la fenêtre et repositionne la tortue au centre, orientée vers la droite de l'écran.

Fonctions SQL

- `COUNT(*)` fonction agrégative qui retourne le nombre de lignes chaque groupe.
- `COUNT(e)` fonction agrégative qui calcule le nombre de valeurs non nulles chaque groupe de lignes.
- `MIN(e)`, `MAX(e)` fonctions agrégatives qui calculent respectivement le minimum et le maximum de `e` pour chaque groupe de lignes ; `e` est une expression de type numérique, chaîne de caractère ou date, heure.
- `SUM(e)` fonction agrégative qui somme `e` pour chaque groupe de lignes ; `e` est une expression de type numérique ou durée.
- `EXTRACT(c FROM d)` extrait la composante `c` de la date `d`, `d` peut être n'importe quelle expression de type `date`, `timestamp`, `time` ou `interval` et `c` peut prendre une des valeurs (liste non exhaustive) 'century', 'year', 'quarter' (trimestre, 1 à 4), 'month' (mois, 1 à 12), 'day' (jour du mois, 1 à 31), 'dow' (jour de la semaine, 0 à 6, 0 désignant dimanche), 'doy' (jour de l'année, 1 à 366).

• • • FIN • • •



ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH

ÉPREUVE SPÉCIFIQUE - FILIÈRE PSI

INFORMATIQUE

Durée : 3 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

Le sujet est composé de quatre parties.

L'épreuve est à traiter en langage **Python**. La syntaxe de Python est rappelée en Annexe en fin de sujet.

Les différents algorithmes doivent être rendus dans leur forme définitive sur le Document Réponse dans l'espace réservé à cet effet en respectant les éléments de syntaxe du langage (les brouillons ne sont pas acceptés).

La réponse ne doit pas se limiter à la rédaction de l'algorithme sans explication, les programmes doivent être expliqués et commentés.

Énoncé : 16 pages

Document Réponse et Annexe : 10 pages

Seul le Document Réponse doit être rendu dans son intégralité.

Assemblage et déformation de pièces automobiles

Introduction

L'objectif de cette étude est la modélisation du comportement géométrique de pièces déformables par la méthode des éléments finis. La **partie I** déterminera le posage des pièces et sera illustrée par un modèle 2 dimensions. La **partie II** permettra d'établir le maillage d'une surface plane pour pouvoir appliquer dans la **partie III** la méthode de résolution par éléments finis. La **partie IV** présentera la synthèse de l'étude et une application sur le cas d'une portière et de la caisse d'un véhicule.

Dans tout le sujet, il sera supposé que les modules python `numpy`, `matplotlib.pyplot` sont déjà importés dans le programme. Les fonctions et méthodes autorisées sont indiquées dans l'**annexe** à la fin du **Document Réponse**. De manière générale, l'utilisation des méthodes autres que les méthodes `append` et `remove` ne sera pas acceptée. L'utilisation des fonctions `min` ou `max` sera proscrite.

Partie I - Posage des pièces

I.1 - Présentation

Le posage consiste à mettre deux pièces en contact. Les pièces n'étant jamais parfaites, le nombre des points en contact n'est jamais infini comme cela pourrait être le cas lorsque l'on pose 2 surfaces parfaitement planes l'une sur l'autre. Ces points de contacts ponctuels doivent être identifiés car ils peuvent être à l'origine de variations géométriques et d'efforts particuliers.



Contact parfait



Contact réel

Figure 1 - Illustration du problème de posage sur des surfaces réelles

Pour illustrer la question du posage, on travaille en 2 dimensions et on se limite au posage d'une pièce sur une autre. Les géométries ont été discrétisées en un nombre fini de points. Un prétraitement, qui ne sera pas abordé ici, permet d'extraire les coordonnées des points concernés par le posage. En 2 dimensions, pour chaque point d'abscisse $x_k = k \Delta x$, est associée son altitude z_k où Δx correspond à une longueur élémentaire associée à la discréttisation spatiale. Ces données sont stockées dans une liste P de N nombres dont l'indice k des éléments permet de définir l'abscisse des points et les valeurs leur altitude. Ainsi, $x_k = k \Delta x$ et $z_k = P[k]$ où $0 \leq k < N$. La variable `Delta_x` sera associée à la longueur élémentaire Δx et sera supposée déjà définie. Il sera inutile de la passer en argument des différentes fonctions.

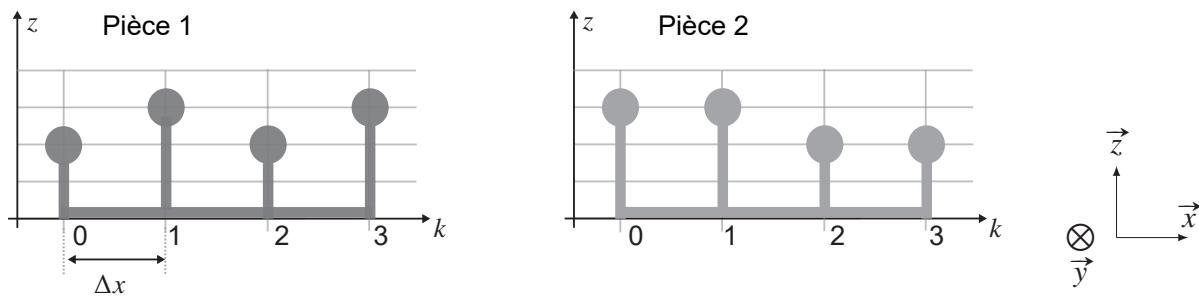


Figure 2 - Exemple de géométries discrètes associées à des pièces en 2D

Par exemple, sur la **figure 2**, on associe à la pièce de gauche la liste $P1 = [2., 3., 2., 3.]$.

Q1. Donner la liste $P2$ associée à la pièce 2 sur la **figure 2**.

Pour répondre au problème du posage, il faut commencer par retourner la pièce en effectuant une rotation autour de l'axe (O, \vec{y}) , pour que les points soient bien en vis-à-vis.

Q2. Écrire une fonction `retourne(Ls)` qui prend une liste Ls de nombres en argument et qui renvoie une nouvelle liste dont l'ordre des éléments est inversé et le signe de chaque élément lui aussi inversé. Par exemple, `retourne([1., 2., 3.])` renverra `[-3., -2., -1.]`.



Figure 3 - Exemple de posages entre 2 pièces. À gauche : posage aux points 1 et 3. À droite : posage aux points 1 et 2 avec collision en 3

On appellera posage optimal, le posage qui vérifie les deux contraintes suivantes :

- ◊ le minimum des distances algébriques entre les points en vis-à-vis doit être positif pour valider le fait qu'il n'y ait pas de collision,
- ◊ le maximum des distances entre les points en vis-à-vis doit être le plus petit possible.

Q3. À partir des deux pièces de la **figure 3**, représenter la pièce 2 associée au posage (contact) aux points 0 et 1, puis la pièce 2 associée au posage aux points 2 et 3. Indiquer dans chaque cas si le posage génère des collisions ou non.

Pour caractériser le posage, la méthode mise en œuvre consiste à repérer la position des points du maillage de la deuxième pièce par rapport à ceux de la première pièce. On se placera dans le cadre d'une pièce faiblement inclinée (sur la **figure 4**, l'inclinaison est amplifiée). Pour chacune des deux pièces, on associe une droite passant par les 2 points de contacts. Les deux droites ainsi définies seront coïncidentes lors du contact entre les deux pièces comme l'illustre la **figure 4**.

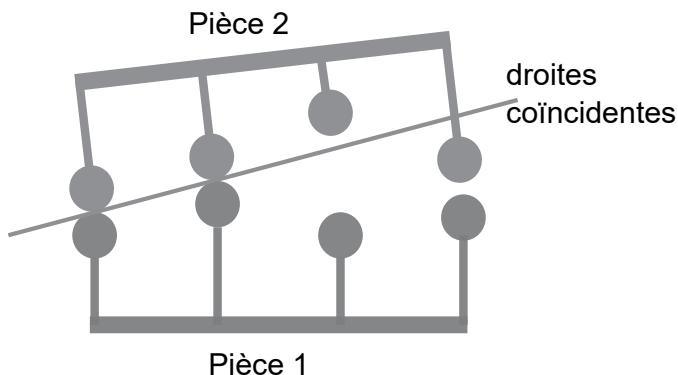


Figure 4 - Exemple de posage associé à deux autres pièces

On considère le posage sur les points n et p . On note z_{rn} et z_{rp} les altitudes des points de contact sur la pièce référence, et z_{pn} et z_{pp} les altitudes des points de contact sur la pièce à poser.

L'équation d'une droite qui passe par deux points de contact ayant pour coordonnées $(n\Delta x, z_n)$ et $(p\Delta x, z_p)$ est :

$$z = ax + b \quad (1)$$

avec pour coefficient directeur a et pour ordonnée à l'origine b tels que :

$$a = \frac{z_n - z_p}{(n - p)\Delta x} \quad \text{et} \quad b = z_n - \frac{n(z_n - z_p)}{n - p}.$$

- Q4.** Écrire la fonction `droite(Ls, n, p)` qui prend en arguments la liste des altitudes `Ls` et deux entiers `n` et `p` correspondant aux indices des deux points de contact et qui renvoie une liste contenant la pente a et l'ordonnée à l'origine b de la droite associée.

On applique cette fonction au bâti (pièce 1) et à la pièce 2 retournée. Pour un point donné d'une pièce donnée, on définit son altitude comme la distance algébrique entre ce point et le point sur la droite de même abscisse. Cette altitude sera comptée positivement si le point est au-dessus de la droite et négativement s'il est situé en-dessous. Pour une abscisse donnée, la distance algébrique entre deux points en vis-à-vis (chacun appartenant à une des deux pièces) est la différence entre leur altitude respective par rapport à la droite.

- Q5.** Écrire une fonction `posage(Ls1, Ls2, n, p)` qui prend en arguments 2 listes de points (`Ls1` et `Ls2`) et 2 entiers `n` et `p` qui correspondent aux indices des points de contact. La pièce associée à `Ls1` est située en-dessous de la pièce associée à `Ls2` (qui a été préalablement retournée). Cette fonction renvoie une liste de 2 éléments correspondant à la distance algébrique maximale et à la distance algébrique minimale entre les pièces discrétisées dans cette situation. Cette fonction devra faire appel à la fonction précédente `droite`.

- Q6.** Écrire une fonction `posage_opt(Ls1,Ls2)` qui prend en arguments 2 listes de même taille `Ls1` et `Ls2` (dans la position retournée) et qui renvoie une liste de 2 éléments correspondant aux abscisses des points associés au posage optimal. Cette fonction doit faire appel à la fonction définie en **Q5**. On ne traitera pas le cas où plusieurs couples de points peuvent conduire au posage optimal.
Indiquer en justifiant succinctement la complexité de cette fonction en terme de nombre de comparaisons.

Partie II - Maillage de la surface

On souhaite appliquer la méthode des éléments finis pour déterminer la déformation de la surface suite aux contraintes qui découlent du posage. Cette méthode nécessite, au préalable, un découpage de la surface en surfaces élémentaires. Cette étape s'appelle le maillage.

Le maillage proposé lors de cette étude est un maillage triangulaire. Chaque surface élémentaire est alors un triangle dont les sommets sont des nœuds de la surface à étudier. Le maillage est une étape déterminante permettant d'obtenir des résultats cohérents ou non par la méthode des éléments finis. En effet, un mauvais maillage comme illustré à gauche **figure 5** peut être à l'origine d'instabilités numériques ou encore d'erreurs d'arrondis. Au contraire, une triangulation de Delaunay illustrée à droite **figure 5** présente de bons résultats lors d'un traitement numérique. Dans ce maillage, les triangles ont des arêtes courtes, des angles ni trop petits, ni trop grands.

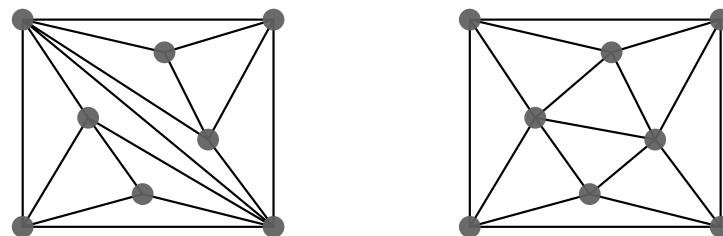
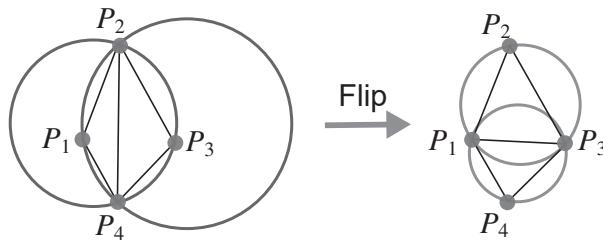


Figure 5 - Exemples de 2 triangulations obtenues à partir de nœuds identiques

Une triangulation est dite de **Delaunay** lorsque le cercle circonscrit à chaque triangle ne contient aucun autre point du maillage.

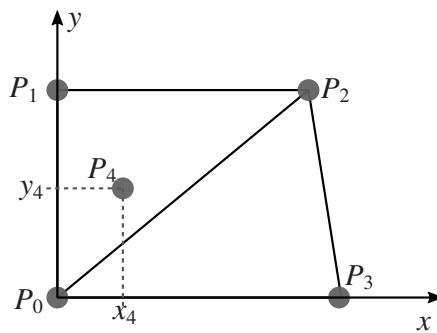
Une triangulation peut devenir une triangulation de Delaunay par une suite de basculements (flips) successifs : quand 2 triangles ne respectent pas la condition de Delaunay, il faut remplacer l'arête commune P_2P_4 par l'arête P_1P_3 comme l'illustre la **figure 6**.

L'algorithme mis en œuvre pour réaliser une triangulation de Delaunay est un algorithme itératif. La construction s'appuie sur un maillage initial de Delaunay. Ce maillage constitué d'un faible nombre de nœuds est construit de telle sorte que la surface ainsi délimitée contienne l'ensemble des nœuds qui devront être ajoutés à chaque itération. Ce maillage initial peut être composé d'un ou de plusieurs triangles vérifiant le critère de Delaunay.

**Figure 6** - Principe du basculement

Chaque nœud P_k restant du maillage est alors ajouté en suivant l'algorithme :

- ◊ repérer à quel triangle T_{in} appartient le point P_k (par exemple sur la **figure 7**, P_4 appartient au triangle $P_0P_1P_2$),
 - ◊ construire 3 triangles à partir du point P_k et des sommets du triangle T_{in} . Ajouter ces triangles à la liste des triangles \mathbf{T} . Supprimer le triangle T_{in} de la liste \mathbf{T} .
- La triangulation ainsi obtenue n'est pas nécessairement de Delaunay. Pour chaque nouveau triangle :
- vérifier que les triangles voisins respectent la triangulation de Delaunay. Si ce n'est pas le cas, basculer l'arête commune (**figure 6**).
 - répéter cette opération pour les triangles modifiés après le basculement.

**Figure 7** - Ajout d'un point à un maillage de Delaunay

Structure des données

On dispose de N nœuds P_k d'une surface plane repérés par leurs coordonnées cartésiennes (x_k, y_k) . Ces nœuds sont stockés dans le tableau `noeud` qui est un tableau Numpy de flottants de taille $N \times 2$. Chaque nœud est repéré par son indice dans le tableau `noeud`. L'élément `k` du tableau `noeud` contient les coordonnées cartésiennes du nœud d'indice `k`.

Ainsi `noeud[k] = array([xk, yk])`.

On souhaite construire un tableau `T` qui contient des listes à 3 éléments. Ces listes représentent l'indice des nœuds d'un même triangle. Dans le cas de la **figure 7**, la liste `T` s'écrit : `T = [[0,1,2], [2,3,0]]`. L'ordre des indices des nœuds est sans importance.

- Q7.** On considère le maillage de Delaunay de la **figure 7**. On souhaite ajouter le nœud P_4 à ce maillage et faire les transformations nécessaires pour obtenir une nouvelle triangulation de Delaunay. Sur les figures du DR, représenter les différentes transformations qui permettent d'aboutir à une triangulation de Delaunay. Indiquer la liste `T` qui résulte de cette triangulation.

L'algorithme de Delaunay doit permettre de construire la liste T des triangles du maillage. Cette liste est initialisée par 2 triangles contenant tous les nœuds du maillage. Ensuite, à chaque nouveau nœud ajouté, on applique l'algorithme de triangulation, pour mettre à jour la liste T .

L'implémentation de cet algorithme nécessite l'utilisation de plusieurs fonctions :

- ★ `addT(indN, indT)` : fonction qui prend en arguments l'indice d'un nœud `indN` à l'intérieur du triangle d'indice `indT`. Cette fonction modifie la liste T en insérant les nouveaux triangles et en supprimant le triangle d'indice `indT`;
- ★ `circle(T1)` : fonction qui prend en argument une liste à 3 éléments (les indices des sommets d'un triangle) et qui renvoie les coordonnées du centre du cercle circonscrit ainsi que le rayon $[[xc, yc], R]$;
- ★ `insideT(indN, T1)` : fonction qui prend en arguments l'indice d'un nœud `indN` et une liste correspondant aux indices des sommets d'un triangle `T1`. Cette fonction renvoie un booléen suivant si le nœud appartient ou non au triangle considéré ;
- ★ `insideC([xc, yc], R, [x, y])` : fonction qui prend en argument les coordonnées du centre, le rayon et les coordonnées d'un point. Cette fonction renvoie un booléen suivant si le point considéré appartient ou non au cercle de centre (xc, yc) de rayon R ;
- ★ `edge(indN, indT)` : fonction qui prend en arguments l'indice d'un nœud `indN` et l'indice d'un triangle `indT`. Cette fonction renvoie l'indice du triangle ayant pour arête commune l'arête à l'opposé du nœud considéré ;
- ★ `flip(indT1, indT2)` : fonction qui prend en arguments 2 indices (associés à des triangles de la liste T). Cette fonction permet de basculer l'arête commune en modifiant en conséquence la liste T .

Les tableaux `noeud` et `T` sont des variables globales qu'il sera inutile de passer en arguments des fonctions et qui pourront être utilisées et modifiées à l'intérieur de celles-ci. Dans la suite du sujet, on considérera que les différents nœuds sont toujours strictement inclus dans un triangle de la liste T et qu'ils ne sont jamais situés sur l'une des arêtes.

Q8. Écrire la fonction `addT(indN, indT)` décrite ci-dessus.

Q9. La fonction `flip(indT1, indT2)` qui permet de basculer l'arête commune à 2 triangles en modifiant les sommets de 2 triangles de la liste T est définie ci-dessous. Choisir l'une des 3 propositions données pour compléter les instructions manquantes (indiquées par **instructions à compléter**).

```

1 def flip(indT1, indT2):
2     T1 = T[indT1]
3     T2 = T[indT2]
4
5     Ledge, Lflip = [], []
6
7     #instructions à compléter
8
9     T[indT1] = Lflip + [Ledge[0]]
10    T[indT2] = Lflip + [Ledge[1]]
```

Proposition 1

```

Ls = T1+T2
for k in range(len(Ls)):
    for j in range(k+1,len(Ls)):
        if Ls[k] == Ls[j]:
            Ledge+=[Ls[k]]
        else:
            Lflip+=[Ls[k]]

```

Proposition 2

```

for k in range(3):
    if T2[k] in T1:
        Ledge+=[T2[k]]
    else:
        Lflip+=[T2[k]]
for noeud in T1:
    if noeud not in Ledge :
        Lflip+=[noeud]

```

Proposition 3

```

for k in range(3):
    if T2[k] == T1[k]:
        Ledge+=[T2[k]]
    else:
        Lflip+=[T2[k]]
        Lflip+=[T1[k]]

```

Pour vérifier si un point M du plan cartésien appartient à un triangle ABC, on propose de faire le produit vectoriel $\vec{AB} \wedge \vec{AM}$ et de vérifier que le résultat est du même signe que le produit vectoriel $\vec{AB} \wedge \vec{AC}$. On procédera de la même manière pour chaque sommet en comparant les signes de $\vec{BC} \wedge \vec{BM}$ à $\vec{BC} \wedge \vec{BA}$ et enfin de $\vec{CA} \wedge \vec{CM}$ à $\vec{CA} \wedge \vec{CB}$. L'annexe du sujet rappelle la syntaxe et la définition du produit vectoriel sous numpy.

Q10. Écrire la fonction `insideT(indN, T1)` qui prend en argument l'indice d'un nœud `indN` et une liste `T1`, associée à un triangle. Cette fonction renvoie un booléen suivant si le nœud appartient au triangle ou non (True si le noeud appartient au triangle, False sinon).

Une fois le nœud d'indice `indN` inséré, pour se ramener à une configuration de Delaunay, on applique la fonction récursive `checkedge(indN, indT)` où `indN` est l'indice du nœud inséré, et `indT` un triangle qui a pour sommet le nouveau nœud. Cette fonction doit d'abord vérifier si le nœud inséré est inclus dans les cercles circonscrits des triangles voisins, c'est-à-dire, les triangles qui possèdent une arête commune aux nouveaux triangles. Par exemple, sur la **figure 7**, P_0P_2 est une arête commune à un nouveau triangle formé par l'insertion du nœud P_4 et au triangle défini par les nœuds P_0 , P_2 et P_3 . Si le nœud inséré appartient effectivement à l'un des cercles circonscrits, il faut alors procéder à un basculement grâce à la fonction `flip`. Une fois ce basculement réalisé, il faut à nouveau s'assurer que le nœud d'indice `indN` et les triangles ainsi modifiés vérifient le critère de Delaunay en appliquant la fonction `checkedge`.

```

1 def checkedge(indN , indT) :
2     indA = edge(indN , indT)
3     if indA != None:
4         C,R = cercle(T[indA])
5         if insideC(C,R,noeud[indN]) :
6             instruction à compléter
7             instruction à compléter
8             instruction à compléter

```

Q11. Indiquer l'intérêt de la condition ligne 3 de la fonction `checkedge`. Compléter les lignes 6, 7 et 8 de cette fonction.

Q12. Écrire la fonction `delaunay(noeud)` qui prend en arguments un tableau de nœuds (de dimension $N \times 2$) et qui renvoie la liste des triangles `T` correspondant à une triangulation de Delaunay. Cette fonction pourra utiliser les fonctions `insideT`, `addT` et `checkedge` déjà définies. On considère toujours que chaque nœud inséré est strictement inclus dans un triangle de la distribution. De plus, les 4 premiers éléments du tableau `noeud` permettent d'initialiser le tableau des triangles : `T = [[0,1,2], [2,3,0]]`.

Partie III - Détermination de la matrice de rigidité d'une pièce

À partir de ce maillage, l'objectif est maintenant de calculer numériquement la déformation de la pièce étudiée.

Dans la suite du sujet, on considérera que toutes les constantes `N`, `E`, `S`, `L`, `Fx`, `p0` et `n` ont été préalablement définies de manière globale.

Pour illustrer cette partie, nous travaillons sur un modèle 1D où 2 nœuds du maillage sont reliés par une barre.

III.1 - Modélisation mécanique

Chaque barre se déforme proportionnellement à l'effort subit, comme un ressort. La raideur dépend de la section de la barre S de sa longueur L et de l'élasticité du matériau E (module d'Young) :

$$\vec{F} = k \Delta L \vec{v} = \frac{ES}{L} \Delta L \vec{v} \quad (2)$$

où \vec{v} est un vecteur unitaire de même direction que la barre (**figure 8**).

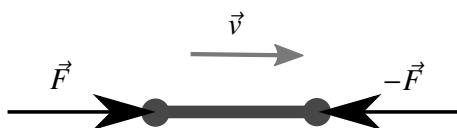


Figure 8 - Efforts exercés au niveau des barres

On considère que les efforts extérieurs ne s'exercent que sur les nœuds. Le problème consiste à déterminer les déplacements des nœuds.

Sur des structures complexes, le système d'équations peut atteindre plusieurs centaines de milliers voire des millions d'inconnues. Une mise en œuvre numérique est alors indispensable.

III.2 - Calcul de la déformation d'une poutre en traction

Pour introduire la méthode de résolution par éléments finis, nous allons travailler sur un problème simple à une dimension : la poutre en traction.

La poutre représentée **figure 9** est encastrée en O . Cette poutre de longueur L de section S est soumise à un effort $F_x = 100 \text{ N}$ en A et à un effort réparti, également suivant x , représenté par une densité linéique de force $p(x) = p_0 = 500 \text{ N/m}$ (non représenté).

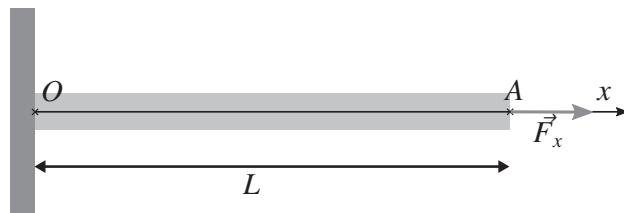


Figure 9 - Paramétrage du problème de la poutre

La théorie de la résistance des matériaux nous fournit les équations suivantes :

$$\begin{cases} \frac{dN}{dx} + p(x) = 0 \\ N = ES \frac{du}{dx} \end{cases} \quad \text{avec comme conditions aux limites} \quad \begin{cases} u(0) = 0 \\ N(L) = F_x \end{cases}$$

où N correspond à l'effort normal dans la poutre et u le déplacement longitudinal dû à la traction. En combinant ces deux premières équations, on obtient :

$$ES \frac{d^2u}{dx^2} = -p(x). \quad (3)$$

Après intégration, on obtient simplement l'expression analytique suivante :

$$u_{th}(x) = -\frac{p_0}{2ES}x^2 + \frac{(F_x + Lp_0)}{ES}x.$$

III.3 - Tracé de la solution analytique

Q13. Écrire les instructions permettant de créer un vecteur x de n points régulièrement répartis entre 0 et L inclus, puis permettant de créer le vecteur u_{th} représentant le déplacement dans la poutre des points d'abscisse x . Enfin, écrire les instructions permettant de tracer la solution analytique $u_{th}(x)$.

L'équation différentielle peut être résolue analytiquement dans ce cas simple, mais dans le cadre général des problèmes d'élasticité 2D ou 3D, ce n'est pas possible. On envisage alors une résolution numérique en appliquant la méthode d'approximation que sont les éléments finis.

III.4 - Recherche de la solution par éléments finis

La méthode de résolution par éléments finis consiste à transformer l'équation différentielle en un système linéaire d'équations.

En repartant de l'équation (3), la multiplication de l'équation différentielle par une fonction test Φ , puis son intégration permettent d'aboutir à la relation suivante :

$$\int_0^L ES \frac{d^2u}{dx^2} \Phi(x) dx = \int_0^L -p(x) \Phi(x) dx$$

où la fonction test Φ (dont un exemple est donné **figure 10**) doit vérifier les mêmes conditions aux limites que la fonction u , soit $\Phi(0) = 0$.

Une intégration par parties nous amène au résultat suivant :

$$\int_0^L ES \frac{du}{dx} \frac{d\Phi}{dx} dx = \int_0^L p(x)\Phi(x) dx + N(L)\Phi(L). \quad (4)$$

Nous allons alors chercher une solution linéaire par morceaux à l'équation (4) ce qui va nous amener à la discrétisation du problème et à son écriture matricielle.

Nous cherchons alors $u(x)$ sous la forme :

$$u(x) = \sum_{k=0}^n u_k \Phi_k(x) \quad (5)$$

où les fonctions Φ_k sont appelées fonctions de forme illustrées sur la **figure 10**. Ces fonctions ont les propriétés suivantes :

- ◊ $x_k = k \frac{L}{n}$ et $0 \leq k \leq n$;
- ◊ Φ_k est continue ;
- ◊ Φ_k est affine sur $[x_{k-1}, x_k]$ et sur $[x_k, x_{k+1}]$;
- ◊ $\Phi_k(x_k) = 1$;
- ◊ $\forall x \in [0, x_{k-1}] \cup [x_{k+1}, L], \Phi_k(x) = 0$;
- ◊ même si les fonctions Φ_k ne sont pas dérivables en tout point de $[0, L]$, on considérera les dérivées $\frac{d\Phi_k}{dx}$ comme des fonctions continues par morceaux, en les prolongeant de manière quelconque aux points de "cassure". Le choix du prolongement n'aura de toute façon aucun impact sur les calculs d'intégrales demandés.

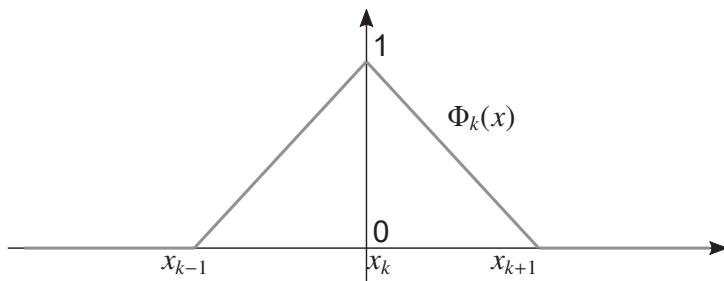


Figure 10 - Fonction de forme $\Phi_k(x)$

Les scalaires u_k constituent les inconnues du problème et représentent le déplacement associé à chaque noeud k . Une fois les déplacements u_k déterminés, il est possible de calculer la fonction déplacement u en n'importe quel point x de la poutre grâce à l'équation (5).

Q14. Écrire une fonction `phi(k, x)` qui prend en argument un entier k et un flottant $x \in [0, L]$ et qui renvoie la valeur $\Phi_k(x)$.

Nous pouvons alors réécrire le problème sous forme matricielle. En notant U et Φ les vecteurs respectivement associés aux déplacements u_k et aux fonctions $\phi_k(x)$, on peut écrire la fonction u de la manière suivante :

$$U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \end{bmatrix} \quad \Phi(x) = \begin{bmatrix} \Phi_0(x) \\ \Phi_1(x) \\ \vdots \\ \Phi_n(x) \end{bmatrix} \quad u(x) = {}^t U \Phi(x) \quad (6)$$

où ${}^t U$ correspond à la transposée du vecteur U . La dérivée de u par rapport à x s'écrit alors simplement :

$$\frac{du}{dx} = {}^t U \frac{d\Phi}{dx}. \quad (7)$$

Pour résoudre l'équation (4) il faut alors déterminer la matrice $M = \int_0^L \frac{d\Phi}{dx} {}^t d\Phi dx$.

Q15. Exprimer le produit des dérivées $\frac{d\Phi_i}{dx} \frac{d\Phi_j}{dx}$ pour $x \in [0, L]$ en fonction de n et de L lorsque $i = j$, $i = j = 0$, $i = j = n$, $|i - j| = 1$ et dans les autres cas. En déduire l'intégrale $\int_0^L \frac{d\Phi_i}{dx} \frac{d\Phi_j}{dx} dx$ à nouveau en fonction de n et L lorsque $i = j$, $i = j = 0$, $i = j = n$, $|i - j| = 1$ et dans les autres cas.

Les fonctions de forme ainsi choisies permettent d'aboutir à une matrice M tridiagonale :

$$M = C \begin{bmatrix} \alpha/2 & \beta & & & \\ \beta & \alpha & \beta & & \\ & \ddots & \ddots & \ddots & \\ & & \beta & \alpha & \beta \\ & & & \beta & \alpha/2 \end{bmatrix}.$$

Q16. À partir des résultats de la question précédente et en posant $\beta = -1$, exprimer les coefficients C et α .

Le problème peut alors se réécrire, à partir de l'équation (4) de la manière suivante :

$$ES MU = P + F_x \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad (8)$$

où P est le vecteur associé à l'effort appliqué aux nœuds de la poutre. Par la suite, on considère que le vecteur P et la matrice M ont été définis comme des tableaux numpy.

Cette équation peut se mettre sous la forme suivante et correspond à un système linéaire de n équations à n inconnues :

$$AU = B \quad (9)$$

Q17. Exprimer la matrice A et le vecteur B en fonction de M , P et des différentes constantes.
Écrire les instructions permettant de créer A et B .

La résolution d'un tel système linéaire peut se faire à l'aide de l'algorithme du pivot partiel de Gauss.

La fonction `resolution` prend en arguments un tableau R de dimensions $(n \times n)$ de type flottant et un vecteur à n éléments de type flottant. Cette fonction renvoie un vecteur à n éléments solution de l'équation $RX = Y$.

Les fonctions `combinaison` et `echanger_ligne` permettent respectivement de faire une combinaison linéaire entre deux lignes du système et d'échanger deux lignes du système.

```

1 | def resolution(R,Y) :
2 |     Syst = []
3 |     nb = len(M)
4 |     for k in range(nb):
5 |         ligne = []
6 |         for j in range(nb):
7 |             ligne.append(R[k,j])
8 |             ligne.append(Y[k])
9 |         Syst.append(ligne)
10 |    #Mise sous forme triangulaire
11 |    for i in range(nb):
12 |        j=pivot(Syst,i)
13 |        echanger_lignes(Syst,i,j)
14 |
15 |        for k in range(i+1,nb):
16 |            #instruction à compléter
17 |            combinaison(Syst,k,i,mu)
18 |    #Remontée
19 |    X=[0]*nb
20 |
21 |    #instructions à compléter
22 |
23 |    return X

```

Q18. Indiquer l'intérêt des lignes 2 à 9 de la fonction `resolution`.

Q19. À la ligne 12, la fonction `resolution` fait appel à la fonction `pivot` qui prend en arguments le tableau `Syst` et un indice `k` et qui renvoie l'indice du pivot le plus grand en valeur absolue parmi les pivots encore disponibles dans la colonne d'indice `k`. Indiquer la structure du tableau `Syst`. Expliquer en quoi il est pertinent d'appliquer cet algorithme avec le pivot le plus grand en valeur absolue. Écrire la fonction `pivot` répondant au problème.

Q20. La fonction `combinaison` prend en arguments un tableau `Syst`, 2 entiers `k` et `i` et un flottant `x`. Cette fonction modifie la ligne `k` du tableau `Syst` en réalisant la transvection :

$$\text{Ligne } k = \text{Ligne } k + mu \times \text{Ligne } i.$$

Compléter la ligne 16 du code proposé en définissant la variable `mu`. Écrire une fonction `combinaison` qui réalise l'opération souhaitée.

Une fois le problème mis sous forme triangulaire, il reste à créer le vecteur des solutions `X`. C'est la phase de remontée.

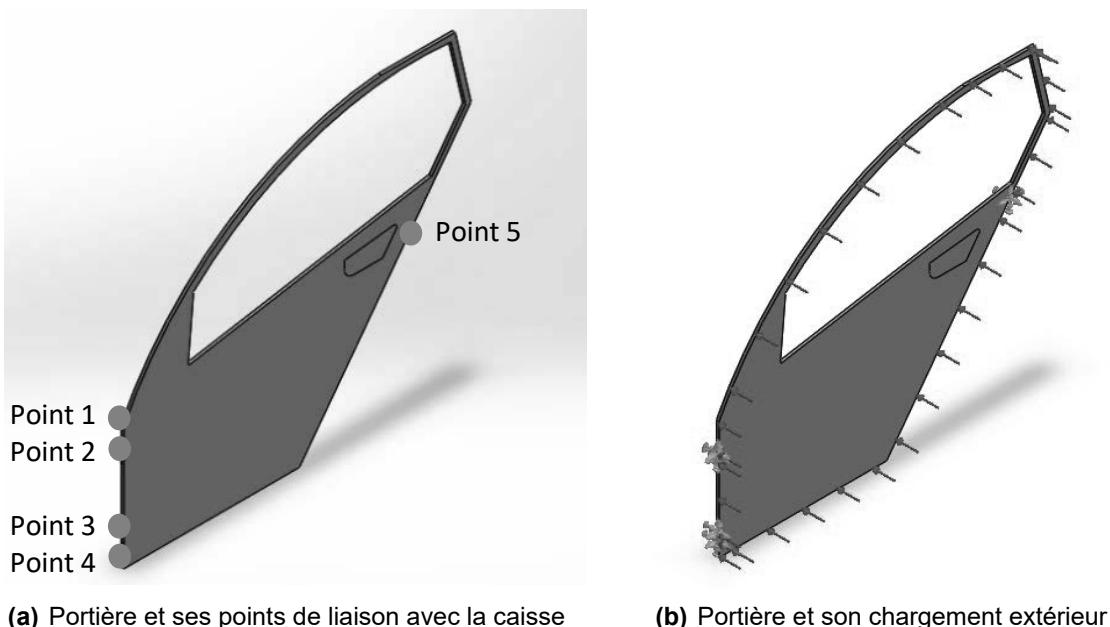
Q21. Proposer les instructions permettant de construire le vecteur `X` à partir du système triangulaire.

Q22. En utilisant la matrice `A` et le vecteur `B` définis en **Q17**, écrire l'instruction permettant de résoudre l'équation (8) et d'affecter le résultat à un vecteur `U`. À partir des fonctions précédemment définies, définir la fonction `uEF` qui construit $u(x)$ le déplacement u défini à l'équation (5) et solution du problème à partir du vecteur `U` et de la fonction `phi(k, x)`.

Partie IV - Analyse et comparaison des résultats

Pour illustrer cette étude sur un exemple 3D, nous proposons un modèle discréteisé de portière de véhicule. Cette portière est posée sur la caisse du véhicule (**figure 11(a)**) :

- au niveau des points 1 et 2 pour la charnière haute,
- au niveau des points 3 et 4 pour la charnière basse,
- au niveau du point 5 pour la gâche.



(a) Portière et ses points de liaison avec la caisse (b) Portière et son chargement extérieur

Figure 11 - Modélisation d'une portière

IV.1 - Illustration de la question du posage

On modélise des défauts de position des points de la caisse afin de visualiser leur impact sur la position de la portière. Ainsi, on peut déterminer pour chaque type de défaut dans quel ordre il est préférable d'assembler la portière sur la caisse. On va proposer deux scénarios d'assemblage :

- scénario 1 : commencer par poser la caisse au niveau des points 1, 2 et 5 (**figure 12 en haut**) avant de la déformer pour assembler la charnière basse ;
- scénario 2 : commencer par poser la caisse au niveau des points 3, 4 et 5 (**figure 12 en bas**) avant de la déformer pour assembler la charnière haute.

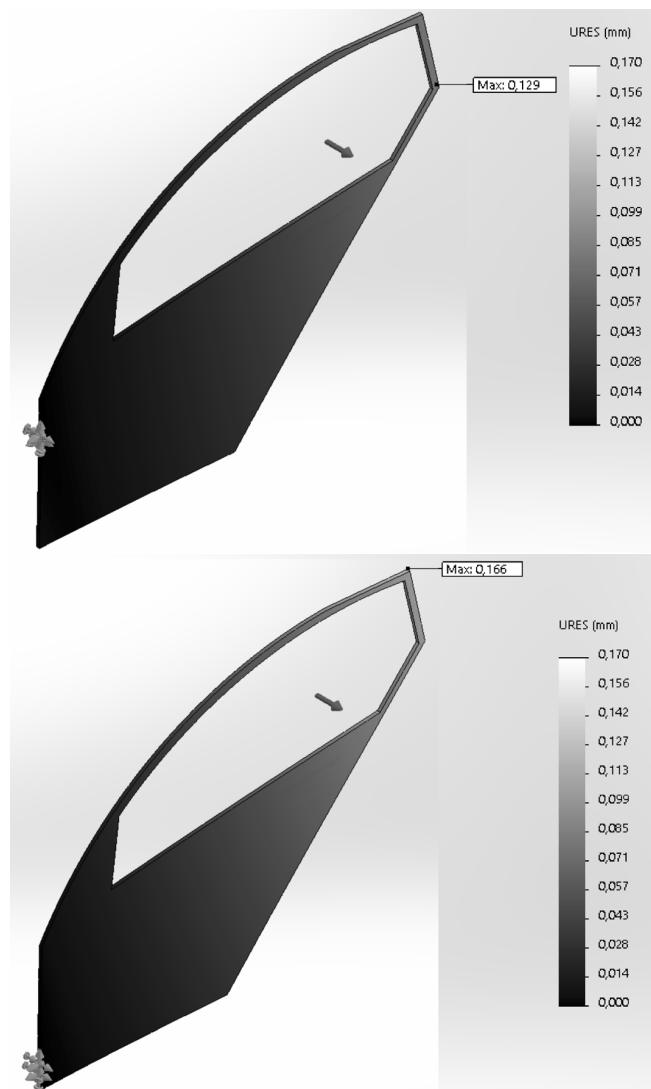


Figure 12 - Illustration du posage. En haut : portière posée au niveau des points 1, 2 et 5. En bas : portière posée au niveau des points 3, 4, 5.

Q23. Indiquer en justifiant succinctement quel posage vous semble respecter les règles associées au posage optimal définies en partie I de ce sujet.

IV.2 - Illustration de l'influence de la finesse du maillage

Pour visualiser l'impact de la taille des triangles du maillage, on réalise plusieurs simulations numériques. Un déplacement est imposé aux 5 points de liaison avec la caisse et un chargement est imposé sur tous les nœuds du pourtour de la portière comme l'illustre la **figure 11(b)**.

	Maillage fin	Maillage intermédiaire	Maillage grossier
Nombre de nœuds	83 336	16 163	4 957
Nombre de triangles	46 925	7 763	2 266
Déplacement maximal (mm)	12,154	12,664	13,120

Tableau 1 - Tableau de comparaison des différentes simulations

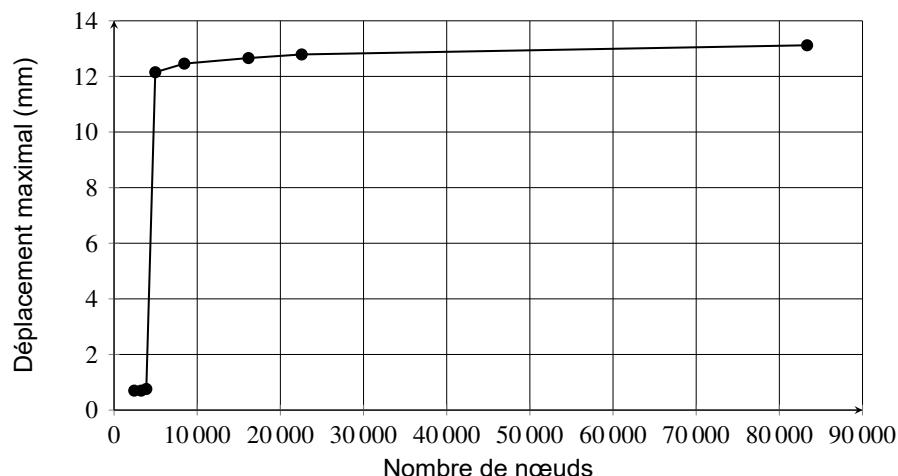


Figure 13 - Valeur maximale du déplacement (en mm) en fonction du nombre de nœuds.

Q24. En exploitant les données du **tableau 1** et la **figure 13**, déterminer l'ordre de grandeur du nombre minimal d'éléments du maillage pour que le résultat du calcul soit satisfaisant (précision des résultats inférieure à 1 mm).

Q25. En exploitant la forme spécifique de la géométrie de la portière, justifier le seuil observé pour 5 000 éléments environ dans le **tableau 1** et **figure 13**.

FIN

 CINP CONCOURS COMMUN INP	Numéro d'inscription <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/>	Nom : _____
	Numéro de table <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 50px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/>	Prénom : _____
	Né(e) le <input style="width: 25px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 25px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 25px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/> <input style="width: 25px; height: 30px; border: 1px solid black; margin-right: 10px;" type="text"/>	
Emplacement QR Code	Filière : PSI	Session : 2022
Épreuve de : INFORMATIQUE		
Consignes	<ul style="list-style-type: none"> • Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer • Rédiger avec un stylo non effaçable bleu ou noir • Ne rien écrire dans les marges (gauche et droite) • Numérotter chaque page (cadre en bas à droite) • Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre 	

PSI5IN

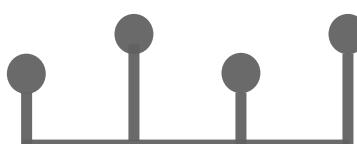
DOCUMENT RÉPONSE

Ce Document Réponse doit être rendu dans son intégralité.

Q1 - Liste P2

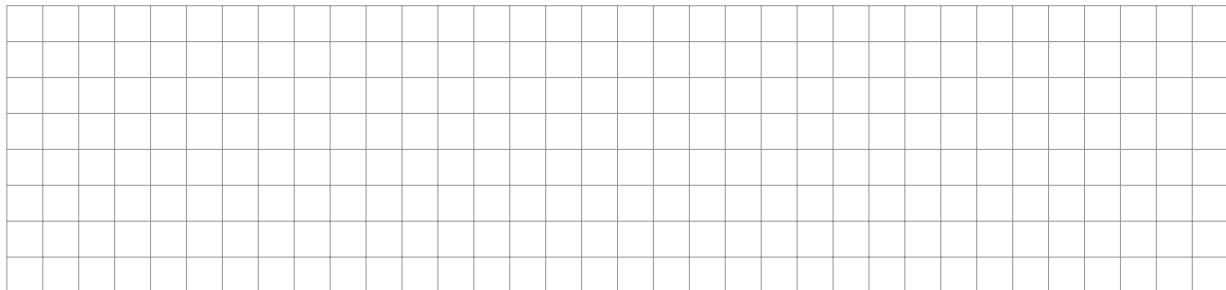
Q2 - Fonction retourne(Ls)

Q3 - Représentation du posage aux points 0 et 1, puis aux points 2 et 3



NE RIEN ÉCRIRE DANS CE CADRE

Q4 - Fonction droite(Ls,n,p)



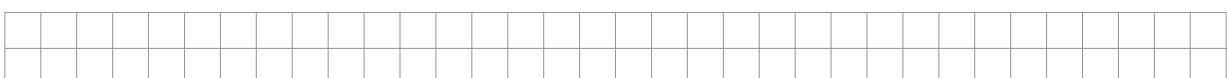
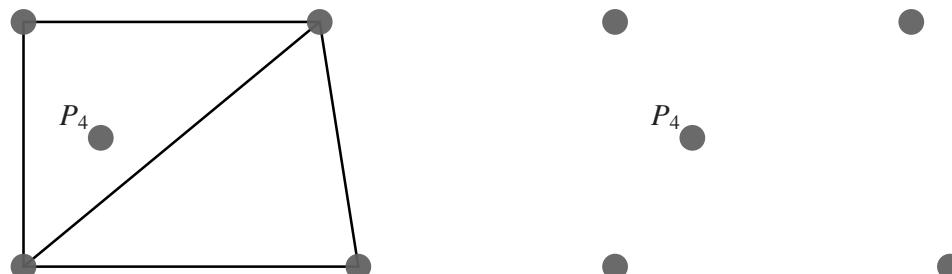
Q5 - Fonction posage(Ls1,Ls2,n,p)



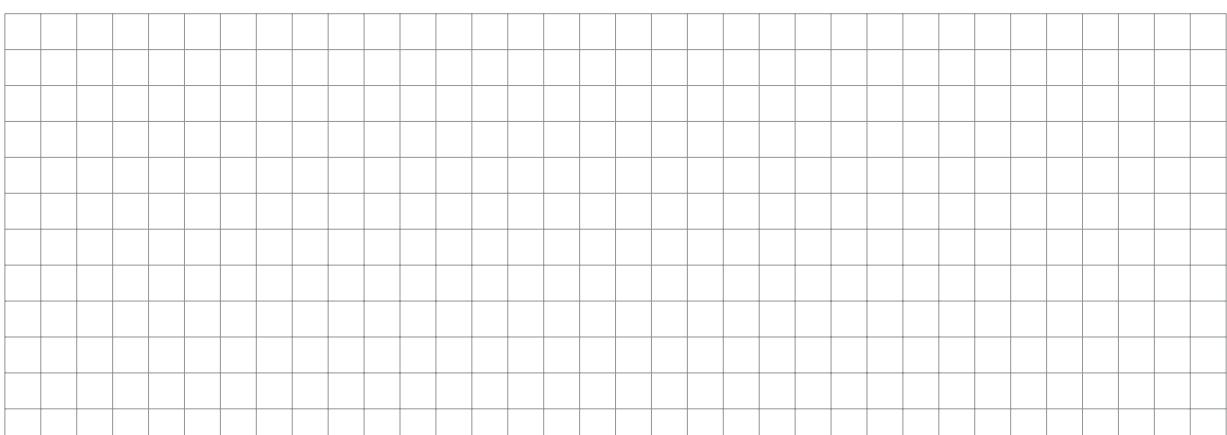
Q6 - Fonction posage_opt(Ls1,Ls2). Complexité



Q7 - Algorithme de Delaunay



Q8 - Fonction addT(indN,indT)



Q9 - Fonction flip(indT1,indT2)

Q10 - Fonction insideT(indN,T1)

Q11 - Intérêt de la ligne 3 de la fonction `checkedge(indN,indT)`. Instructions manquantes



**Numéro
d'inscription**

Numéro de table

Né(e) le

Nom : _____

Prénom : _____

Filière : PSI

Session : 2022

Épreuve de : INFORMATIQUE

Consignes

- Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer
 - Rédiger avec un stylo non effaçable bleu ou noir
 - Ne rien écrire dans les marges (gauche et droite)
 - Numérotter chaque page (cadre en bas à droite)
 - Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre

Emplacement
QR Code

Q12 - Fonction delaunay (noeud)

Q13 - Instructions pour tracer la solution analytique.

NE RIEN ÉCRIRE DANS CE CADRE

Q14 - Fonction $\phi(k, x)$

A large rectangular grid consisting of 20 columns and 25 rows of small squares, intended for sketching the graph of the function $\phi(k, x)$.

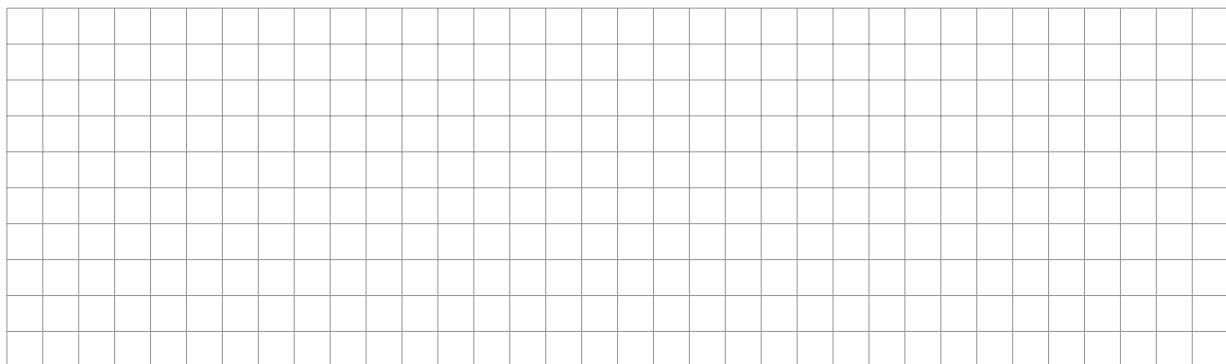
Q15 - Dérivées et matrice M

A large rectangular grid consisting of 20 columns and 25 rows of small squares, intended for working on derivatives and matrix M .

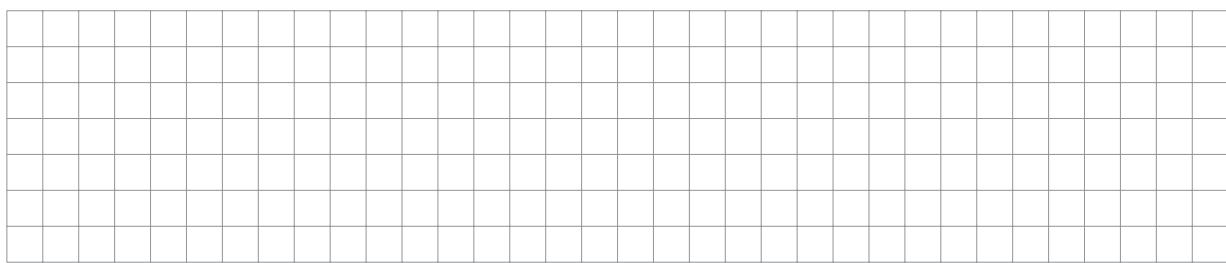
Q16 - Coefficients C et α

A large rectangular grid consisting of 20 columns and 25 rows of small squares, intended for working on coefficients C and α .

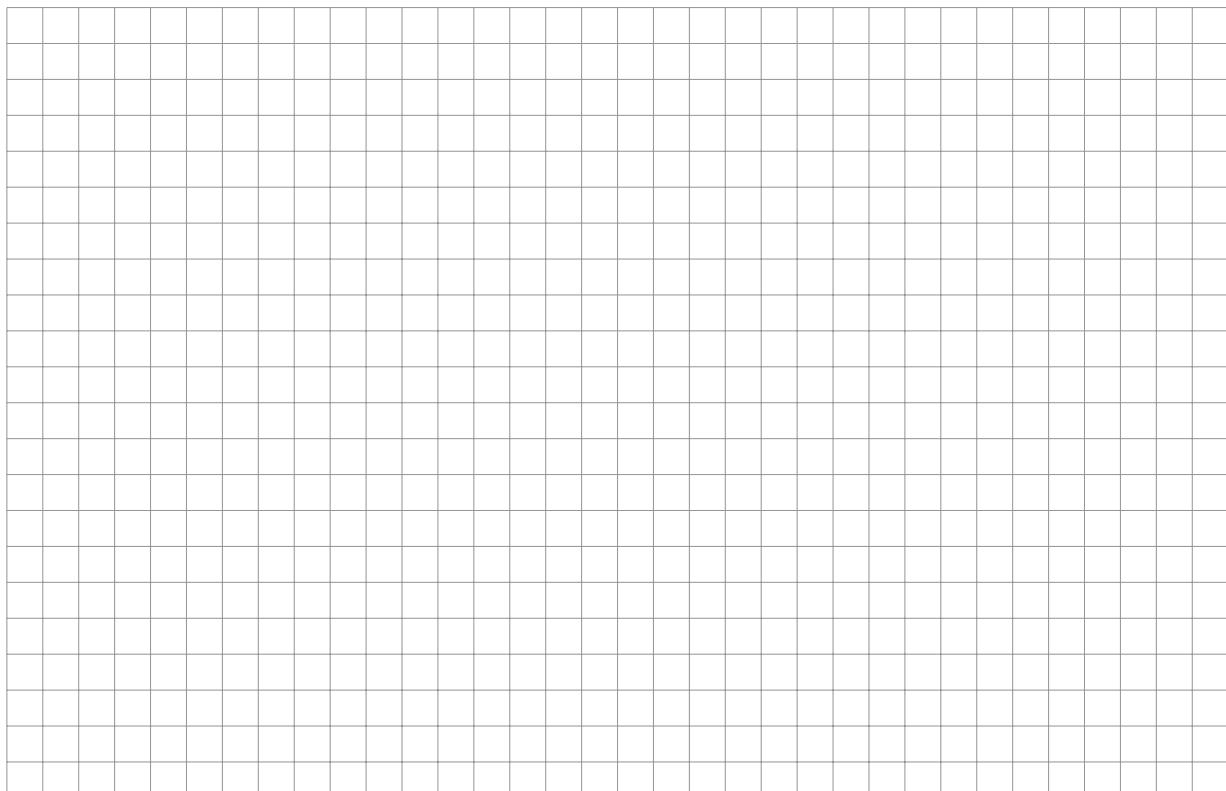
Q17 - Expressions de la matrice A et du vecteur B . Instructions pour créer A et B



Q18 - Intérêt des lignes 2 à 9



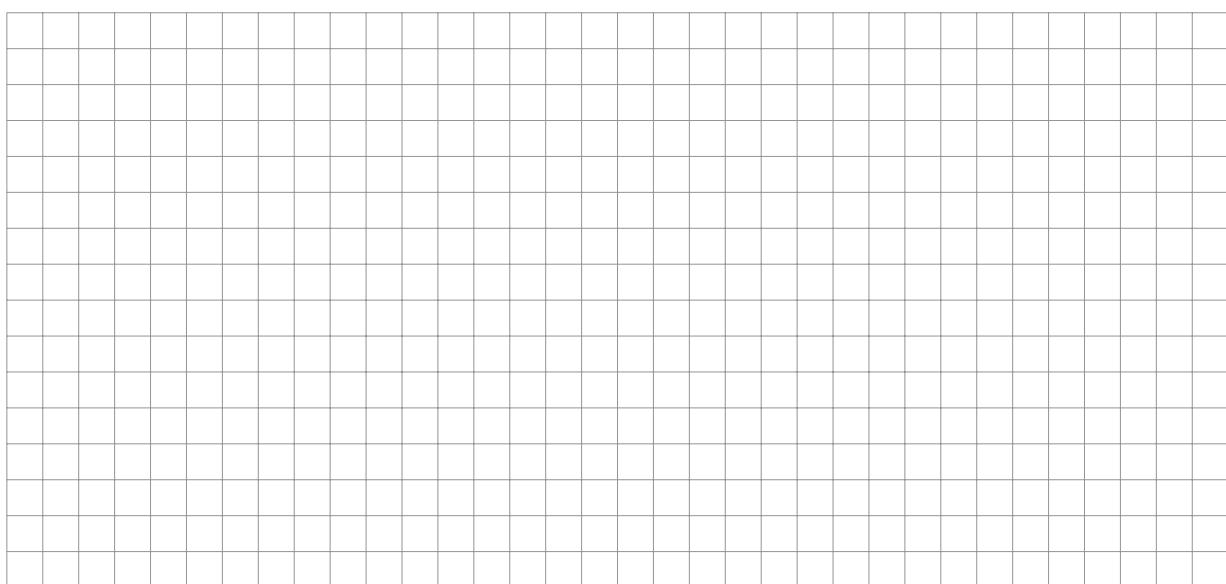
Q19 - Fonction pivot



Q20 - Fonction `combinaison(M,i,k,mu)`



Q21 - Construction du vecteur \mathbf{X}



Q22 - Instruction créant \mathbf{U} . Fonction `uEF(x)`



 CONCOURS COMMUN INP	Numéro d'inscription <input style="width: 50px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/>	Nom : _____
Numéro de table <input style="width: 50px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/>		Prénom : _____
Né(e) le <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/> <input style="width: 20px; height: 30px; border: 1px solid black; margin-bottom: 10px;" type="text" value=""/>		
Emplacement QR Code	Filière : PSI	Session : 2022
Épreuve de : INFORMATIQUE		
Consignes	<ul style="list-style-type: none"> • Remplir soigneusement l'en-tête de chaque feuille avant de commencer à composer • Rédiger avec un stylo non effaçable bleu ou noir • Ne rien écrire dans les marges (gauche et droite) • Numérotter chaque page (cadre en bas à droite) • Placer les feuilles A3 ouvertes, dans le même sens et dans l'ordre 	

PSI5IN

Q23 - Posage correct

Q24 - Choix de l'ordre de grandeur du nombre minimal d'éléments du maillage

Q25 - Seuil observé pour 5 000 éléments

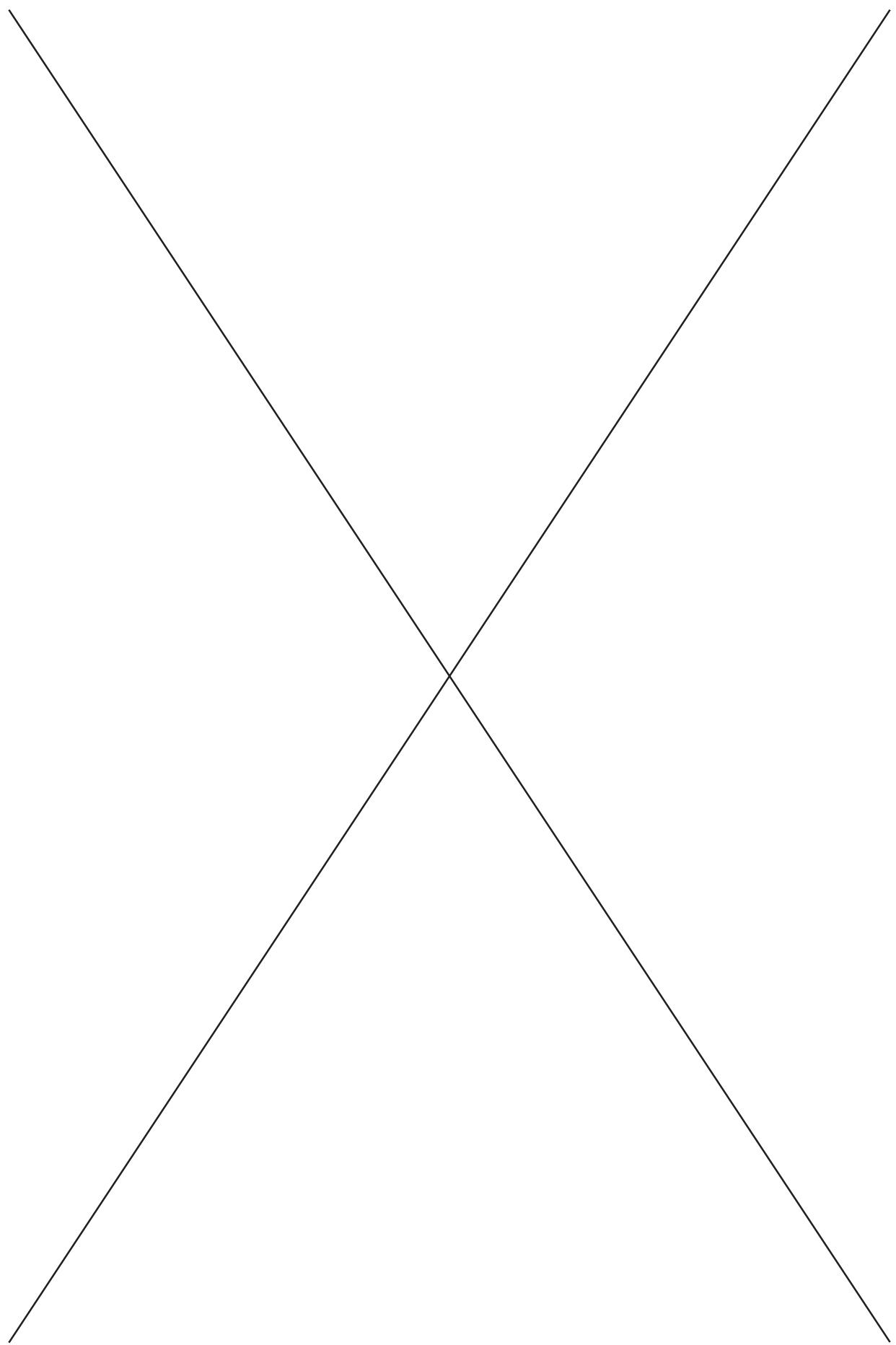
NE RIEN ÉCRIRE DANS CE CADRE

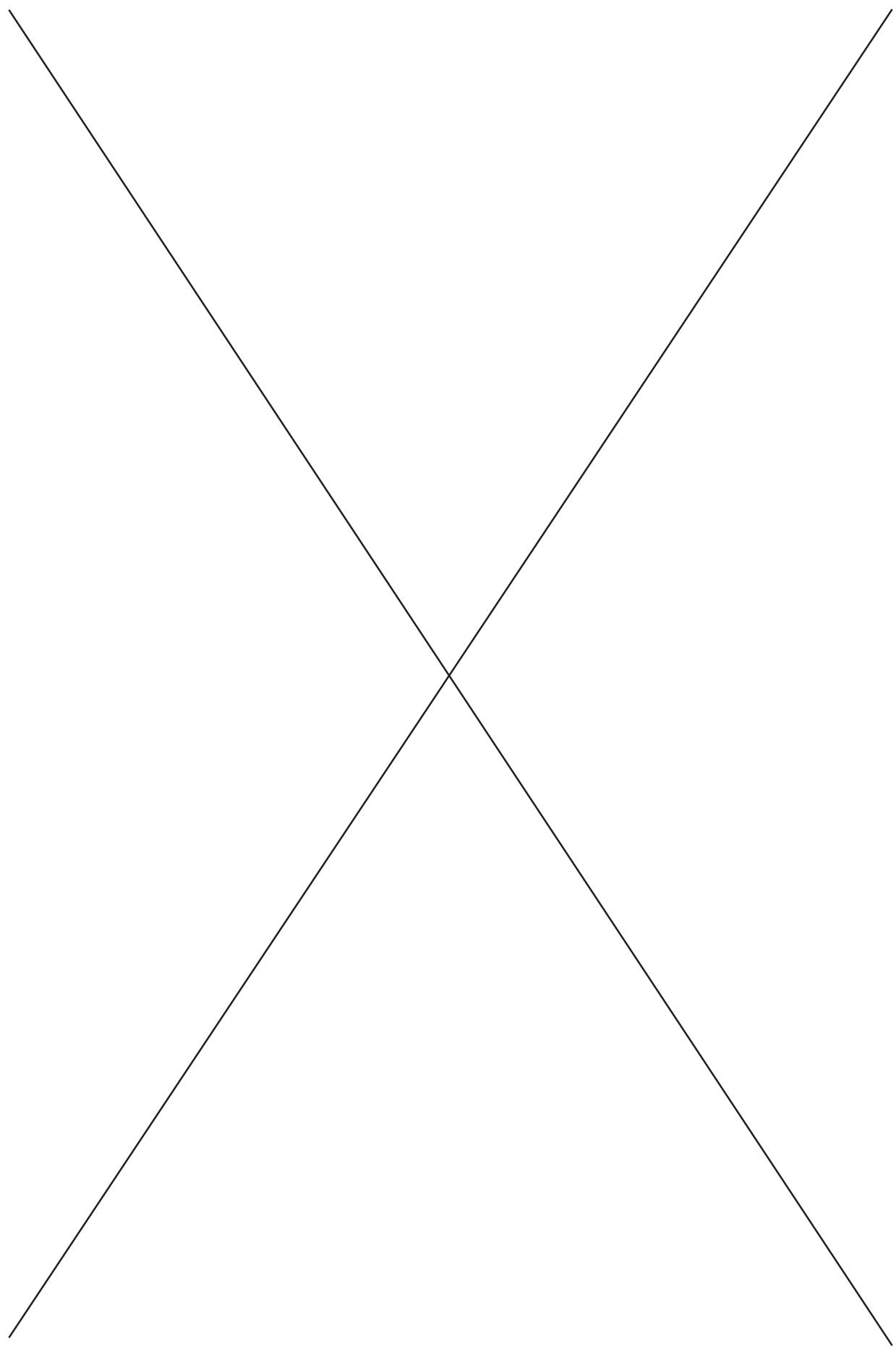
ANNEXE

Rappels des syntaxes en Python

Remarque : sous Python, l'import du module numpy permet de réaliser des opérations pratiques sur les tableaux : `from numpy import *`. Les indices de ces tableaux commencent à 0.

	Python
tableau à une dimension	<code>L=[1,2,3]</code> (liste) <code>v=array([1,2,3])</code> (vecteur)
accéder à un élément	<code>v[0]</code> renvoie 1 (<code>L[0]</code> également)
ajouter un élément	<code>L.append(5)</code> uniquement sur les listes
supprimer la première occurrence d'un élément d'une liste	<code>L.remove(element)</code>
calculer le produit vectoriel entre 2 vecteurs coplanaires	<code>cross([x1,y1],[x2,y2])</code> donne $x_1y_2 - x_2y_1$
évaluer la racine carrée d'un nombre flottant	<code>sqrt(4)</code>
évaluer la valeur absolue d'un nombre flottant	<code>abs(-2)</code>
tableau de 0 (2 lignes, 3 colonnes)	<code>zeros((2,3))</code>
séquence équirépartie quelconque de 0 à 10.1 (exclu) par pas de 0.1	<code>arange(0,10.1,0.1)</code>
séquence de 100 valeurs équiréparties de 0 à 10.0 (inclus)	<code>linspace(0,10,100)</code>
affiche y en fonction de x où x et y sont des listes ou vecteurs de même dimension.	<code>plot(x,y)</code>





SESSION 2022

TSI5IN



ÉPREUVE SPÉCIFIQUE - FILIÈRE TSI

INFORMATIQUE

Durée : 3 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

Le sujet est composé de sept parties et de trois annexes (voir fin du sujet).

Chaque Partie peut être traitée de manière indépendante.

Les fonctions Python créées dans une question (même non traitée) pourront être utilisées.

Représentation de données géolocalisées sur une carte numérique

Présentation du thème du sujet

Les fichiers contenant des photos possèdent des informations sur celles-ci, elles sont communément appelées données exif. Entre autres, on y trouve :

- la date où elle a été prise ;
- les coordonnées GPS de l'endroit où elle a été prise.

Le but de ce sujet est de faire un script qui va placer géographiquement sur une carte numérique les noms de fichiers photos archivés. On ajoutera des fonctionnalités à ce script au fur et à mesure de l'avancement du sujet, et la dernière partie permettra de travailler sur des informations concernant ces fichiers de photos en interrogeant une base de données. Dans tout le sujet, on supposera les modules Python importés.

Partie I - Géolocalisation de photos

Q1. Qu'affiche le script ci-dessous ?

Vous expliquerez votre résultat.

```
x='2020:06:12 21:12:40'  
y='2020:06:12 21:32:40'  
print(x>y)
```

Lors des questions **Q2**, **Q3** et **Q4**, les chaînes de caractères contiennent une fois et une seule le caractère '.'.

Exemple : mot1='chat.jpg', mot2='chien.jpeg', mot3='fichier.ipynb'

Les chaînes de caractères qui commencent par '.' sont exclues.

Exemple : '.ipynb', '.jpg' sont exclues.

Q2. Écrire une fonction nommée placeDuPoint qui prend en entrée une chaîne de caractères et qui retourne l'indice du point dans cette chaîne caractères.

Exemple : dans 'devoir.ipynb' le point a pour indice 6.

Q3. Écrire une fonction nommée coupeExtension qui prend en entrée une chaîne de caractères et qui retourne la chaîne de caractères précédent le caractère '.'.

Exemple : la fonction appliquée à 'devoir.ipynb' retourne 'devoir'.

Q4. Écrire une fonction nommée jpgTohtml qui prend en entrée une chaîne de caractères représentant le nom d'un fichier avec son extension et qui retourne le nom avec l'extension .html.
Exemple : la fonction appliquée à 'photo.jpeg' retourne 'photo.html'.

Partie II - Lecture de données exif

Dans cette partie, il est important de lire attentivement l'**annexe 2** partie **module exif**.

On rappelle l'utilisation de la fonction openImage :

```
my_image=openImage("photo.jpg")
```

La variable **my_image** permet alors d'accéder aux **données exif** et on fera référence à cette variable dans toute la suite du sujet.

La question suivante se réfère à la partie Coordonnées GPS- Règles de Conversion de l'**annexe 2**.

Q5. Écrire une fonction `convertToDecimal` qui prend en entrée un tuple (degré, minutes, secondes) et qui retourne le flottant correspondant.

On définit un type Point comme étant un tuple (`latitude`, `longitude`) de deux flottants signés.

Q6. Écrire une fonction `imageToGpsPoint` qui prend en entrée une variable `my_image` et qui retourne une variable de type Point.

Q7. Écrire une fonction `listeAvantMidi` qui prend en entrée une liste de photos (les photos sont représentées par leur nom qui est une chaîne de caractères) et qui retourne la liste des photos prises avant midi.

Partie III - Module Folium

Présentation du module folium

Python possède un module nommé **folium** qui facilite la visualisation des données.

Les données que nous allons manipuler sont des photos ayant des données exif, plus particulièrement des coordonnées GPS, puis nous allons mettre en place le script qui permet de placer leurs noms sur une carte numérique. On précise que l'on peut ajouter le script pour les visualiser, mais cela nécessite du code **Html**, ce qui n'est pas abordé dans ce sujet.

Chaque nom de photo sera positionné sur cette carte grâce à ses coordonnées GPS. Un marqueur (nommé **marker** dans le module folium) permet de repérer cette photo positionnée sur la carte et une option **popup** permet de nommer ce marqueur. La carte peut être centrée sur des coordonnées particulières et peut être plus ou moins "zoomée".

Par commodité et pour permettre une lecture fluide du document, nous utiliserons les anglicismes suivants :

- **marker** pour tout marqueur (ou symbole) placé sur la carte numérique à certaines coordonnées GPS ;
- **popup** : option qui permet de donner un nom au **marker** par le biais d'une chaîne de caractères.

Pour la création d'une carte numérique, on procède comme ci-dessous :

```
import folium #importation du module
location=(35.9279, -114.9721) #location est de type Point
#ce tuple représente le couple (latitude, longitude)
#Pour créer une carte vide centrée sur location:
carteTest=folium.Map(location,zoom_start=20) # l'option de zoom est ici à 20 arbitrairement
#Pour ajouter un marker à la carte créée ici nommée carteTest
#Ce marker est placé aux coordonnées location:
folium.Marker(location, popup='Las Vegas').add_to(carteTest)
#popup donne un nom au marker, ici 'Las Vegas', sur la carte
#Pour créer et sauvegarder la carte au format html
carteTest.save('MaCarte.html') #Le nom de sauvegarde est ici Macarte.html
```

Dans tout ce qui suit, le module folium sera supposé importé.

On rappelle que chaque carte doit être centrée sur un point dont les coordonnées GPS sont connues.

- Q8.** Créer une fonction nommée carteVide qui prend en entrée une donnée de type Point nommé centre et qui retourne une carte vide créée avec un zoom de 20, et qui doit être centrée sur centre.
- Q9.** Écrire le script de la fonction nommée transfertTocarte, qui prend en entrée une carte vide appelée carte, un objet image de type my_image et une chaîne de caractères (le popup à placer) appelée popupPassé et qui ajoute à la carte le marker au coordonnées du Point et le popup. Cette fonction retourne la carte ainsi modifiée.

L'exemple de code ci-dessous permet d'ajouter plusieurs markers (ayant des noms et/ou des coordonnées GPS différents) à une carte :

```
centre=(46.87, 4.00261)
c=folium.Map(location=centre,zoom_start=20)
folium.Marker((46.877, 4.00261),popup="LosAngeles1").add_to(c)
folium.Marker((46.87, 4.003),popup="LosAngeles2").add_to(c)
folium.Marker((46.88, 4.003),popup="LosAngeles3").add_to(c)
c.save('maCarte.html') # sauvegarde de la carte
#le nom doit avoir l'extension .html pour obtenir une page web lisible
```

- Q10.** Créer une fonction transfertListeTocarte, qui prendra en paramètres d'entrée un point de type Point appelé centre, une liste de photos appelée listePhotos. Cette fonction crée une carte centrée sur centre et y ajoute chaque nom de la liste listePhotos. Le script doit de plus répondre aux exigences suivantes :
- chaque popup aura pour nom le nom ajouté ;
 - la fonction retournera la carte créée.

- Q11.** Écrire le code Python, qui à partir d'une liste nommée listePhotos, crée une carte sauve au nom de "Las Vegas" et qui ajoute tous les noms par le biais de markers. Cette carte sera centrée sur le premier terme de la liste.

Exemple de liste de photos :

```
listePhotos=['Venitian.jpeg','Bellagio.jpeg','Palazzo.png']
```

III.1 - Coordonnées GPS

On rappelle qu'un objet de type Point est un tuple de deux nombres décimaux représentant une latitude et une longitude dans cet ordre. Ainsi, `gps=(35.9279, -114.9721)` est le tuple (latitude, longitude) des coordonnées GPS de Las Vegas en flottants signés.

On donne deux Point appelés inf et sup et on suppose que le segment formé par ces deux Point est la diagonale d'un rectangle non aplati où le coin inférieur gauche est inf et le coin supérieur droit est sup.

On ne souhaite positionner que les noms des photos situées dans ce rectangle.

Dans cette sous-partie, nous parlerons d'une liste de chaînes de caractères qui représentent des noms de photos (type 'photo01.jpg') et ayant des données exif.

- Q12.** Écrire une fonction testRectangle qui a deux paramètres d'entrée de type Point nommés point1, point2 et qui retourne le booléen :
- Vrai si point1 et point2 forment un rectangle non aplati, dont point1 sera le coin inférieur gauche et point2 le coin supérieur droit ;
 - Faux sinon.

Q13. Écrire une fonction `milieu` qui a deux paramètres d'entrée de type `Point` nommés `point1`, `point2` et qui retourne un `Point` qui est le milieu du segment `point1 - point2`.

Q14. On suppose que `inf`, `sup` sont deux objets de type `Point` qui forment un rectangle non aplati. Écrire une fonction `intRectangle` qui a trois paramètres d'entrée de type `Point` nommés `point`, `inf`, `sup`, et qui retourne un booléen. Ce booléen sera vrai si `point` appartient au rectangle formé par `inf`, `sup` et faux sinon.
On précise que le bord du rectangle est exclu.

III.2 - Ajout de markers à une carte numérique

On reprend la notion de rectangle présentée dans la **Q14**.

On dispose d'une liste de noms de photos, les `markers` seront des noms de photos.

Parmi ces photos on souhaite sélectionner uniquement celles dont les coordonnées GPS sont incluses strictement dans le rectangle délimité par `inf` et `sup`, et positionner leur nom sur une carte numérique.

On rappelle que pour sauvegarder une carte (de nom '`nomDeLaCarte`'), il suffit d'utiliser le code :

```
carte.save('nomDeLaCarte'+'.html')
```

L'extension '`.html`' permet de rendre la carte interprétable par un moteur de recherche.

Le code Python pour l'affichage dans une page web sera alors :

```
webbrowser.open('nomDeLaCarte.html')
```

Q15. Créer la fonction `listeToMap` dont les paramètres d'entrée sont :

- une liste de photos au format chaîne de caractères (exemple : '`Bellagio.jpeg`'), nommée `listePhotos`;
- `inf` et `sup` de type `Point`;
- `titre` qui sera le nom de la carte sous forme de chaîne de caractères (sans extension);

Cette fonction crée une carte dont le nom est `titre`, et y positionne uniquement le nom des photos sélectionnées. La carte sera centrée sur le milieu du segment `inf - sup`.

Cette fonction retourne `False` si le nombre d'éléments ajoutés à la carte est nul, sinon elle sauvegarde la carte et la retourne.

Partie IV - Ajout d'une ligne entre chaque marker

On dispose d'une liste de noms de photos que l'on veut placer sur une carte numérique, les `markers` seront des noms de photos.

Chaque `marker` sera relié à un autre `marker` par une ligne dessinée sur la carte.

Chaque ligne dessinée est faite suivant l'ordre d'apparition des noms dans la liste.

Pour faire une ligne qui relie un `marker` à un autre `marker` dans l'ordre d'une liste, on procède comme ci-dessous.

Attention, dans l'exemple, les éléments de la liste sont des tuples de coordonnées GPS.

```
p1=[39.900908, -73.040335]
p2=[40.768571, -73.861603]
p3=[41.011522, -73.960004]
centre=[40.70000, -73.70000]
coordinates=[p1,p2,p3]
m=folium.Map(location=centre,zoom=20)
aline=folium.PolyLine(locations=coordinates,weight=2,color='blue')
# Ajout d'une ligne polygonale p1-p2-p3 de couleur bleue largeur 2
aline.add_to(m)
```

Q16. Créer la fonction appelée listeTomapLine dont les paramètres d'entrée sont :

- une liste de noms de photos au format chaîne de caractères (exemple : 'Bellagio.jpeg'), nommée listePhotos;
- inf et sup de type Point;
- titre qui sera une chaîne de caractères (sans extension) représentant le nom de la carte;

Cette fonction crée une carte dont le nom est titre, et y positionne uniquement le nom des photos si elles sont dans le rectangle délimité par inf et sup, et relie chaque élément consécutif ajouté à la carte par une ligne rouge de largeur 1.

La carte sera centrée sur le milieu du segment inf - sup.

Cette fonction retourne False si le nombre d'éléments ajoutés à la carte est nul, sinon elle sauvegarde la carte et la retourne.

Q17. Écrire le script qui permet de dessiner un rectangle bleu (épaisseur 1) sur la carte précédente dont la diagonale est inf-sup. Le nom de sauvegarde de cette nouvelle carte est monRectangle.

On donne la liste ordonnée des points de ce rectangle à utiliser dans cette question :

```
coordinates=[inf,inf2,sup,sup2,inf]
```

Partie V - Recherche de photos dans un dossier

On suppose que les photos ont été stockées dans un dossier de sauvegarde. Mais avec le temps, ce dossier s'est rempli. Maintenant, il contient des photos mais aussi des dossiers de photos ou des dossiers de dossiers de photos, etc.

Le module os permet de gérer ces dossiers et nous allons utiliser en particulier la fonction os.listdir qui prend en paramètres une chaîne de caractères (le nom d'un dossier) et qui retourne la liste de noms de fichiers avec leur extension ou de dossiers.

Par exemple : l'appel de la fonction os.listdir('Photos') retourne la liste des noms des fichiers et des sous-dossiers du dossier appelé ici Photos.

Exemple :

```
import os
liste=os.listdir('Photos') # os.listdir ne peut prendre en entrée qu'un répertoire
print(liste)
>>>['Photos2019','Photos2018','photo1.jpg','photo2.jpg','photo3.jpg']
liste2=os.listdir('dossier\dossier2')
print(liste2)
>>>['dossier 3 bis','dossier 3 ter','dossier3','photo2_1.png','photo2_2.jpg']
```

On précise que si un dossier nommé 'Photos2019' est un sous-dossier du dossier nommé 'Photos' pour accéder aux éléments de 'Photos2019' il faut donner le chemin 'Photos\Photos2019'. Ainsi, si 'Photos2019' contient le dossier 'Photos2019Janvier', alors pour accéder aux éléments de 'Photos2019Janvier', il faut écrire : 'Photos\Photos2019\Photos2019Janvier'.

On rappelle que le nom d'un fichier contient une seule et unique fois le caractère '.' et que le nom d'un dossier ne comporte pas ce caractère. Dans les **Q17**, **Q18** et **Q19**, les chaînes de caractères utilisées ne peuvent être que des noms de dossiers ou des noms de fichiers.

- Q18.** Écrire le code Python qui permet de lister toutes les photos de la liste données ci-dessous.
 Avec cette liste on n'a qu'une vue partielle des données, mais on sait que chaque dossier de cette liste ne contient que des photos et rien d'autre.

```
donnees=['photo.jpeg','dir1','dir2','photo2.jpeg',...,'photo1211.png','dir4110']  
#'dir1','dir2',...,'dir4110' sont des dossiers
```

Conseil : pour ajouter un élément, ou une liste à une liste, se référer à **l'annexe 1**.

- Q19.** Que contient liste après l'exécution du code Python ci-dessous :

```
# mot est une variable qui contient une chaîne de caractères  
# qui est le nom d'un dossier ou d'une photo  
monDir='rep' # le dossier initial  
if '..' not in mot:  
    monDir=monDir+'\''+mot  
liste=os.listdir(monDir)
```

- Q20.** Écrire une fonction listerFichiers qui prend en entrée une chaîne de caractères (on suppose que c'est un nom de dossier) et qui retourne la liste des fichiers dans ce dossier.
 Rappelons que ce dossier peut contenir des photos mais aussi des dossiers de photos ou des dossiers de dossiers, etc.

Partie VI - Une fonction mystère et son application

On donne ci-dessous la fonction mysteryMachine :

```
def mysteryMachine(liste):  
    x=1  
    while(x<len(liste)):  
        nom=liste[x]  
        i=x  
        while(openImage(nom).datetime<openImage(liste[i-1]).datetime): # Question 2  
            liste[i]=liste[i-1]  
            i=i-1  
        if(i==0): # Question 3  
            break  
        liste[i]=nom  
        x+=1  
    return liste
```

- Q21.** 1. Quel type de données attend cette fonction (réponse détaillée attendue)?
 2. Que fait la boucle while numérotée # Question 2.
 3. Justifiez le test numéroté # Question 3.
 4. Quel algorithme de tri est en jeu dans cette fonction ?
 5. Quelle est la complexité de cet algorithme ?
 6. Que fait cette fonction ?

- Q22.** Dans le script suivant, inf et sup sont les deux Point de la diagonale d'un rectangle non aplati et inf2, sup2 sont les deux sommets manquants du rectangle.

Décrire de manière détaillée ce que fait ce script :

```
nom=input("Nom dossier?")
liste=listerFichiers(nom)
liste=mysteryMachine(liste)
titre=jpgTohtml(liste[0])
carte=listeTomap(liste,inf,sup,titre)
coordinates=[inf,inf2,sup,sup2,inf]
aline=folium.PolyLine(locations=coordinates,weight=1,color='blue')
aline.add_to(carte)
carte.save(titre)
```

Partie VII - Interrogation de Bases de données SQL

SQLite permet de créer et de gérer des bases de données. Le module Python `pysqlite3` permet d'interagir avec des bases de données SQLite.

Les tables utiles pour répondre aux questions sont en **annexe 3**. On remarquera que dans la table Photos, les latitudes et longitudes sont des entiers : une division par 10 000 permet d'avoir la latitude et la longitude réelle.

On notera que dans la table Photos, Date correspond à la date de création (année/mois/jour) de photos présentes dans le dossier (colonne nommée Dossier). On précise qu'à un nom de photo correspond une unique photo et que chaque dossier ne contient que des photos.

Q23. Écrire la requête SQL qui permet d'avoir le nom, la latitude et la longitude des photos faites le 2020/06/16.

Q24. Écrire la requête SQL qui permet d'avoir le nom des photos faites le 2020/06/16 et entre les latitudes 35935 et 35940.

Q25. Écrire la requête SQL qui permet de compter le nombre de photos situées dans le Dossier C :\Images\Las Vegas\Bellagio.

Q26. Écrire la requête SQL qui permet de trouver le nom des photos prises le 2020/06/17 et situées dans le Dossier C :\Images\Las Vegas\Bellagio.

On donne le prototype de la fonction `requete_to_liste` écrite en Python qui attend en entrée une requête SQL au format chaîne de caractères : tout simplement une requête SQL usuelle “entourée de guillemets”.

```
def requete_to_liste(sql):
    ...
    return liste
```

Cette fonction retourne une liste d'informations, par exemple une liste de noms de photos, qui répond à la requête.

Q27. En utilisant des fonctions définies précédemment, écrire le script Python qui permet de savoir si, pour le Dossier C :\Images\Las Vegas \Bellagio, de la table Dir, toutes les photos qui y sont enregistrées, dont la date de prise de vue est comprise entre le 2020/06/15 et le 2020/06/21, sont dans la base Photos. Le script devra afficher la liste des photos ordonnées par date qui ne sont pas dans la base Photos.

ANNEXE 1

Rappels sur Python

Rappels sur les listes

```
#On suppose que maListe est une liste
len(maListe) # donne la longueur de maListe
maListe.append(objet) # ajoute l'objet à maListe
#il sera le dernier élément une fois ajouté
#soit autreListe une liste
maListe.append(autreListe)#ajoute tous les éléments de autreListe à maliste
#autreListe est non modifiée
```

Rappels sur les tuples

Un tuple est la donnée d'un n-uplet non mutable (qui ne peut pas être modifié).

```
monTuple=(donnee1,donnee2)
monTuple[0]# permet d'accéder à donnee1
monTuple[1]# permet d'accéder à donnee2
#on peut aussi accéder aux données en faisant
val1,val2=monTuple
print(val1)
>>>donnee1
print(val2)
>>>donnee2
#Pour parcourir un tuple :
for x in monTuple:
    print(x)
>>>donnee1
>>>donnee2
```

Une fonction peut retourner un tuple dont on peut en extraire le contenu directement.

Exemple :

```
def retourneTuple(x,y):
    return (x+y,x-y)
som,diff=retourneTuple(10,3)
print(som)
>>>13
print(diff)
>>>7
```

Rappels sur les chaînes de caractères

Parcours d'une chaîne :

```
mot="test"
for lettre in mot:
    print(lettre)
#Résultat console :
T
e
s
t
```

Rappels sur le Slicing :

```
print(mot[:3])
>>>Bon
print(mot[3:])
>>>jour
```

Rappel sur la fonction len :

```
print(len(mot))
>>>7
```

ANNEXE 2

Présentation des données exif

Module exif

Chaque fichier de photo numérique contient des informations appelées données exif (exif signifie « EXchangeable Image File » ou fichier d'échange de données). Ces informations, créées par l'appareil photo lors de la prise de vue, sont stockées dans le fichier généré par l'appareil.

Voici deux exemples d'informations accessibles :

- . Date de la prise de vue.
- . Coordonnées GPS du lieu de la prise de vue.

Les données exif sont accessibles dans un script Python en important le module exif, puis en accédant à Image dans ce module. À l'issue du script suivant, la variable my_image permet d'accéder aux données exif :

```
from exif import Image
with open("photo.jpg", 'rb') as imageFile:
    my_image= Image(imageFile)
```

Afin de faciliter le codage, on donne ci-dessous la fonction openImage qui sera utilisée lors de ce sujet :

```
def openImage(nom): #nom est une chaîne de caractères qui représente ici une photo
    with open(nom, 'rb') as imageFile:
        return Image(imageFile)
    imageFile.close()
#Exemple d'appel de cette fonction :
my_image=openImage("photo.jpg")
```

Suite à l'appel de la fonction, la variable my_image permet d'accéder aux données exif.

La liste des données exif qui nous intéressent est :

```
my_image.datetime # une date au format chaîne de caractères
my_image.gps_longitude # un tuple de trois nombres
my_image.gps_longitude_ref # une chaîne de caractères 'E' ou 'W'
my_image.gps_latitude # un tuple de trois nombres
my_image.gps_latitude_ref # une chaîne de caractères 'N' ou 'S'
```

Coordonnées GPS - Règles de conversion

Le module folium, dont on a besoin ici, utilise des coordonnées gps au format tuple (latitude, longitude) où les deux valeurs sont en décimales signées suivant l'orientation Nord-Sud ou Est-Ouest. Les données gps issues des données exif d'une image sont au format tuple (degré, minutes, secondes). Il faut donc procéder à une conversion.

On donne la règle de conversion suivante :

1 degré = 1

1 minute = 1/60

1 seconde = 1/3600

L'orientation pour la latitude est 'N' ou 'S' et pour la longitude 'E' ou 'W'.

Les valeurs décimales sont signées : négative si on est 'W' ou 'S' et positive sinon. L'accès à cette orientation est donnée par le code Python ci-dessous :

```
my_image.gps_latitude_ref # donne 'N' ou 'S'  
my_image.gps_longitude_ref # donne 'E' ou 'W'
```

Un exemple :

```
my_image.gps_latitude # donne par exemple (36,10,11.78)  
my_image.gps_longitude # donne par exemple (115,8,23.38)  
my_image.gps_latitude_ref # donne par exemple 'N'  
my_image.gps_longitude_ref # donne par exemple 'W'
```

Ce qui donne au format décimal avec l'orientation Nord et Ouest : 36.169939, -115.13983.

Date de prise de vue

La date de prise de vue est une chaîne de caractères de longueur 19, le format est :

```
#année:mois:jour heure:minute:seconde avec un seul espace entre jour et heure  
'1967:06:17 11:15:00'  
'2020:12:24 23:59:59'
```

Pour accéder à l'information de date, il suffit d'écrire le script :

```
var_date=my_image.datetime  
#var_date sera une chaîne de caractères qui contient les informations de date.
```

ANNEXE 3

Tables SQL

Nom	Date	Latitude	Longitude	Id
Photo10	2020/06/11	35937	-1149686	1
Photo11	2020/06/22	35937	-1149685	2
Photo24	2020/06/17	35937	-1149686	3
Photo15	2020/06/17	35938	-1149686	4
Photo33	2020/06/16	35939	-1149688	5
:	:	:	:	:

Table Photos

Id	Dossier	NomPhoto
1	C :\Images\LasVegas\Bellagio	Photo10
2	C :\Images\LasVegas\Palazzo	Photo11
3	C :\Images\LasVegas\Flamingo	Photo24
:	:	:
10	C :\Images\LasVegas\Flamingo	Photo40
11	C :\Images\LasVegas\Palazzo	Photo30
12	C :\Images\LasVegas\Flamingo	Photo15
:	:	:

Table Dir

Table	Champ	Type
Photos	Nom	Chaîne de caractères
Photos	Date	Chaîne de caractères
Photos	Latitude	Entier
Photos	Longitude	Entier
Photos	Id	Entier
Dir	Id	Entier
Dir	NomPhoto	Chaîne de caractères
Dir	Dossier	Chaîne de caractères

Type des données par Table

FIN



ÉPREUVE MUTUALISÉE AVEC E3A-POLYTECH

ÉPREUVE SPÉCIFIQUE - FILIÈRE PC

MODÉLISATION DE SYSTÈMES PHYSIQUES OU CHIMIQUES

Durée : 4 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

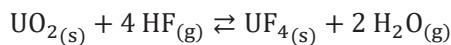
Les calculatrices sont autorisées.

Le sujet est composé de trois parties
et d'une Annexe en fin d'énoncé.

Étude d'une réaction à solide consommable

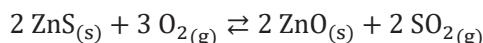
Présentation du problème

Les réactions à solide consommable sont courantes dans les procédés de transformation de la matière. On peut citer la réaction du dioxyde d'uranium (UO_2) avec l'acide fluorhydrique (HF) pour produire le tétrafluorure d'uranium (UF_4) selon la stoechiométrie suivante :



Cette réaction est impliquée dans la conversion du dioxyde d'uranium en hexafluorure d'uranium (UF_6) qui est utilisé dans les deux principaux procédés d'enrichissement de l'uranium (la diffusion gazeuse et l'ultracentrifugation) en raison de son point triple à 64 °C (à 150 kPa).

Un autre exemple est la réaction d'oxydation du sulfure de zinc en présence d'air pour former l'oxyde de zinc, un matériau semi-conducteur qui pourrait remplacer le dioxyde de titane dans les cellules photovoltaïques. La réaction s'écrit :



Dans les deux cas, la réaction met en jeu une phase fluide et une phase solide. Une particularité de ce type de réactions est la formation d'un produit solide poreux qui remplace progressivement le réactif initial. Dans le cas de particules sphériques, on observe un déplacement du front de réaction (l'interface entre le réactif solide non consommé et le produit solide) de l'extérieur de la particule vers le centre en fonction du temps (**figure 1**).

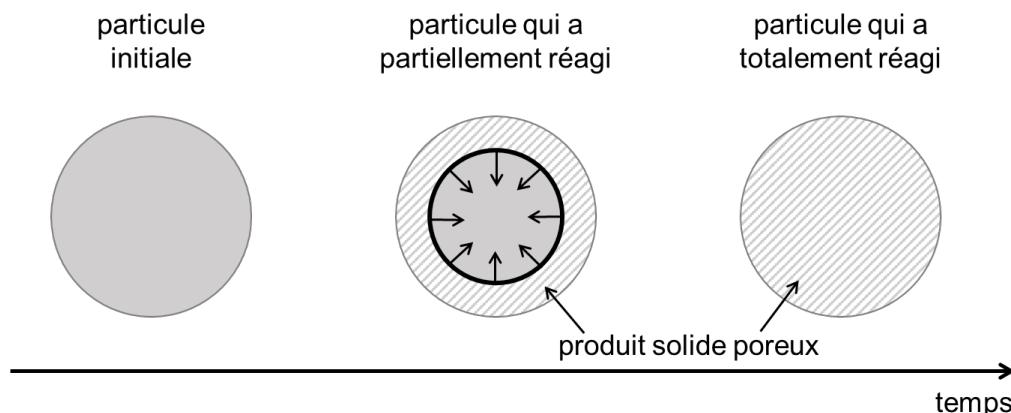


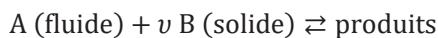
Figure 1 - Représentation schématique de l'évolution du front de réaction en fonction du temps

Les phénomènes physiques et chimiques mis en jeu lors de la réaction sont :

- le transport du fluide réactif à travers la couche limite externe des particules (généralement par convection),
- le transport du fluide réactif à travers la couche de produit solide poreux par diffusion dans les pores,
- la réaction chimique qui est localisée au niveau du front de réaction,
- le transport du produit fluide (s'il existe) dans les pores, puis à travers la couche limite externe.

Le modèle qui permet de représenter ces phénomènes porte le nom de *modèle à cœur rétrécissant*. Dans une version simple de ce modèle, on suppose que le produit poreux a une densité similaire à celle du réactif et que par conséquent le volume final de la particule qui a totalement réagi est le même que celui de la particule initiale. Dans le cas général, la consommation des particules est contrôlée par les trois premiers phénomènes cités précédemment. Cependant, dans certains cas limites, il se peut que l'un des trois phénomènes soit beaucoup plus lent que les deux autres et c'est lui qui va imposer la vitesse de consommation des particules. On parle alors de *phénomène limitant*. Par exemple, si la réaction chimique est très lente par rapport aux phénomènes de transport, ce sera le phénomène limitant ; la vitesse de consommation des particules sera contrôlée par la réaction chimique et on parlera de régime chimique.

Dans la suite, pour généraliser, la réaction entre les particules solides (B) et la phase fluide (A) sera écrite de la manière formelle suivante où ν est le coefficient stœchiométrique devant le solide :



Le but de cette épreuve est de mettre en équation le modèle à cœur rétrécissant permettant de prédire la consommation des particules en fonction du temps.

La **première partie** consiste, dans un premier temps, à établir l'équation aux dérivées partielles régissant l'évolution de la concentration du fluide A dans la couche de solide poreux B ; puis, dans un second temps, une version simplifiée du modèle, permettant d'obtenir des résultats facilement sans avoir recours à la résolution de l'équation aux dérivées partielles, sera déterminée à l'aide d'hypothèses sur le fonctionnement du système physico-chimique.

La **deuxième partie** est relative à la résolution numérique de l'équation aux dérivées partielles dans le cas général (sans hypothèses simplificatrices).

La **dernière partie** traite de la comparaison des résultats obtenus par les deux méthodes de résolution du problème.

Partie I - Mise en équation du modèle à cœur rétrécissant

L'objectif de cette première partie est d'établir l'équation différentielle qui régit l'évolution de la concentration du fluide A dans la couche de solide poreux B en fonction du temps et de l'espace. Une première version simplifiée du modèle à cœur rétrécissant sera alors déterminée.

I.1 - Étude de la diffusion du fluide dans la couche de produit poreux

Le transport du fluide dans la couche de produit de la réaction a lieu par diffusion dans les pores de ce matériau poreux. La porosité d'un matériau est définie comme le volume des pores divisé par le volume total de matériau poreux. Le volume de matériau poreux inclut le volume de solide (appelé volume structurel) et le volume des pores. La porosité est donc la fraction volumique de vide dans le matériau.

Une étude au laboratoire a permis de déterminer les masses volumiques structurelle (sans la porosité), ρ_s , et apparente (tenant en compte la porosité), ρ_{app} , du produit poreux formé au cours de la réaction. Les valeurs obtenues sont $\rho_s = 2\ 500 \text{ kg} \cdot \text{m}^{-3}$ et $\rho_{app} = 1\ 250 \text{ kg} \cdot \text{m}^{-3}$.

Q1. Justifier que $\rho_s > \rho_{app}$.

Donner les valeurs théoriques minimale et maximale de la porosité.

Calculer la porosité, ϵ , du produit à partir des valeurs des masses volumiques structurelle et apparente mesurées au laboratoire.

Pour simplifier, on considère que la particule est un parallélépipède rectangle comme représenté sur le schéma de la **figure 2**. L'épaisseur de la particule est de $2e$ et on suppose que la longueur L et la largeur l sont très grandes devant l'épaisseur.

On note S la section latérale correspondant au produit $L \times l$ et on considère que la particule réagit avec le fluide uniquement par les deux faces latérales de surface S comme représenté sur le schéma de la **figure 2**.

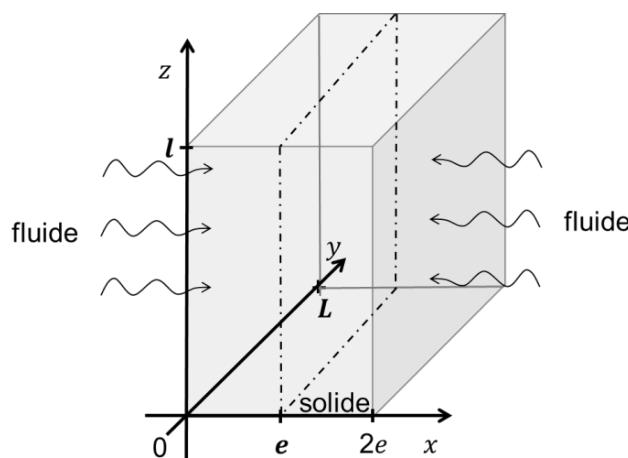


Figure 2 - Schéma de la particule de solide d'épaisseur $2e$

En représentation cartésienne, on note $\vec{j}_N(x, y, z, t)$ le vecteur densité de flux de particules associé aux phénomènes de diffusion que l'on étudie.

Q2. Indiquer les conséquences de l'hypothèse L et $l \gg 2e$ sur la direction du déplacement du front de réaction et sur les invariances du système.

Expliquer pourquoi on peut restreindre cette étude à l'intervalle $[0, e]$.

I.2 - Établissement de l'équation de la diffusion de A dans la couche de produit poreux B

Les phénomènes de diffusion qui nous intéressent sont régis par les équations suivantes :

$$\vec{J}_N(x, t) = -D \overrightarrow{\text{grad}}(n(x, t)) \quad (\text{loi de Fick}) \quad \text{et} \quad \frac{\delta N}{dt} = \iint \vec{J}_N d\vec{S} \quad (\text{flux de } \vec{J}_N)$$

Q3. Rappeler la signification physique de la loi de Fick.

Préciser le nom et les unités des grandeurs qui la constitue.

On considère un volume élémentaire de la couche de produit poreux de section S et d'épaisseur dx (**figure 3a**). La **figure 3b** est une vue de profil de la **figure 3a**. La **figure 3c** est également une vue de profil sur laquelle le volume structurel et le volume des pores ont été séparés de manière fictive.

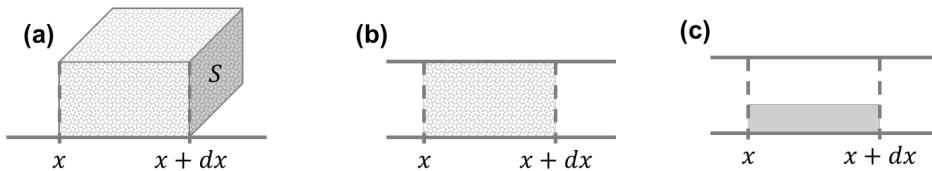


Figure 3 - Volume élémentaire de la couche de produit poreux de section S et d'épaisseur dx

(a) Vue en 3 dimensions (b) Vue de profil

(c) Vue de profil avec séparation fictive du volume structurel et du volume des pores

Q4. Reproduire sur votre copie le schéma du volume élémentaire de la **figure 3c** en y faisant figurer les flux de matière aux abscisses x et $x + dx$.

Q5. Réaliser un bilan de matière sur ce volume élémentaire traduisant la conservation du nombre de molécules et en déduire l'équation de la diffusion dans la couche de produit poreux :

$$\epsilon \frac{\partial n}{\partial t} = D_e \frac{\partial^2 n}{\partial x^2} \quad (1)$$

où D_e est un coefficient de diffusion effectif qui prend en compte le caractère poreux du matériau.

On utilise plus généralement l'équation de la diffusion de A dans la couche de produit poreux en considérant la concentration molaire C plutôt que le nombre de molécules par unité de volume :

$$\epsilon \frac{\partial C}{\partial t} = D_e \frac{\partial^2 C}{\partial x^2} \quad (2)$$

Cette équation s'accompagne des deux conditions aux limites suivantes :

- en $x = 0$, $D_e \frac{\partial C}{\partial x} \Big|_{x=0} = k_D(C_e - C_s)$ (3)

- en $x = x_f$, $D_e \frac{\partial C}{\partial x} \Big|_{x=x_f} = k_C C_f$ (4)

avec :

- k_D la conductance de transfert du fluide dans la couche limite (en $\text{m} \cdot \text{s}^{-1}$) ;
- C_e la concentration molaire du fluide dans la phase gazeuse en dehors de la couche limite du fluide ;
- C_s la concentration en surface de la particule, en $x = 0$ (donc $C_s = C(x = 0)$) ;
- C_f la concentration au niveau du front de réaction, en $x = x_f$ (donc $C_f = C(x = x_f)$) ;
- k_C la constante cinétique de la réaction de consommation du solide B par le fluide A (en $\text{m} \cdot \text{s}^{-1}$) (**figure 4**).

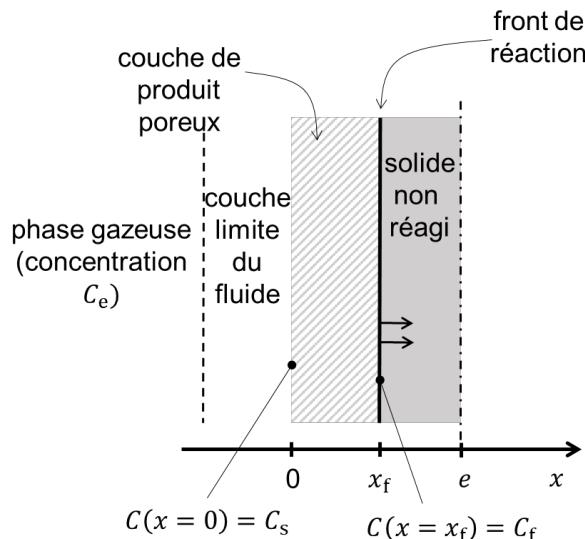


Figure 4 - Schéma de la particule de solide (sur l'intervalle $[0,e]$) partiellement consommée avec la représentation de la couche limite du fluide et du front de réaction

- Q6.** Vérifier à l'aide d'une analyse dimensionnelle que les conditions aux limites (**équations (3)** et **(4)**) sont bien homogènes.

Expliquer comment les deux conditions aux limites ont été obtenues (on pourra faire une analogie avec la loi des transferts thermiques de Newton : $\vec{J}_{\text{th}} = h(T_s - T_f)\vec{n}$ avec \vec{J}_{th} la densité de courant thermique reçu par un fluide en mouvement à la température T_f au voisinage d'un solide de température de surface T_s et h le coefficient de transfert thermique).

I.3 - Modèle simplifié : hypothèse du régime stationnaire pour la phase fluide

Pour simplifier la résolution de l'équation aux dérivées partielles (**équation (2)**), nous supposons que la phase fluide évolue en régime stationnaire. La validité de cette hypothèse sera développée dans la partie numérique.

- Q7.** Simplifier l'**équation (2)** de la diffusion de A dans le cas du régime stationnaire pour la phase fluide.

Montrer que $F_A = -D_e \frac{\partial C}{\partial x} S$, où F_A est le flux molaire de A, c'est-à-dire le flux du vecteur densité de flux de particules molaire à travers une section S de particule solide.

Indiquer comment varie F_A dans la couche de produit poreux.

- Q8.** On rappelle que $C_s = C(x=0)$ et on note C_x la concentration du fluide A pour une abscisse x comprise dans l'intervalle $[0, x_f]$.

Démontrer l'expression suivante de F_A en s'appuyant sur la **question Q7** et en précisant les différentes étapes de la démonstration :

$$F_A = \frac{D_e S}{x} (C_s - C_x) \quad (5)$$

L'hypothèse du régime stationnaire pour la phase fluide entraîne l'égalité des trois flux molaires de A : celui qui traverse la couche limite externe, celui qui diffuse dans la couche de produit poreux et celui consommé par la réaction.

L'égalité des flux molaires de A conduit à l'expression suivante :

$$F_A = k_D S(C_e - C_s) = \frac{D_e S}{x} (C_s - C_x) = k S C_x \quad (6)$$

dans le cas particulier où x est la position du front de réaction.

Cette situation est formellement analogue à un montage électrique constitué de 3 résistances placées en série traversées par un même courant. La **figure 5** représente cette analogie.

Dans notre cas, ces trois résistances en série sont dans l'ordre :

- la résistance au transfert externe (notée R_{te}) dans la couche limite externe ;
- la résistance au transfert interne (notée R_{ti}) par la diffusion dans la porosité ;
- la résistance par la réaction chimique (notée R_{rc}) localisée au niveau du front de réaction.

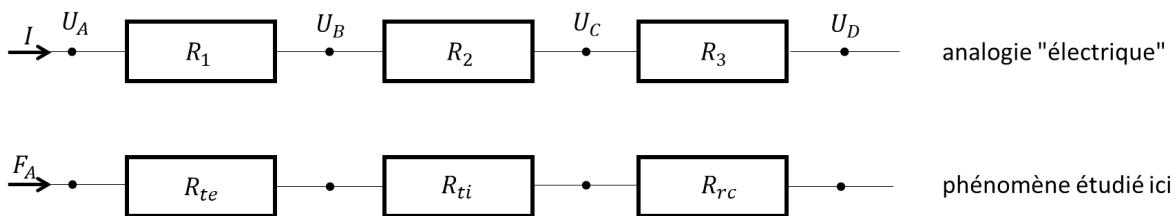


Figure 5 - Représentation schématique du problème en termes de résistances en série

- Q9.** Indiquer quel type de grandeurs du problème étudié correspondent aux potentiels U_A , U_B , U_C et U_D de la **figure 5**.

Transformer l'**équation (6)** pour faire apparaître ces résistances en se basant sur l'analogie "électrique".

Montrer que l'on peut écrire $F_A = \frac{C_e}{R_{\text{éq}}}$, où $R_{\text{éq}}$ est une résistance équivalente dont l'expression est à préciser en fonction des paramètres D_e , k , k_D , x et S .

On note N_B la quantité de matière du solide B. L'évolution de N_B en fonction du temps est régie par l'équation différentielle suivante :

$$F_A = -\frac{1}{v} \frac{dN_B}{dt} \quad (7)$$

où v est le coefficient stœchiométrique défini dans la présentation.

- Q10.** Expliquer comment a été obtenue l'équation différentielle précédente (**équation (7)**).
Indiquer comment N_B évolue en fonction du temps.

- Q11.** Soit $N_B(x)$ la quantité de matière de B comprise entre les abscisses x et e .

Pour tout x compris entre x_f et e , exprimer $N_B(x)$ en fonction de ρ_B , M_B , S , x et de e , avec ρ_B la masse volumique de B et M_B sa masse molaire.

Mettre l'**équation (7)** sous la forme $F_A = Q \frac{\partial x}{\partial t}$ avec Q une constante qui dépend de v , ρ_B , M_B et de S et dont l'expression est à préciser.

La combinaison de l'expression de F_A obtenue à la **Q9** et de celle obtenue à la question précédente permet d'obtenir l'équation à variables séparées suivante :

$$dt = K \left(\frac{1}{k_D} + \frac{x}{D_e} + \frac{1}{k} \right) dx, \text{ avec } K = \frac{\rho_B}{v C_e M_B}. \quad (8)$$

Q12. Montrer que l'intégration de l'**équation (8)** entre la date $t = 0$ (pour laquelle le front de réaction est en $x = 0$) et la date courante $t = t_f$ (front de réaction en x_f) permet d'obtenir la relation :

$$t_f = t_{0e} \frac{x_f}{e} + t_{0i} \left(\frac{x_f}{e} \right)^2 + t_{0c} \frac{x_f}{e} \quad (9)$$

avec :

- $t_{0e} = K \frac{e}{k_D}$, le temps caractéristique de transport dans la couche limite externe ;
- $t_{0i} = K \frac{1}{D_e} \frac{e^2}{2}$, le temps caractéristique de transport interne par diffusion dans la porosité ;
- $t_{0c} = K \frac{e}{k_C}$, le temps caractéristique de consommation par la réaction chimique.

On introduit le taux de conversion de B, X_B , défini par le rapport entre la quantité de B consommée et la quantité initiale de B, notée N_{B0} .

Q13. Rappeler la relation entre X_B , N_{B0} et N_B , la quantité de B restante.

Q14. Donner la relation entre X_B et x_f , puis exprimer t_f en fonction des trois temps caractéristiques et de X_B .

Q15. Donner la relation entre le temps total de consommation des particules, t_0 , et les trois temps caractéristiques.

Q16. Une expérience en laboratoire sur des particules d'épaisseur $2e$ a conduit aux valeurs suivantes : $t_{0e} = 60$ s, $t_{0i} = 300$ s et $t_{0c} = 120$ s.

Calculer le facteur par lequel est multiplié le temps total de consommation des particules si l'épaisseur des particules est multipliée par un facteur 2.

L'analyse des valeurs des trois temps caractéristiques permet de mettre en évidence le poids de chaque phénomène dans la résistance au transfert de matière. Dans certains cas extrêmes, l'un de ces trois phénomènes peut prendre le dessus sur les autres. On va alors parler de régime limitant. Dans le cas de cette étude, le régime limitant peut donc être le transport externe, la diffusion dans le produit poreux ou la réaction chimique. D'un point de vue pratique, on considère qu'un régime est limitant lorsque le temps caractéristique associé est supérieur d'un ordre de grandeur aux temps caractéristiques associés aux deux autres phénomènes (par exemple, si le transport de matière externe était le phénomène limitant, on aurait $t_{0e} \gg t_{0i}$ et $t_{0e} \gg t_{0c}$).

Q17. Donner les conditions sur k_D , k_C et D_e dans le cas où le régime limitant est le régime de diffusion interne.

Partie II - Résolution numérique du problème

Note : les codes numériques demandés au candidat devront être réalisés dans le langage Python. On supposera la bibliothèque « numpy » chargée. Une **annexe** présentant les fonctions usuelles de la bibliothèque numpy de Python est disponible page 16. Les commentaires suffisants à la compréhension du programme devront être apportés et des noms de variables explicites devront être utilisés lorsque ceux-ci ne sont pas imposés.

On souhaite vérifier que l'hypothèse du régime stationnaire pour le fluide A est valable et on propose de résoudre numériquement l'équation de la diffusion de A (**équation 2**). On s'intéresse au cas particulier où l'un des trois phénomènes est limitant (il sera à déterminer au préalable dans une des questions qui suit). L'épaisseur totale de la plaque ($2e$) est de 2,0 mm. Le fluide est de l'air synthétique constitué de dioxygène (fraction molaire de 0,21) et de diazote (fraction molaire de 0,79). Le solide réagit uniquement avec le dioxygène. La réaction est menée à une pression de 2,5 bars et à une température de 900 °C. On suppose que la concentration de fluide ne varie pas au cours de l'avancement de la réaction (large excès). La masse molaire du solide est égale à 97,5 g·mol⁻¹ et sa masse volumique est de 4 130,0 kg·m⁻³. La porosité du produit solide formé est de 0,5. Le coefficient de diffusion du fluide dans le produit poreux, D_e , est égal à 1,25·10⁻⁶ m²·s⁻¹. La conductance de transfert et la constante cinétique de la réaction sont toutes les deux égales à 100 m·s⁻¹. Le coefficient stœchiométrique v vaut 1.

Le **tableau 1** contient une partie des grandeurs mathématiques et des objets Python utilisés par la suite. Certaines grandeurs, moins importantes ou faisant l'objet d'une question spécifique, ne figurent volontairement pas dans ce tableau.

Symbol mathématique	Objet Python	Signification
$C(x, t)$		concentration du fluide A
	<code>vect_C</code>	vecteur des concentrations
	<code>vect_Cprec</code>	copie du vecteur des concentrations
	<code>vect_R</code>	vecteur des indices du front de réaction
x		abscisse
	<code>vect_x</code>	vecteur des abscisses
e	<code>epaisseur</code>	moitié de l'épaisseur d'une particule
N	<code>N</code>	nombre d'intervalles
Δx	<code>Dx</code>	pas d'espace
i	<code>i</code>	indice spatial
i_{fr}	<code>ifr</code>	indice du front de réaction
t		date
	<code>vect_t</code>	vecteur des dates
k	<code>k</code>	indice temporel
Δt	<code>Dt</code>	pas de temps
$ItMax$	<code>ItMax</code>	nombre maximum d'itérations

Tableau 1 - Tableau de correspondance entre grandeurs mathématiques et objets Python

On suppose que le transport interne du fluide (par diffusion dans les pores du solide) est le régime limitant.

Q18. Calculer la concentration du fluide A dans la phase gazeuse, C_e (mol·m⁻³), à l'aide des données de l'énoncé sachant que le solide réagit uniquement avec le dioxygène contenu dans le flux gazeux. On pourra supposer que le fluide A se comporte comme un gaz parfait à la pression de l'étude. Pour la constante des gaz parfaits on prendra $R = 8,314 \text{ J} \cdot \text{mol}^{-1} \cdot \text{K}^{-1}$.

Q19. Calculer le temps total de consommation des particules dans le cas où l'on ferait l'hypothèse du régime stationnaire pour le fluide à l'aide de l'**équation (9)**.

Q20. Ecrire une fonction **Cgaz(x, P, T)** qui prend en arguments la fraction molaire x , la pression totale P et la température T d'un gaz supposé parfait et qui renvoie sa concentration molaire C (en utilisant les unités SI).

La méthode utilisée pour la résolution numérique de l'équation aux dérivées partielles représentant la diffusion dans le produit poreux est la méthode des différences finies avec un schéma explicite. La première étape est la transformation de l'**équation (2)** pour la discréteriser dans l'espace et dans le temps.

Pour discréteriser l'espace, on choisit de découper l'intervalle $[0, e]$ en 30 parties de même longueur (**figure 6**). On utilise l'indice i pour identifier le $i^{\text{ème}}$ point de l'intervalle (i varie de 0 à N).

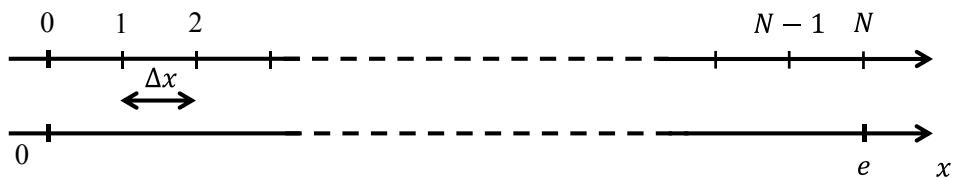


Figure 6 - Discréétisation spatiale dans la direction x

Q21. Préciser la valeur de N .

Donner le code permettant de calculer le pas d'espace Δx (variable Python : **Dx**).

Donner le code permettant de construire le vecteur **vect_x** qui contiendra les abscisses x .

Pour obtenir l'évolution de la concentration en chaque point de l'espace en fonction du temps, on discréteise le temps en intervalles de durée Δt , appelé le pas de temps (variable Python : **Dt**). On utilise l'indice k pour identifier la date particulière t_k tel que $t_k = k \Delta t$. L'indice k commence à 0 et peut prendre $ItMax$ valeurs (variable Python : **ItMax**). $ItMax$ est fixée par l'utilisateur et représente le nombre maximum d'itérations.

La discréétisation de l'équation aux dérivées partielles est obtenue en écrivant des développements limités. Dans la suite, on note $C_{i,k}$ la concentration au point d'abscisse x_i à l'instant t_k . Ainsi :

- $C_{i,k+1}$ représente la concentration au point d'abscisse x_i à la date $t_{k+1} = t_k + \Delta t$;
- $C_{i+1,k}$ représente la concentration au point d'abscisse $x_{i+1} = x_i + \Delta x$ à la date t_k .

Q22. À l'aide d'un développement limité, donner l'expression de $C(x, t + \Delta t)$ à l'ordre 1 en fonction

de $C(x, t)$ et de $\frac{\partial C}{\partial t}\Big|_{x,t}$, la dérivée de C par rapport au temps évaluée en x à l'instant t .

Q23. Donner alors l'expression de $\frac{\partial C}{\partial t}\Big|_{x_i,t_k}$, la dérivée partielle de C par rapport au temps évaluée à l'abscisse x_i à l'instant t_k , en fonction de $C_{i,k}$, $C_{i,k+1}$ et Δt .

L'utilisation de deux développements limités a conduit à l'expression discréétisée (admise) suivante pour l'approximation de la dérivée seconde de C par rapport à x :

$$\frac{\partial^2 C}{\partial x^2}\Big|_{x_i,t_k} \approx \frac{C_{i+1,k} - 2C_{i,k} + C_{i-1,k}}{(\Delta x)^2}. \quad (10)$$

Q24. Préciser l'ordre auquel ont été faits les développements limités qui ont permis d'obtenir l'expression approchée donnée par l'**équation (10)**.

Q25. Montrer que l'équation aux dérivées partielles de diffusion (**équation (2)**) peut se mettre sous la forme suivante :

$$C_{i,k+1} = C_{i,k} + r (C_{i+1,k} - 2C_{i,k} + C_{i-1,k}) \quad (11)$$

et donner l'expression de la variable r en fonction du coefficient de diffusion D_e , de la porosité ε , du pas de temps Δt et du pas d'espace Δx .

Pour des raisons de convergence numérique, on admet que la quantité $\frac{D_e}{\varepsilon} \frac{\Delta t}{\Delta x^2}$ doit être inférieure à $1/2$.

Q26. Donner la gamme de valeurs de Δt satisfaisant à la condition ci-dessus en admettant que $\Delta x = 3,33 \cdot 10^{-5}$ m.

Par la suite, on choisit un pas de temps suffisamment petit, $\Delta t = 1,00 \cdot 10^{-4}$ s, pour satisfaire à la condition $\frac{D_e}{\varepsilon} \frac{\Delta t}{\Delta x^2} < 1/2$.

Q27. Avec ce pas de temps et en supposant une durée totale de consommation $t_0 \approx 3,15 \cdot 10^3$ s, donner une estimation du nombre d'itérations nécessaires pour réaliser la simulation.

Au fur et à mesure de l'avancement de la réaction, le front de réaction se déplace de $x = 0$ à $x = e$. Lors du déplacement du front de réaction, la particule est constituée :

- pour $0 \leq x < x_f$, d'une partie qui a réagi dans laquelle diffuse le réactif ;
- pour $x_f \leq x < e$, d'une partie qui n'a pas encore réagi et dans laquelle la concentration en réactif fluide est nulle.

L'indice spatial correspondant à la position du front de réaction sera noté i_{fr} (variable Python : `i_fr`).

L'**équation (11)** est valable uniquement pour les indices i de 1 à $i_{fr} - 1$.

Pour les indices $i = 0$ et $i = i_{fr}$, il est nécessaire d'utiliser les conditions aux limites (**équations (3)** et **(4)**) pour calculer les valeurs particulières de $C_{0,k}$ et $C_{i_{fr},k}$. La discréttisation des conditions aux limites conduit aux deux relations de récurrence suivantes :

$$\bullet \text{ en } x = 0, C_{0,k} = \frac{C_e + sC_{1,k}}{1+s} \quad (12)$$

$$\bullet \text{ en } x = x_f, C_{i_{fr},k} = \frac{u}{1+u} C_{i_{fr}-1,k} \quad (13)$$

Q28. Donner les expressions des paramètres s et u qui interviennent dans les relations de récurrence permettant de calculer les concentrations aux limites.

Pour résoudre numériquement l'**équation (2)**, on utilise l'algorithme décrit par le **code Python 1** dont les étapes principales sont les suivantes :

- initialisation des variables ;
- boucle d'intégration :
 - calcul de l'indice i_{fr} correspondant au front de réaction ;
 - calcul des concentrations à chaque pas d'espace à l'aide des relations de récurrence ;
 - stockage des concentrations dans une matrice.

Certaines portions du **code Python 1** sont volontairement absentes car elles seront à compléter. Elles sont signalées par " ____ ".

Q29. Compléter le **code Python 1**. La variable utilisée pour définir le compteur sera appelée « **compteur** ». Les seules variables à considérer pour l'initialisation sont celles relatives au compteur de la boucle.

```

1  #Boucle itérative
2  #Initialisation des variables
3  ---                               #valeur ItMax
4  ---                               #valeur initiale pour la variable compteur
5
6  #Début de la boucle
7  --- compteur < --- :
8
9   #Calcul de l'indice ifr (code non demandé ici)
10
11  #Calcul des concentrations à chaque pas d'espace (code non demandé ici)
12
13  #stockage des concentrations dans une matrice (code non demandé ici)
14
15  #Incrémantation du compteur
16  ---
17

```

Code Python 1 - Structure de la portion du code correspondant à la boucle itérative

Le **code Python 2** permet de calculer les concentrations en tout point de l'espace à la date t_{k+1} à partir des concentrations en tout point de l'espace à la date t_k . Les concentrations seront stockées dans les vecteurs nommés **vect_C** et **vect_Cprec** qui ont la même dimension que le vecteur qui contient les abscisses (**vect_x**). On suppose que les valeurs des paramètres **r**, **s** et **u** ont déjà été définies ainsi que celles du vecteur **vect_x**. Initialement les concentrations sont nulles en tout point de l'espace ($C_{i,0} = 0$ pour tout i).

```

1  #Initialisation de vect_C et de vect_Cprec
2  vect_C = np.zeros(len(vect_x))
3  vect_Cprec = np.zeros(len(vect_x))
4
5  #Calcul des concentrations
6  #Pour i = 0
7  vect_C[0] = (Ce + s * vect_C[_]) / (1 + s)
8
9  #Pour i de 1 à ifr - 1
10 for i in range (_,_):
11     vect_C[i] = r * vect_Cprec[_] + (1-2*r)*vect_Cprec[_] + r * vect_Cprec[_]
12
13 #Pour i = ifr
14 vect_C[ifr] = u / (1 + u) * vect_C[_]
15
16 #Stockage des valeurs de C à l'instant k pour le calcul à l'instant k+1
17  ---
18

```

Code Python 2 - Code de calcul de la concentration du fluide

Q30. Préciser les indices des vecteurs manquant dans le **code Python 2** aux lignes 7, 11 et 14 (ils sont indiqués comme suit : **[_]**).

Indiquer les indices de début et de fin de la boucle for à la ligne 10.

Expliquer le rôle du vecteur **vect_Cprec** et compléter le code de la ligne 17.

L'indice du front de réaction, i_{fr} , correspond à la partie entière du rapport entre l'épaisseur de la couche de produits poreux, notée R , et du pas d'espace, Δx . Pour l'obtenir, il faut calculer R à chaque itération. Pour cela, on utilise l'expression discrétisée du bilan de matière en régime transitoire sur le réactif solide, qui s'écrit :

$$\frac{R_{k+1} - R_k}{\Delta t} = - \frac{D_e}{V_{\text{molB}}} \frac{C_{i_{fr}, k+1} - C_{i_{fr}-1, k+1}}{\Delta x} \quad (14)$$

avec R_k l'épaisseur de la couche de produit poreux à la date t_k (c'est aussi la position du front de réaction) et V_{molB} le volume molaire de B (variable Python : `VmolB`).

Q31. Donner le code permettant de calculer la valeur de R_{k+1} , la valeur de R à l'instant t_{k+1} . Indiquer avec précision où il sera inséré dans le **code Python 1**.

Q32. Donner le code permettant de calculer la valeur de l'indice du front de réaction i_{fr} à partir de la valeur de R_k .

Le nombre d'itérations étant très grand, on ne souhaite pas enregistrer les valeurs de concentrations à toutes les dates, mais seulement toutes les 100 000 itérations. Pour ces itérations, on enregistre :

- la valeur des concentrations dans la matrice `mat_C` ;
- la date dans le vecteur `vect_t` ;
- l'abscisse du front de réaction dans le vecteur `vect_R`.

Q33. Justifier les dimensions de la matrice `mat_C`, du vecteur `vect_t` et du vecteur `vect_R`.

Q34. Compléter le **code Python 3** correspondant à l'enregistrement des concentrations et de la date aux instants souhaités (les portions de code à compléter sont identifiées par " ____ ").

```

1  #Boucle itérative
2  #Initialisation des variables
3  mat_C = ____                      #matrice des concentrations
4  vect_t = ____                      #vecteur des temps
5
6  #Début de la boucle
7
8
9      #Calcul de l'indice ifr (code non demandé ici)
10
11     #Calcul des concentrations à chaque pas d'espace (code non demandé ici)
12
13     #stockage des concentrations dans la matrice mat_C et des dates dans vect_t
14     if j * 100000 ____ :
15         mat_C[____, ____] = vect_C
16         vect_t [j] = j * ____ * Dt
17         ____ - - -
18
19     #Incrémantation du compteur (code non demandé ici)
20
21

```

Code Python 3 - Stockage des concentrations calculées et de la date.

Partie III - Comparaison entre les deux modèles

Les taux de conversion du solide (défini à la **question (13)**) obtenus par les deux méthodes de calcul sont représentés en fonction du temps sur le graphe de la **figure 7**. La résolution de l'équation aux dérivées partielles prédit une consommation totale du solide après 30 417 663 itérations.

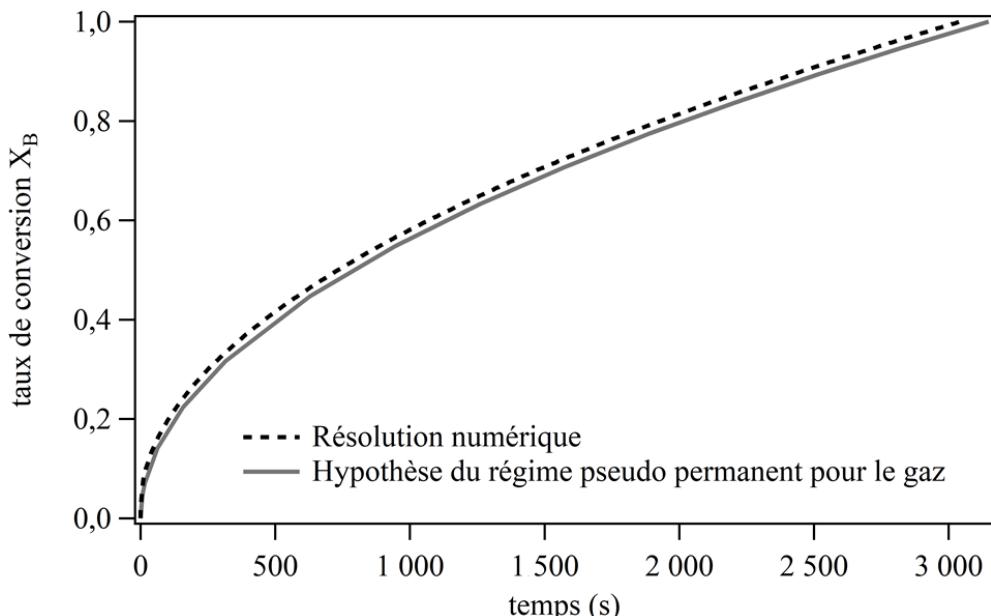


Figure 7 - Comparaison des taux de conversion obtenus par les deux modèles

Q35. Indiquer les deux erreurs qui se sont glissées dans les lignes 5 et 12 du **code Python 4** dont le but est de :

- tracer la conversion du solide en fonction du temps sur un premier graphe ;
- tracer l'évolution de la concentration de fluide réactif en fonction de l'abscisse pour le temps final d'intégration sur un second graphe.

```

1  import matplotlib.pyplot as pl
2
3  #Premier graphe
4  fig = pl.figure(1)
5  pl.plot(vect_t[:,],vect_R[:,])
6  pl.xlabel('temps')
7  pl.ylabel('conversion du solide')
8  pl.show()
9
10 #Deuxième graphe
11 fig = pl.figure(2)
12 pl.plot(vect_x,mat_C[:,ItMax])
13 pl.xlabel('abscisse')
14 pl.ylabel('concentration')
15 pl.show()
16

```

Code Python 4 - Code pour le tracé des graphes

Q36. Calculer le temps total de consommation correspondant (arrondi à la seconde près) à partir du nombre d'itérations nécessaires pour atteindre la consommation totale du solide. Comparer cette valeur avec celle obtenue en faisant l'hypothèse du régime stationnaire pour le fluide. Conclure quant à l'accord des deux modèles.

Q37. En considérant que l'hypothèse du régime stationnaire pour le gaz est acceptable et que le phénomène limitant est la diffusion du réactif fluide dans la couche de produit poreux, dessiner sur un même graphe l'allure des profils de concentration du réactif fluide entre $x = 0$ et $x = e$ pour trois configurations différentes :

- une particule qui n'a pas réagi ;
- une particule qui a totalement réagi ;
- une particule qui a partiellement réagi.

On fera figurer les valeurs de concentrations et les abscisses caractéristiques sur le graphe.

ANNEXE

Bibliothèque numpy de Python

Import de la bibliothèque **numpy** :

```
>>> import numpy as np
```

Création d'un tableau **numpy** à 2 lignes et 3 colonnes, appelé **M** :

```
>>> M = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(m)
[[1.5 2. 3.]
 [4. 5. 6.]]
```

Accès à un élément de **M** :

```
>>> M = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(M[0, 0])
1.5
>>> M[1, 2] = -7
>>> print(M)
[[1.5 2. 3.]
 [4. 5. -7.]]
```

Sélection d'une ligne ou d'une colonne de **M** :

```
>>> M = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(M[:, 1]) #Sélection de la colonne 1
[2. 5.]
>>> print(M[1, :]) #Sélection de la ligne 1
[4. 5. 6.]
```

Création d'un vecteur nul ou d'une matrice nulle :

```
>>> vec = np.zeros(5)
>>> print(vec)
[0. 0. 0. 0. 0.]
>>> mat = np.zeros((3,2))
>>> print(mat)
[[0. 0.]
 [0. 0.]
 [0. 0.]]
```

FIN



ÉPREUVE SPÉCIFIQUE - FILIÈRE PSI

MODÉLISATION ET INGÉNIERIE NUMÉRIQUE

Durée : 4 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

Le sujet est composé de trois parties indépendantes.

Modélisation de la prévention des tsunamis

Présentation générale

Les tsunamis font partie des catastrophes naturelles les plus destructrices. Provoqués par de brutales perturbations de la couche d'eau de l'océan, les tsunamis sont des phénomènes correspondant à une série de vagues océaniques pouvant atteindre de grandes amplitudes en approchant des côtes.

La majorité des tsunamis observés sont générés par des séismes (tremblements de terre) dont le foyer est situé sous l'eau et créant une perturbation verticale de la surface de l'eau. Il peut se passer plusieurs minutes, voire plusieurs heures entre le tremblement de terre et le moment où le tsunami atteint les côtes.

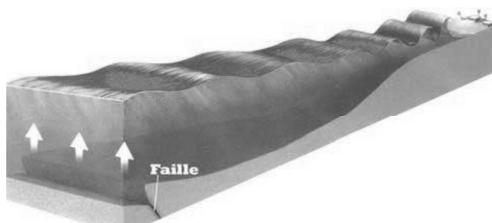


Figure 1 - Tsunami créé par une faille sismique

Des systèmes d'alerte permettent de prévenir la population dans les zones à risques. Ces systèmes d'alerte sont composés de stations munies de sismomètres et d'un réseau de bouées D.A.R.T. (Deep-ocean Assessment and Reporting of Tsunamis). Les données des sismomètres en réseau permettent de localiser le séisme et les données des bouées permettent de surveiller la propagation du tsunami en mer.

Ce sujet propose d'aborder la modélisation de trois points clés des systèmes d'alerte des tsunamis :

- La **partie I** modélise la détection d'un séisme par un sismomètre.
- La **partie II** s'intéresse à la modélisation du principe permettant de localiser un séisme à partir des données des sismomètres.
- La **partie III** propose une modélisation simple de la propagation du tsunami permettant d'estimer son temps de propagation.

Partie I - Les sismomètres

Un sismomètre est un instrument de mesure qui permet d'enregistrer les secousses sismiques, c'est-à-dire les mouvements du sol. Pour les caractériser de manière complète, une station sismique doit comporter trois sismomètres dans trois directions orthogonales formant une base de l'espace : deux horizontaux (Nord-Sud et Est-Ouest) et un vertical. La **figure 2** montre un sismomètre vertical, un sismomètre horizontal, ainsi qu'un exemple d'enregistrement obtenu au cours d'un séisme.

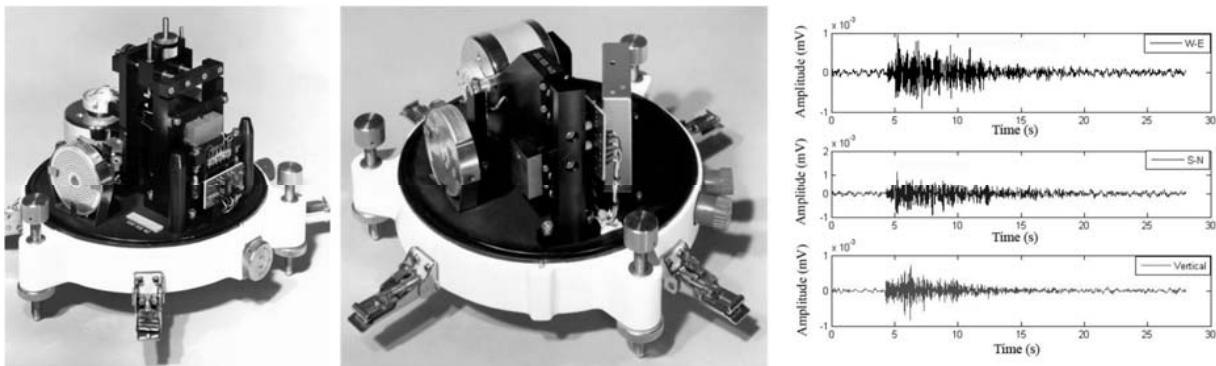


Figure 2 - Sismomètre vertical ZM 500 du CEA (à gauche), sismomètre horizontal HM 500 du CEA (au centre), sismogrammes obtenus (à droite)

Les sismomètres sont classés en fonction du contenu fréquentiel auquel ils sont sensibles. L'une des exigences de ce type de capteur est d'être capable de mesurer les mouvements du sol dans une gamme de fréquence allant de 0,1 Hz à 100 Hz (**tableau 1**) afin de couvrir une large gamme de séisme.

Exigence	Critère	Niveau
Mesurer les mouvements du sol	Gamme de fréquence	de 0,1 à 100 Hz

Tableau 1 - Exigence du sismomètre à vérifier

L'objectif de cette partie est de proposer une modélisation du fonctionnement du sismomètre et d'évaluer la pertinence de sa réponse à une sollicitation sismique en fonction de ses caractéristiques et de la fréquence des ondes sismiques.

I.1 - Principe de fonctionnement des sismomètres

L'objectif de cette sous-partie est d'appréhender le fonctionnement des sismomètres vitaux et horizontaux.

Le sismomètre vertical est constitué d'un socle très rigide encastré au sol et d'une masse M suspendue dans le vide par un ressort attaché au socle (**figure 3**). Un système d'amortissement, relié au ressort, permet d'atténuer les oscillations pour que les mesures réalisées par le système d'acquisition, lié à la masse, soient exploitables et caractérisent le séisme. Le vecteur de l'accélération de la pesanteur est noté \vec{g} . Pendant le séisme, lors du passage du train d'ondes, le mouvement relatif de la masse par rapport au socle est amorti et enregistré via un capteur électromagnétique de vitesse.

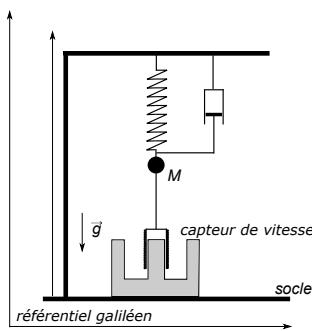


Figure 3 - Schéma d'un sismomètre vertical

- Q1.** La modélisation proposée sur la **figure 3** permet de mesurer des vibrations verticales du sol. Sur le même principe, schématiser, un modèle permettant de mesurer des vibrations horizontales.

La suite de ce sujet s'intéresse uniquement à la modélisation d'un sismomètre vertical.

I.2 - Capteur de vitesse

L'objectif de cette sous-partie est d'évaluer la bande-passante du capteur de vitesse pour que ce dernier respecte le critère d'exigence (**tableau 1**).

Le mouvement de la masse est mesuré grâce à un capteur électromagnétique de vitesse, constitué d'une bobine solidaire de la masse du sismomètre. La bobine se déplace dans le champ magnétique d'un aimant toroïdale solidaire du socle (**figure 4**). La bobine d'inductance propre L est branchée en série avec un résistor de résistance R_S aux bornes duquel on mesure une tension u . On note $\vec{v}_b = v_b(t)\vec{u}_z$ la vitesse de la bobine dans le référentiel du socle.

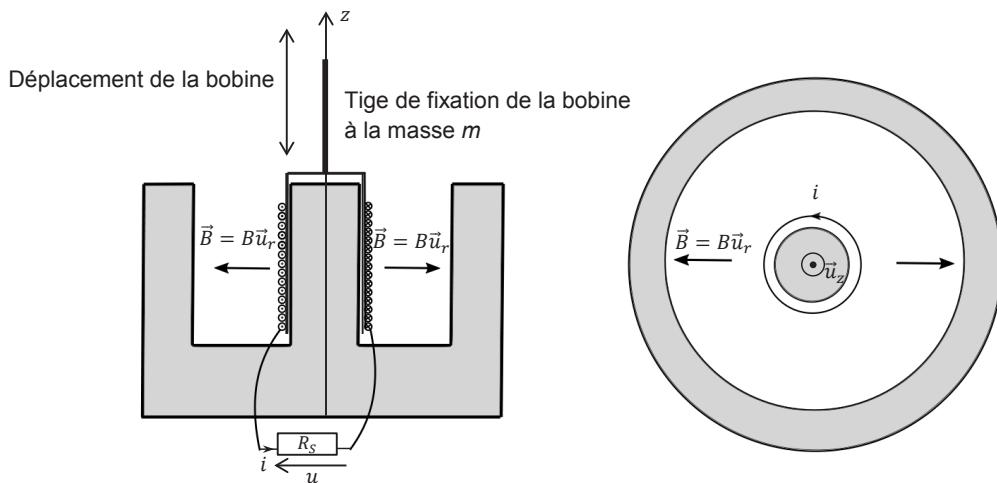


Figure 4 - Schématisation du capteur de déplacement

- Q2.** Rappeler l'expression de la force élémentaire $\delta \vec{F}_{Lap}$ dite de Laplace s'exerçant sur un tronçon conducteur de longueur dl parcourue par un courant électrique d'intensité i plongé dans un champ magnétique \vec{B} .
- Q3.** En déduire la résultante \vec{F}_{Lap} de l'action mécanique due aux forces de Laplace s'exerçant sur la bobine en fonction de B , de i et de la longueur du bobinage l .
- Q4.** Le mouvement de la bobine dans le champ magnétique induit une force-électromotrice $e(t)$ fournissant une puissance opposée à la puissance mécanique développée par \vec{F}_{Lap} . Interpréter physiquement ce résultat, puis donner l'expression de $e(t)$ en fonction de B , l et de $v_b(t)$.
- Q5.** Représenter le schéma électrique équivalent du capteur de déplacement.
- Q6.** Déterminer la relation différentielle liant $u(t)$ à $v_b(t)$.
- Q7.** Pour $L = 100 \text{ mH}$ et en tenant compte du critère d'exigence en fréquence, indiquer la valeur minimale de R_S pour laquelle la tension u peut être considérée comme proportionnelle à la vitesse v_b permettant à ce dispositif de constituer un capteur de vitesse instantanée lors d'un séisme. On admettra ce point par la suite.
- Q8.** Déduire de la question précédente que $\vec{F}_{Lap} = -\lambda_e \vec{v}_b$. Exprimer λ_e en fonction de B , l et de R_S . L'expression de \vec{F}_{Lap} est-elle en accord avec la loi de modération de Lenz ?

I.3 - Modèle de connaissance d'un sismomètre vertical

L'objectif de cette sous-partie est d'établir le modèle de connaissance d'un sismomètre vertical et d'établir le lien entre ses grandeurs caractéristiques et celles des séismes.

Le paramétrage, donné sur la **figure 5**, est adopté pour les 3 configurations suivantes :

- au repos et sans masse suspendue (**a**) ;
- au repos et avec la masse suspendue et le capteur(**b**) ;
- au cours d'un séisme (**c**).

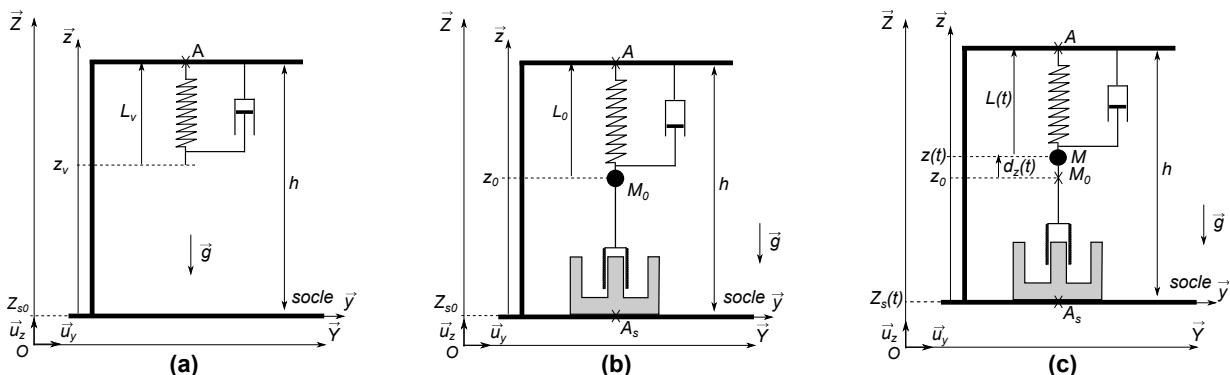


Figure 5 - Trois configurations différentes pour le paramétrage

Hypothèses et modélisation

- Les positions z_0 et $z(t)$ sont repérées par rapport au socle. Lorsque le sol est en mouvement, du fait d'un séisme, le référentiel associé au socle ne peut donc plus être considéré comme galiléen. La **figure 5** introduit cette distinction entre le référentiel \mathcal{R} ($O, \vec{X}, \vec{Y}, \vec{Z}$), supposé galiléen, et le référentiel ($A_s, \vec{x}, \vec{y}, \vec{z}$). La base orthonormée associée est définie par $(\vec{u}_x, \vec{u}_y, \vec{u}_z)$.

- Action mécanique du ressort sur la masse : $\{\mathcal{T}_{ressort \rightarrow \text{masse}}\} = \left\{ \begin{array}{c} F_r \vec{u}_z \\ \vec{0} \end{array} \right\}_A$.

Le ressort, de masse négligeable, de constante de raideur k , de longueur à vide L_v , a une longueur L_0 lorsqu'une masse ponctuelle M lui est solidaire. Au cours du séisme, cette longueur varie et est notée $L(t)$ à un instant t .

- Action mécanique de l'amortisseur sur la masse : $\{\mathcal{T}_{amortisseur \rightarrow \text{masse}}\} = \left\{ \begin{array}{c} -F_a \vec{u}_z \\ \vec{0} \end{array} \right\}_A$.

Le système d'amortissement, de coefficient de frottement visqueux λ_m , exerce une action mécanique modélisée par une force supposée linéaire en vitesse dans le référentiel lié au socle. Cette force s'oppose au mouvement de la masse.

- Action mécanique du capteur sur la masse M : $\{\mathcal{T}_{capteur \rightarrow \text{masse}}\} = \left\{ \begin{array}{c} -F_{Lap} \vec{u}_z \\ \vec{0} \end{array} \right\}_A$.

Quels que soient les résultats obtenus dans la partie précédente, on considère maintenant que le capteur de vitesse induit une force d'amortissement électrique, de coefficient λ_e , s'opposant de manière proportionnelle à la vitesse de déplacement de la masse dans le référentiel lié au socle.

Notations

$$\overrightarrow{OA_s} \vec{u}_z = Z_s(t); \overrightarrow{MA} = L(t) \vec{u}_z; \overrightarrow{M_0A} = L_0 \vec{u}_z; \overrightarrow{M_0M} = d_z(t) \vec{u}_z.$$

Q9. Isoler la masse ponctuelle M et effectuer un bilan des actions mécaniques extérieures dans le cas où le sismomètre est au repos. Écrire le théorème de la résultante statique suivant \vec{u}_z et en déduire la relation à l'équilibre entre z_0 , z_v , k , M et de g .

Q10. Au cours du séisme, déterminer l'expression de l'accélération de la masse M dans le référentiel galiléen \mathcal{R} en fonction des longueurs pertinentes.

Q11. Isoler la masse ponctuelle M et effectuer un bilan des actions mécaniques extérieures au cours d'un séisme. Établir l'équation différentielle vérifiée par $d_z(t)$ lors de la seconde sismique. Simplifier l'expression pour obtenir le résultat uniquement en fonction de $d_z(t)$, de ses dérivées, de $\frac{d^2 Z_s(t)}{dt^2}$ et des constantes du problème k , λ_e , λ_m et M .

Les transformées de Laplace des fonctions $Z_s(t)$ et $d_z(t)$ sont notées respectivement $Z_S(p)$ et $D_Z(p)$.

Q12. Déterminer, dans les conditions d'Heaviside, la fonction de transfert $H(p) = \frac{D_Z(p)}{Z_S(p)}$ et la mettre sous forme canonique. Exprimer la pulsation propre ω_0 et le coefficient d'amortissement ξ en fonction de M , λ_e , λ_m et de k .

Quels que soient les résultats obtenus précédemment, on considère la fonction de transfert :

$$H(p) = \frac{D_Z(p)}{Z_S(p)} = \frac{-\frac{p^2}{\omega_0^2}}{1 + \frac{2\xi}{\omega_0}p + \frac{p^2}{\omega_0^2}}.$$

Au cours des séismes entraînant des tsunamis, le sol est soumis à une vibration verticale dépendante du temps. Le séisme présente une excitation du sol sinusoïdale de la forme :

$$Z_s(t) = Z_s \cos(\omega t)$$

avec Z_s l'amplitude des déplacements du sol et ω la pulsation des oscillations.

Q13. Déterminer le module et l'argument de la fonction de transfert harmonique $\underline{H}(j\omega)$.

Distinguer les résultats en fonction de la valeur du ratio $\frac{\omega}{\omega_0}$.

Q14. En régime permanent, la réponse du sismomètre est une fonction de forme similaire à l'excitation induite par le séisme. Donner la forme de la réponse $d_z(t)$ en fonction des caractéristiques du sismomètre (ξ, ω_0) et des caractéristiques du séisme (Z_s, ω).

La **figure 6** donne l'évolution du gain et de la phase de $\underline{H}(j\omega)$ en fonction du rapport $\frac{\omega}{\omega_0}$ pour différentes valeurs du coefficient d'amortissement ξ .

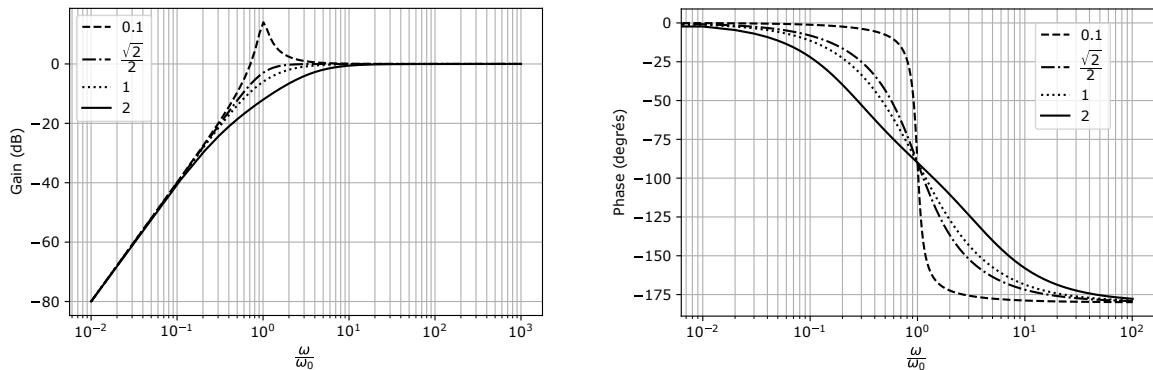


Figure 6 - Évolution du gain et de la phase de la fonction de transfert du sismomètre

Q15. Expliquer, de manière succincte, pourquoi l'évolution du gain et de la phase donnée sur la **figure 6** correspond au modèle du sismomètre proposé précédemment.

Q16. L'amplitude mesurée doit être identique à l'amplitude du séisme sur le plage de fréquence correspondant à l'exigence définie dans le **tableau 1**. Comment faut-il choisir la pulsation propre ω_0 par rapport à la pulsation ω de la secousse sismique ? Commeiner physiquement ce résultat.

Q17. Quel est le meilleur choix pour le coefficient d'amortissement ξ pour que la réponse temporelle du sismomètre à un échelon soit la plus rapide sans dépassement ? À partir des données de la **figure 6**, évaluer la valeur minimale de ω_0 correspondante en fonction de la fréquence des ondes sismiques à mesurer.

Le sismomètre étudié a une pulsation propre de $0,1 \text{ rad} \cdot \text{s}^{-1}$ et un coefficient d'amortissement de $\frac{\sqrt{2}}{2}$.

Q18. Conclure sur les fréquences de séisme que ce sismomètre est capable de mesurer. L'exigence du cahier des charges est-elle vérifiée ?

Partie II - Localisation du séisme

L'objectif de cette partie est de modéliser la localisation de l'épicentre du séisme à partir de l'estimation des distances entre chaque station et le séisme.

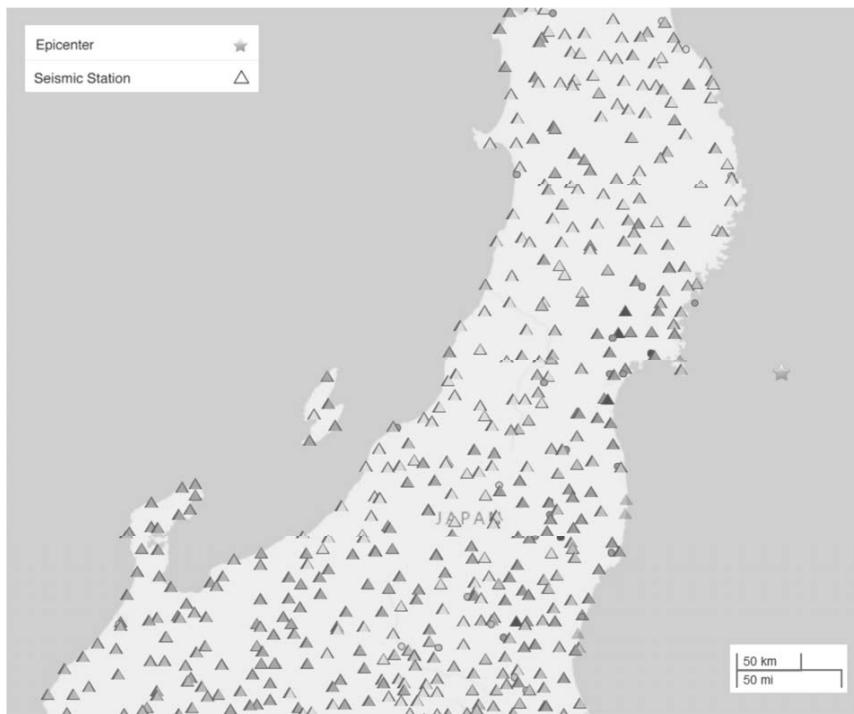


Figure 7 - Localisation de l'épicentre du séisme du 11 mars 2011 au large du Japon (Données USGS earthquake)

La localisation d'un séisme consiste à identifier l'endroit précis où il s'est produit. Les caractéristiques suivantes sont ainsi définies :

- hypocentre : point précis où a débuté le phénomène de rupture : c'est à partir de ce point que sont émis les ondes sismiques mesurées ensuite par les sismomètres. Ses coordonnées sont définies par sa latitude, sa longitude et sa profondeur.
- épicentre : projection à la surface de la Terre de l'hypocentre. Ce point est défini par ses coordonnées géographiques (latitude et longitude). Ces coordonnées sont notées x et y dans la suite du sujet.

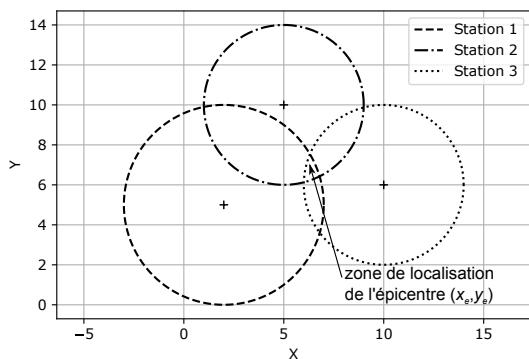
Le principe de la localisation des séismes est la trilateration des informations fournies par les enregistrements de plusieurs stations disposant de sismomètres. Les mesures sont enregistrées dans 3 directions orthogonales : 2 horizontales (Est-Ouest et Nord-Sud) et une verticale.

La **figure 7** donne la localisation de l'épicentre du séisme au large du Japon en mars 2011. Des centaines de stations ont enregistré le séisme via des sismomètres et ont pu estimer la localisation de l'épicentre du séisme.

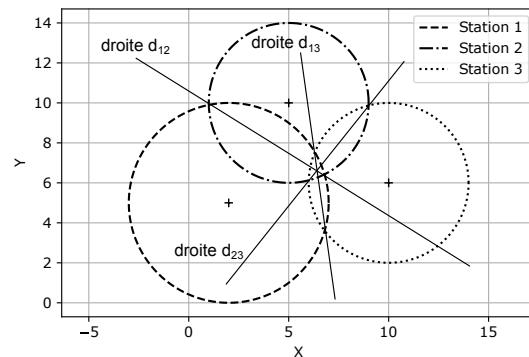
L'analyse des données des sismogrammes permet de déterminer la distance r_i entre la station i située aux coordonnées (x_i, y_i) et l'épicentre du séisme situé aux coordonnées (x_e, y_e) . Cette analyse complexe n'est pas étudiée dans ce sujet.

Pour cela, chaque station i , de coordonnées (x_i, y_i) , est positionnée sur une carte. Le rayon du cercle de centre (x_i, y_i) qui passe par l'épicentre (x_e, y_e) est noté r_i . L'évaluation des distances r_i est complexe et peut engendrer des erreurs. De ce fait, le recouplement des données issues des différentes stations de mesure ne fait que situer l'épicentre dans une certaine zone (**figure 8a**).

Pour approximer ce point (x_e, y_e) , le principe de la modélisation plane associée est donnée sur la **figure 8b** pour 3 stations de mesure.



(a) Intersection de cercles



(b) Intersection de droites

Figure 8 - Principe de localisation de l'épicentre d'un séisme

Pour obtenir un système linéaire à résoudre, les coordonnées de l'épicentre sont définies en créant des droites dont les équations sont données par la différence terme à terme des équations de deux cercles. L'intersection des droites va donner la position de l'épicentre.

Q19. Dans le cas donné sur la **figure 8a**, écrire, de manière littérale, les équations de chaque cercle.

Q20. Développer les équations des cercles des stations 1 et 2. Effectuer la différence terme à terme et en déduire l'équation de la droite d_{12} de la **figure 8b**. Effectuer l'application numérique.

Q21. Dans le cas donné sur la **figure 8b**, écrire, de manière littérale, les équations de chaque droite. En déduire le système d'équations linéaire permettant de localiser l'épicentre et le mettre sous la forme $AX = R$ avec A une matrice 3×2 , X le vecteur donnant les coordonnées de l'épicentre et R un vecteur de 3 composantes.

Q22. Evaluer le rang du système précédent. Simplifier le système matriciel précédent en le mettant sous la forme $A'X = R$ où la nouvelle matrice A' est carrée de dimension 2×2 .

L'évaluation des distances r_i peut présenter des incertitudes, ce qui va entraîner des incertitudes sur la localisation de l'épicentre. L'incertitude sur la distance est notée δR et l'incertitude sur la position de l'épicentre est notée δX . L'équation devient :

$$A'(X + \delta X) = (R + \delta R).$$

Il est possible de montrer que la variation de la localisation du séisme est majorée par la variation de la distance r_i grâce à l'expression :

$$\frac{\|\delta X\|}{\|X\|} = (\|A'\| \cdot \|A'^{-1}\|) \frac{\|\delta R\|}{\|R\|}$$

où les normes utilisées ont été bien choisies.

Le produit des normes $\|A'\| \cdot \|A'^{-1}\|$ s'appelle le conditionnement de la matrice A' . La matrice A' est bien conditionnée lorsqu'une faible variation de R engendre une faible variation de X .

Q23. Parmi les valeurs $\{-10\ 000, -1\ 000, -1, 1, 1\ 000, 10\ 000\}$, lesquelles sont plausibles et laquelle entraînera le meilleur conditionnement de la matrice A' ? Justifier.

En pratique, il n'y a pas seulement 3 stations équipées de sismomètres qui permettent de donner des informations sur la localisation de l'épicentre. En effet, la **figure 7** montre que plusieurs centaines de stations ont enregistré le séisme du 11 mars 2011 au Japon. Il est ainsi nécessaire d'analyser les données issues de toutes les stations qui ont détecté le séisme pour localiser l'épicentre du séisme le plus précisément possible.

Dans ce cas, la matrice A possède 2 colonnes et de nombreuses lignes. La résolution du système $AX = R$ se fait par le calcul de la matrice pseudo-inverse de A . Cette pseudo-inverse permet de donner une solution de norme minimale à un système d'équations où il y a plus d'équations que d'inconnues.

La pseudo-inverse se définit de la manière suivante : si tAA est inversible alors la solution de $AX = R$ de norme minimale est donnée par A^+R où A^+ est la pseudo-inverse de A définie par :

$$A^+ = ({}^tAA)^{-1} \cdot {}^tA.$$

Q24. À partir du système précédemment défini, expliquer comment obtenir la localisation de l'épicentre.

Partie III - Propagation d'un Tsunami

L'objectif de cette partie est de modéliser la propagation du tsunami et d'estimer son temps de propagation à partir de l'épicentre.

Dans cette partie, nous allons nous intéresser à la propagation du tsunami en mer afin de prévoir l'instant d'arrivée du tsunami sur les côtes. Pour cela, étudions la propagation unidimensionnelle selon l'axe (Ox) d'une onde de surface en eau peu profonde (la profondeur est très faible devant la longueur d'onde). L'eau sera considérée comme un fluide parfait et incompressible. Elle occupe au repos tout l'espace compris entre $z = 0$ et $z = h$ (**figure 9**). Lors du passage de l'onde, la surface libre de l'eau en x est déplacée verticalement de $\zeta(x, t)$. On note \vec{v} le champ des vitesses de l'écoulement, celui-ci vérifie l'équation suivante :

$$\frac{\partial \vec{v}}{\partial t} = -\frac{1}{\rho} \overrightarrow{\text{grad}}(P) + \vec{g}. \quad (1)$$

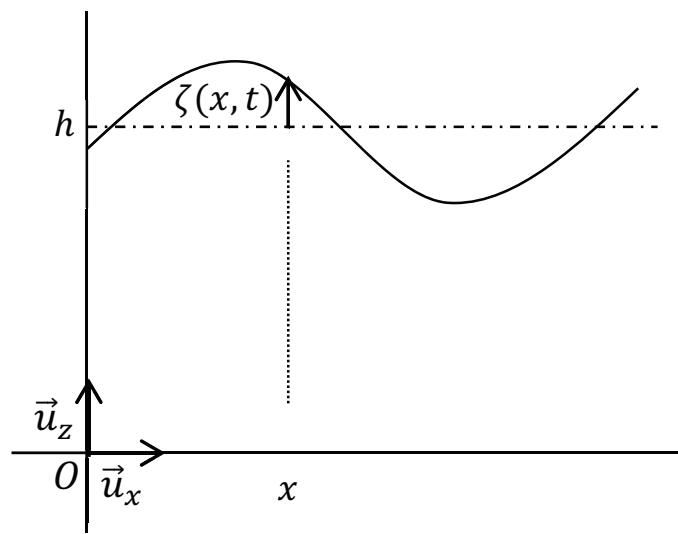


Figure 9 - Surface libre de l'eau au passage de l'onde

Données numériques et notations

- intensité du champ de pesanteur terrestre $g \simeq 10 \text{ m}\cdot\text{s}^{-2}$;
- masse volumique de l'eau $\rho = 1,0 \cdot 10^3 \text{ kg}\cdot\text{m}^{-3}$;
- longueur d'onde d'un tsunami $\lambda_T \simeq 100 \text{ km en haute mer}$;
- amplitude ζ_{max} d'un tsunami en haute mer : quelques décimètres ;
- période d'un tsunami en haute mer, $T_T \simeq 30 \text{ min}$.

Q25. Justifier, à partir des ordres de grandeur proposés, qu'il est difficile de détecter un tsunami en haute mer.

Q26. Justifier que le champ des vitesses vérifie $\vec{V} = v_x(x, z, t)\vec{u}_x + v_z(x, z, t)\vec{u}_z$.

Q27. Les trajectoires des particules de fluide au passage d'une onde sinusoïdale de longueur d'onde λ pour différentes profondeurs d'eau h sont photographiées **figure 10**. L'évolution de l'allure des trajectoires des particules de fluide pour différentes valeurs du rapport $\frac{h}{\lambda}$ est présentée **figure 11**. À la limite $\frac{h}{\lambda} \ll 1$, justifier que $\vec{V} \simeq v_x(x, t)\vec{u}_x$.

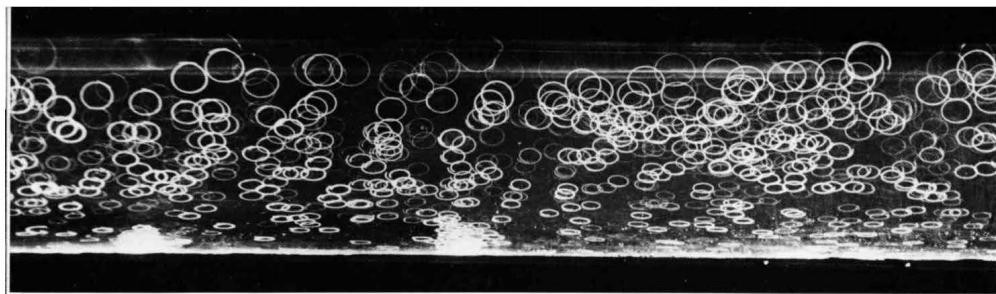


Figure 10 - Photographie des trajectoires des particules de fluide au passage d'une onde sinusoïdale

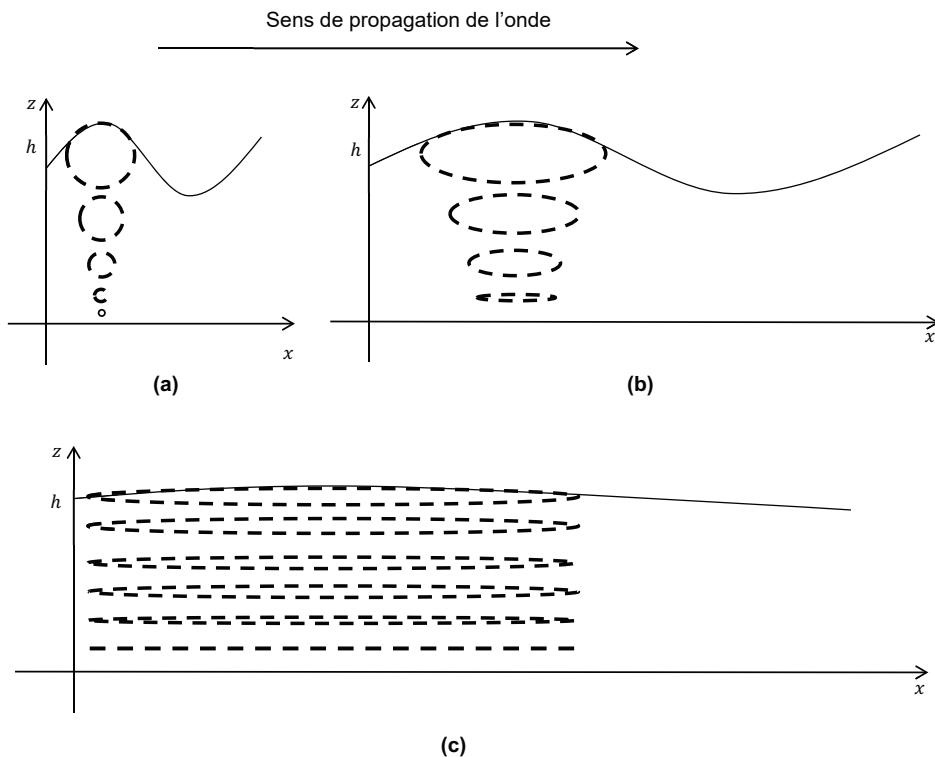


Figure 11 - Évolution de la trajectoire des particules de fluides au passage de l'onde pour des valeurs décroissantes de la figure **(a)** à la figure **(c)** du rapport $\frac{h}{\lambda}$

Q28. En généralisant l'expression du champ des vitesses obtenue à la **Q26**, simplifier l'équation (1), puis la projeter selon les axes x et z .

Q29. En déduire que le champ de pression est donné par $P(x, z, t) = P_0 + \rho g(h + \zeta(x, t) - z)$ où P_0 est la pression atmosphérique. On supposera pour cela que la pression est continue à l'interface air-eau.

Q30. En réalisant un bilan de masse, entre t et $t + dt$, pour la tranche de fluide comprise entre x et $x + dx$, de hauteur $h + \zeta(x, t)$ à l'instant t et de largeur arbitraire L_y , montrer que $\frac{\partial \zeta}{\partial t} \simeq -h \frac{\partial v_x}{\partial x}$. On mettra à profit le fait qu'une augmentation de la masse d'eau comprise entre x et $x + dx$ se traduit par une augmentation de ζ . On exprimera de deux manières la masse δm_e entrant (algébriquement) dans la tranche de fluide, entre t et $t + dt$, et on précisera l'approximation effectuée.

Q31. Finalement, montrer que :

$$\frac{\partial^2 \zeta}{\partial t^2}(x, t) - c^2 \frac{\partial^2 \zeta}{\partial x^2}(x, t) = 0$$

où $c = \sqrt{gh}$. Comment s'appelle ce type d'équation ?

Q32. La propagation des ondes vérifiant cette relation est-elle dispersive lorsque h ne varie pas ?

Sur les **figures 12 et 14**, chaque "pic" correspond à la position d'un paquet d'ondes à un instant donné. Le paquet d'ondes se déplace dans le sens des x croissants.

Q33. D'après les **figures 13 et 14**, le modèle rend-il bien compte d'une augmentation de l'amplitude au voisinage des côtes ? Le modèle rend-il bien compte d'une diminution de la célérité des ondes se propageant dans l'eau lorsque la profondeur diminue ? L'allure des courbes de la figure **figure 14** est-elle compatible avec votre réponse à la question **Q32**. Justifier.

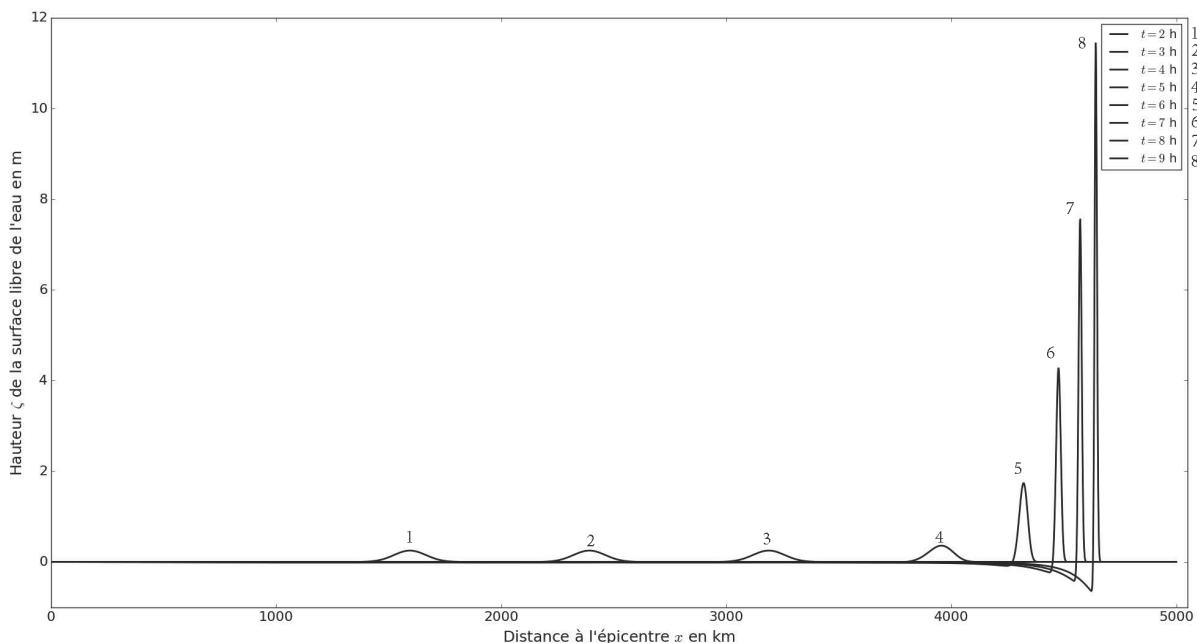


Figure 12 - Propagation "simulée" d'une perturbation "Gaussienne" à la surface de l'eau à partir de l'épicentre, la profondeur est représentée (**figure 13**). Les différentes courbes représentent la surface de l'eau à différents instants

Q34. Justifier que le temps de propagation τ_p de l'épicentre jusqu'à un point d'abscisse d est donné par $\tau_p = \int_0^d \frac{dx}{\sqrt{gh(x)}}$.

On souhaite estimer le temps de propagation du tsunami entre l'épicentre M_0 et un autre point noté M_n . Pour cela, on dispose d'une liste `dist` dont chaque élément, lui-même une liste de deux termes, est associé à un point M_i d'abscisse x_i selon l'axe x . Le premier élément de `dist[i]` est la distance M_iM_{i+1} en km et le deuxième élément est la profondeur $h(x_i)$, exprimée en mètre, de l'eau en M_i .

Q35. Écrire une fonction Python `Tpropag` prenant la liste `dist` comme argument et renvoyant, en utilisant la méthode des rectangles, le temps de propagation, exprimé en heure, du tsunami entre M_0 et M_n .

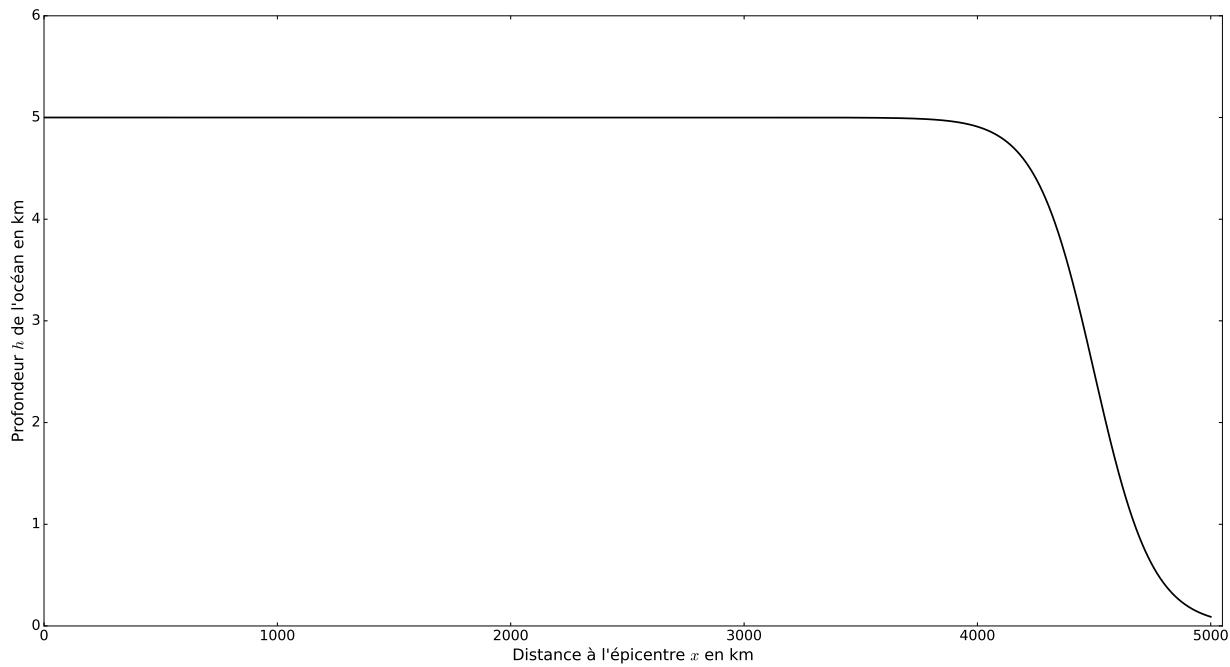


Figure 13 - Profondeur "simulée" h du plancher océanique

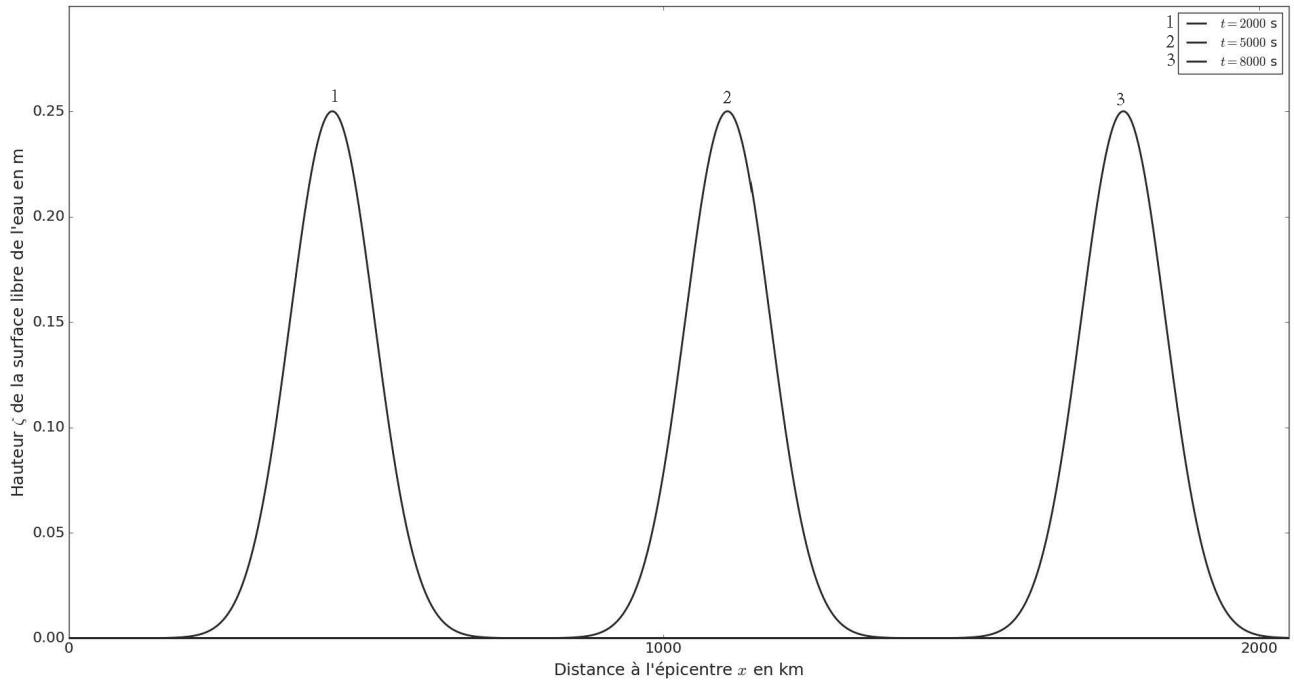


Figure 14 - Propagation "simulée" d'une perturbation "Gaussienne" à la surface de l'eau à partir de l'épicentre, la profondeur est représentée (**figure 13**). Les différentes courbes représentent la surface de l'eau à différents instants

Un réseau de détecteur appelé réseau de bouée D.A.R.T. pour Deep-ocean Assessment and Reporting of Tsunamis a été mis en place pour surveiller la propagation des tsunamis en mer. La bouée n°51407 (située au sud de Hawaï) de ce réseau a enregistré le passage du tsunami du 11 mars 2011. Cet enregistrement est présenté **figure 15**.

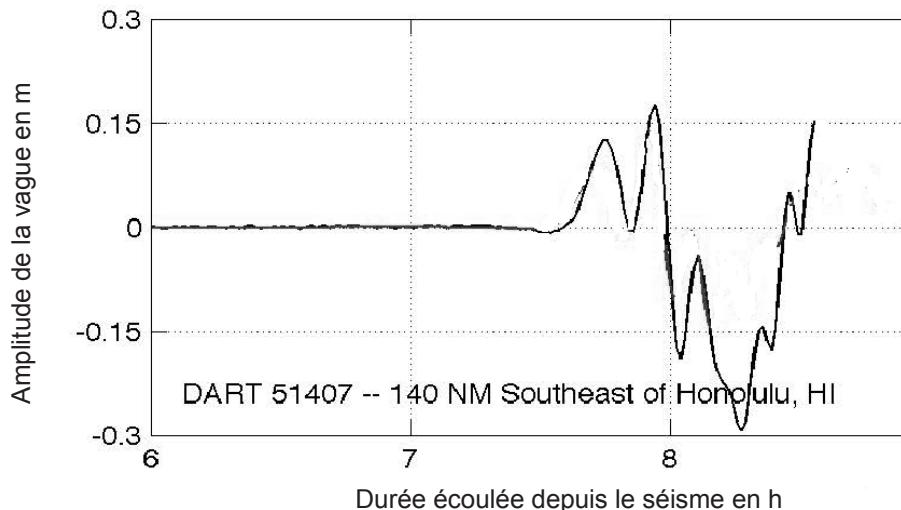


Figure 15 - Enregistrement de la bouée DART n°51407 le 11 mars 2011 (Source : NOAA center for tsunami research)

Q36. L'appel de la fonction `Tpropag` appliquée au cas de la propagation du tsunami de mars 2011 de son épicentre au large du Japon à la bouée D.A.R.T. n°51407 renvoie 7,66. En se fixant une marge égale à 1 % du temps de propagation réel, ce résultat est-il acceptable ?

Q37. La mise en place d'un réseau de sismomètres et de bouées D.A.R.T. permet-elle de prévenir les populations ?

FIN



ÉPREUVE SPÉCIFIQUE - FILIÈRE TSI

MODÉLISATION

Durée : 3 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont interdites.

Le sujet est composé de trois parties toutes indépendantes.

Le sujet comporte 12 pages.

Stabilisateur d'images Slick

Présentation générale

Un stabilisateur d'images est un support sur lequel on fixe une caméra.

Il permet d'obtenir une image parfaitement fluide des différents milieux filmés, tels que terrestre, aérien, sur l'eau et de techniques de films, plongée, contre plongée, travelling, suivi « au ras du sol », etc. (**figure 1**). Le résultat obtenu est une image très agréable à regarder ainsi qu'un suivi au plus près de l'action et des acteurs dans tous les déplacements.



Figure 1 - Photos prises à l'aide du stabilisateur Slick

Le Slick (**figure 2**) est un stabilisateur WeatherProof, il fonctionne sous la pluie ou sous la neige ; il est pratique pour le ski ou pour des activités exposées à de légères éclaboussures d'eau. Avec sa batterie au lithium-polymère de 900 mA·h, le Slick dispose d'une autonomie allant de 3 à 4 heures. Il est composé de matériaux durables : 30 % de fibres de verre et 70 % de nylon.



Figure 2 - Stabilisateur Slick



Figure 3 - Stabilisateur Slick fixé sur un guidon de vélo

Le modèle mécanique du Slick est composé de quatre classes d'équivalence (**figure 4**), chaque repère $(O_i, \vec{x}_i, \vec{y}_i, \vec{z}_i)$ est lié à la pièce $i \in \{0, 1, 2, 3\}$:

- un support **0** qui est par exemple en liaison encastrément avec le guidon du vélo (**figure 3**) ;
- un bras **1** en liaison pivot d'axe (O_0, \vec{z}_0) avec **0** ;
- un bras **2** en liaison pivot d'axe (O_1, \vec{x}_1) avec **1** ;
- un bras **3** sur lequel est fixée la caméra, **3** en liaison pivot d'axe (O_2, \vec{y}_2) avec **2**.

Chacun de ces trois degrés de liberté est piloté à l'aide d'un moteur brushless.

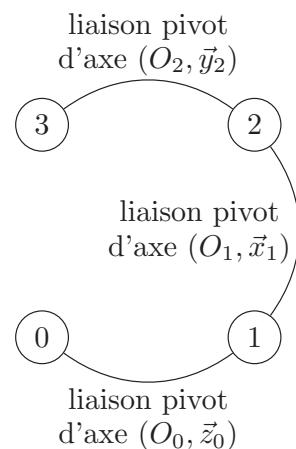


Figure 4 - Graphe des liaisons du Slick

La figure 5 présente les différents composants du Slick.

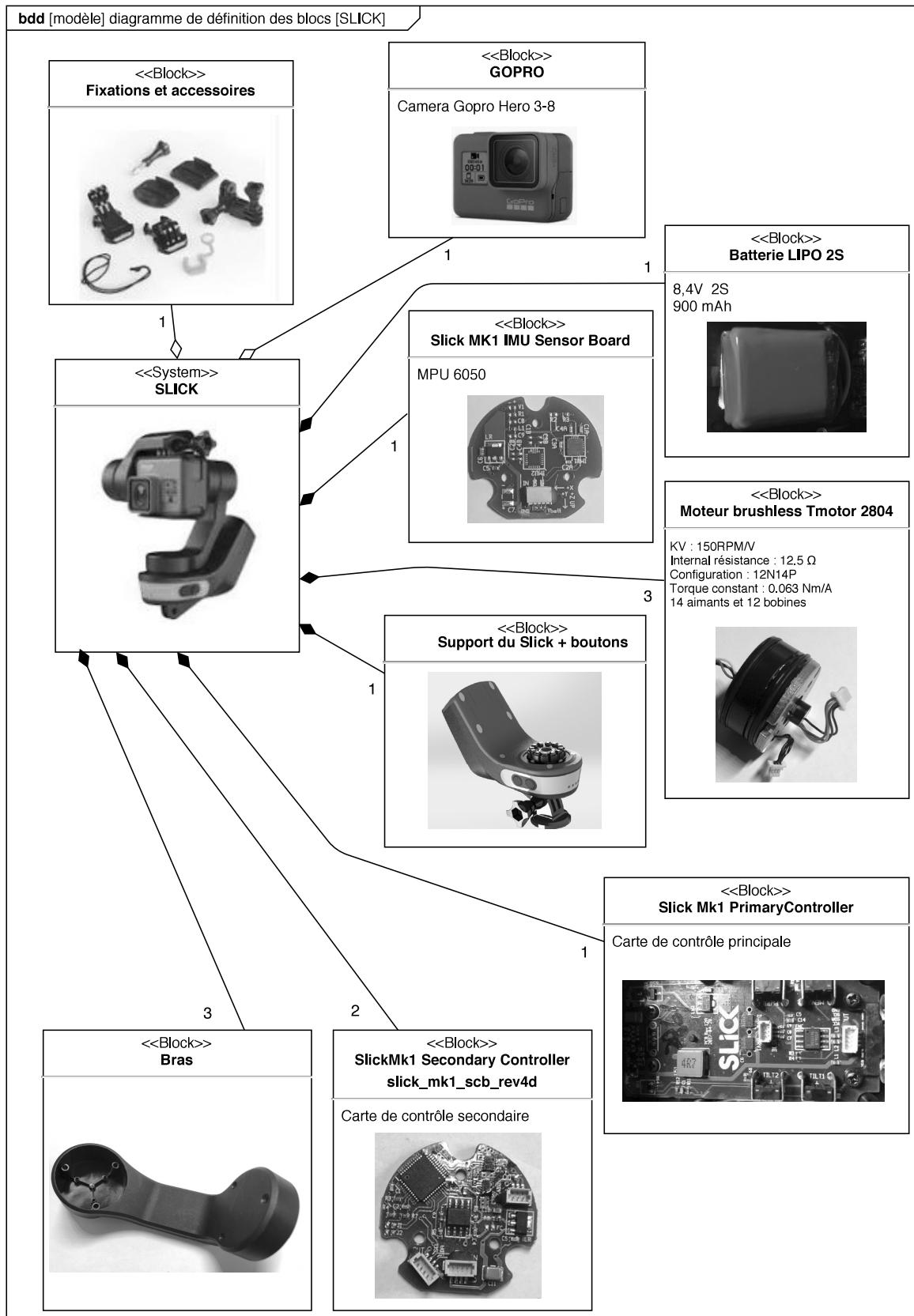


Figure 5 - Diagramme de définition des blocs (bdd) du Slick

Problématiques

1. Le capteur d'accélération de la centrale inertie sera analysé pour comprendre son fonctionnement (**partie I**).
2. Une étude mathématique de deux méthodes pour appréhender l'orientation de la caméra sera abordée (**partie II**).
3. La **partie III** permettra de choisir un nouveau matériau pour le bras **1** qui réalise le meilleur compromis raideur-masse.

Partie I - Étude du capteur d'accélération

Objectif de la partie : relier l'accélération de la caméra à la tension de sortie du capteur.

L'estimation de l'orientation (ou attitude) de la caméra est réalisée par la centrale inertie MPU 6050 fixée sur le bras **3** lié à la caméra. La centrale inertie acquiert, entre autre, des accélérations à l'aide d'un capteur MEMS (Micro-Electro-Mechanical-System) capacitif. Ce capteur est composé de deux éléments principaux (**figure 6**) :

- une partie fixe de même classe d'équivalence que la caméra ;
- une partie dite "en suspension" en liaison glissière par rapport à la partie fixe.

Ces deux éléments ont la forme de lamelles en vis-à-vis (**figure 6**).

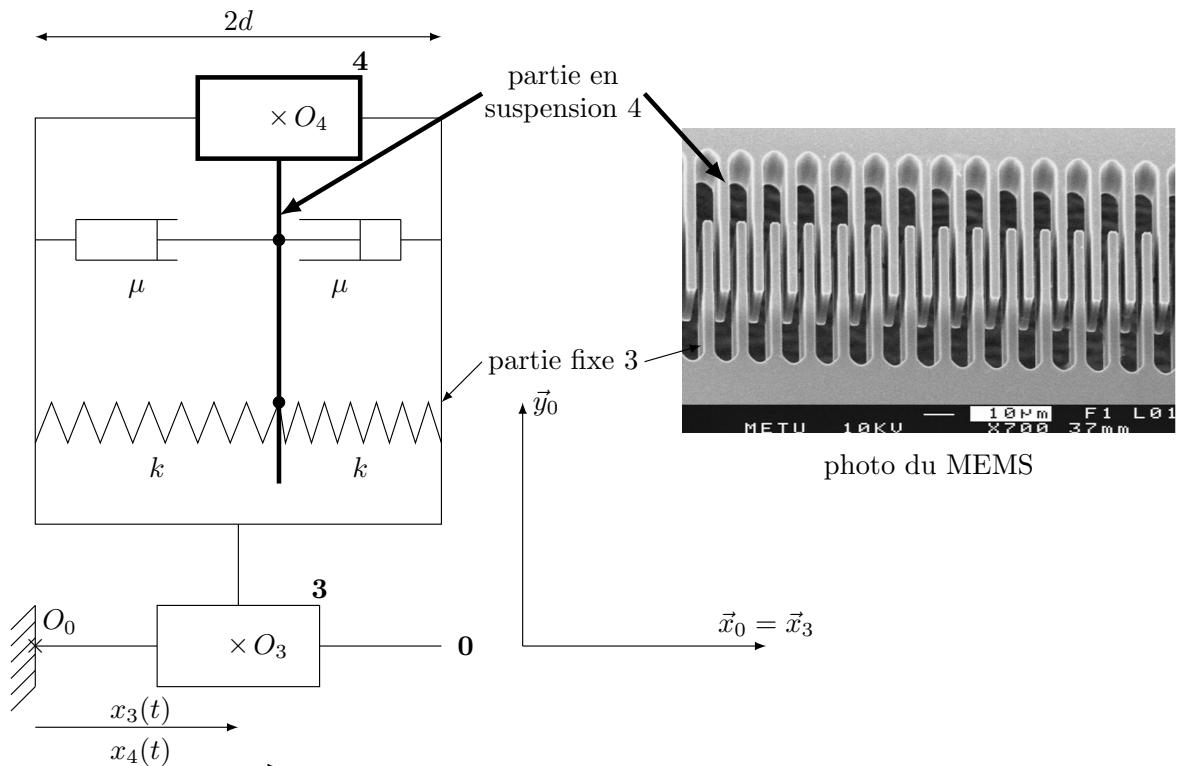


Figure 6 - Modélisation mécanique d'un capteur MEMS

Hypothèses et notations du modèle mécanique (figure 6)

- le repère $(O_0, \vec{x}_0, \vec{y}_0, \vec{z}_0)$ est lié au sol et associé à un référentiel galiléen ;

- le point O_3 est lié au bras 3, à la caméra et à la partie fixe du capteur ;
- le point O_4 est lié à la partie "en suspension" du capteur, noté 4 ;
- on étudie seulement l'accélération suivant la direction \vec{x}_0 , on supposera donc que la partie fixe du capteur (encastrée avec le bras 3 et la caméra) est en liaison glissière de direction \vec{x}_0 avec le sol ;
- on suppose qu'entre 3 et 4, il y a deux ressorts (de raideur k chacun) et deux amortisseurs (de coefficient visqueux μ chacun) ;
- $\overrightarrow{O_0O_3} \cdot \vec{x}_0 = x_3(t)$;
- $\overrightarrow{O_0O_4} \cdot \vec{x}_0 = x_4(t)$;
- $x(t) = x_4(t) - x_3(t)$;
- l'accélération de la caméra par rapport au sol est notée $a(t) = \frac{d^2x_3(t)}{dt^2}$;
- m_4 est la masse de 4 ;
- $2d$ est la distance entre deux lamelles de la même pièce ;
- l'action de pesanteur est négligée.

Q1.

- a) Appliquer le théorème de la résultante dynamique à 4 en projection sur \vec{x}_0 et montrer que :

$$\frac{d^2x(t)}{dt^2} + \frac{2\mu}{m_4} \frac{dx(t)}{dt} + \frac{2k}{m_4}x(t) = -a(t). \quad (1)$$

- b) Proposer, en justifiant votre réponse, une expression de la fréquence de variation de $a(t)$ en dessous de laquelle on peut considérer que :

$$a(t) \approx -\frac{2k}{m_4}x(t). \quad (2)$$

Hypothèses et notations du modèle électrique (figure 7)

- on suppose que les lamelles en vis-à-vis définissent deux condensateurs de capacité C_1 (à gauche sur la **figure 6**) et C_2 (à droite sur la **figure 6**) dépendantes de $x(t)$;
- deux résistors de résistance R sont utilisés pour le conditionnement ;
- l'alimentation du capteur impose : $V_s = 5 \text{ V}$, $v_1(t) = \frac{V_s}{2} + V_1 \sin(\omega t)$, $v_2(t) = \frac{V_s}{2} - V_1 \sin(\omega t)$ avec $V_1 = 2,5 \text{ V}$;
- on note $v_3(t)$ la tension de sortie du capteur.

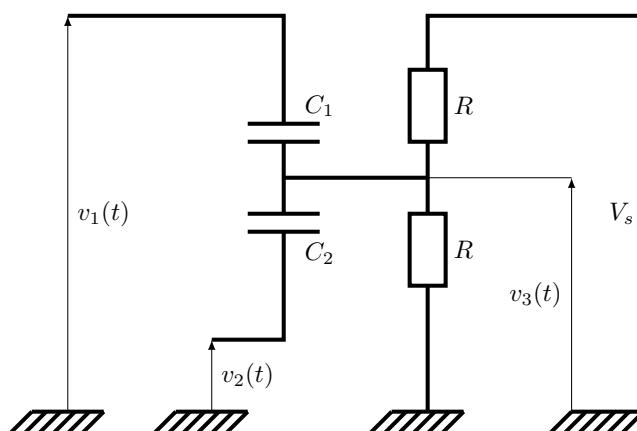


Figure 7 - Modélisation électrique d'un capteur MEMS

Q2.

- a) À partir de la loi des nœuds, donner la relation entre $v_1(t)$, $v_2(t)$, $v_3(t)$, V_s , C_1 , C_2 et R .
b) En déduire l'équation différentielle suivante :

$$\frac{dv_3(t)}{dt} + \frac{1}{\tau} v_3(t) = \frac{C_1 - C_2}{C_1 + C_2} V_1 \omega \cos(\omega t) + \frac{V_s}{2\tau}. \quad (3)$$

Donner l'expression de τ en fonction de R , C_1 et C_2 .

La solution de cette équation différentielle en régime permanent est :

$$v_3(t) = \frac{V_s}{2} + \frac{C_1 - C_2}{C_1 + C_2} V_1 \sin(\omega t). \quad (4)$$

Nous considérerons les condensateurs comme plans (**figure 8**), on rappelle que :

$$C = \frac{\epsilon S}{e}. \quad (5)$$

où :

- C est la capacité du condensateur plan ;
- ϵ est la permittivité diélectrique du milieu entre les armatures ;
- S est la surface d'une armature ;
- e est la distance entre les armatures.

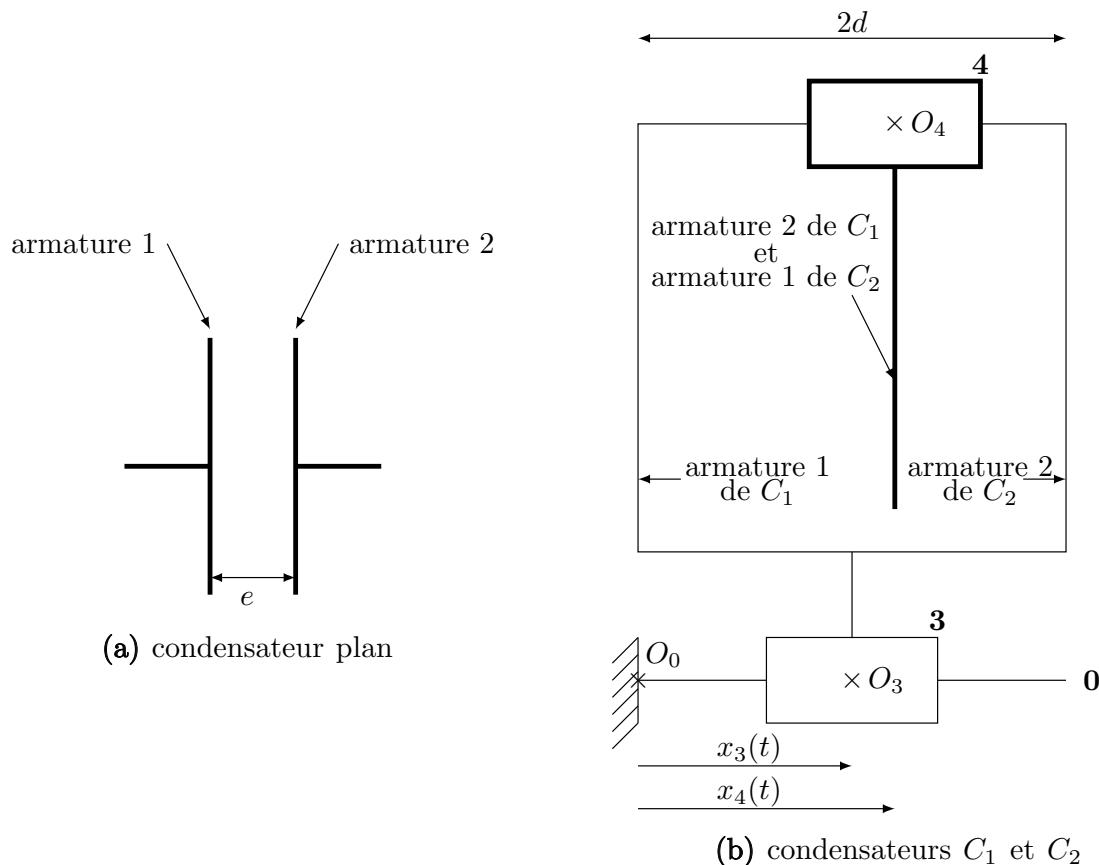


Figure 8 - Modélisation des condensateurs

Q3.

- a) Donner l'expression de C_1 , puis de C_2 en fonction de ε , S , $x(t)$ et de d . En déduire l'expression de $\frac{C_1 - C_2}{C_1 + C_2}$ en fonction de $x(t)$ et d .
- b) En déduire, en utilisant également les équations (2) et (4), l'expression de $a(t)$ en fonction de $v_3(t)$ et des données.

Partie II - Représentation mathématique de l'orientation de la caméra

Nous venons d'analyser comment la centrale inertielle mesure les accélérations.

La centrale inertielle permet également de mesurer le champ magnétique terrestre.

À partir de ces mesures, la centrale inertielle permet en outre de calculer la rotation représentant l'orientation de la caméra et par suite de la redresser.

On note $(\vec{x}_0, \vec{y}_0, \vec{z}_0)$ une base orthonormée directe fixe liée à la Terre : \vec{x}_0 est dirigé vers le nord magnétique et \vec{z}_0 vers le centre de la terre.

On note $(\vec{x}_3, \vec{y}_3, \vec{z}_3)$ une base orthonormée directe liée à la caméra (fixe par rapport à la caméra). Déterminer l'orientation de la caméra revient à identifier la rotation qui permet de passer de la base fixe $(\vec{x}_0, \vec{y}_0, \vec{z}_0)$ à la base liée à la caméra $(\vec{x}_3, \vec{y}_3, \vec{z}_3)$.

L'objectif de cette partie est d'appréhender mathématiquement deux méthodes utilisées pour la représentation et les calculs de l'orientation (ou attitude) de la caméra :

- utilisation des angles de Cardan (sous-partie II.1);
- utilisation des quaternions (sous-partie II. 2).

On se place dans \mathbb{R}^3 muni de sa structure euclidienne canonique et on notera les vecteurs (éléments de \mathbb{R}^3) sous forme de colonnes.

On note $\mathcal{B}_0 = (X_0, Y_0, Z_0)$ la base canonique : $X_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $Y_0 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ et $Z_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Le produit scalaire usuel entre deux éléments $X = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix}$ et $Y = \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix}$ de \mathbb{R}^3 sera noté $(X|Y)$.

C'est-à-dire que $(X|Y) = a_1a_2 + b_1b_2 + c_1c_2$.

Et enfin, on note X_3 , Y_3 et Z_3 les vecteurs de \mathbb{R}^3 contenant les coordonnées respectives de \vec{x}_3 , \vec{y}_3 et \vec{z}_3 dans la base $(\vec{x}_0, \vec{y}_0, \vec{z}_0)$. On notera $\mathcal{B}_3 = (X_3, Y_3, Z_3)$ la base de \mathbb{R}^3 obtenue.

Q4. Justifier, sans calcul, que R , matrice de passage de la base \mathcal{B}_0 à la base \mathcal{B}_3 , appartient à $\text{SO}_3(\mathbb{R})$.

II.1 - Les angles de Cardan

Le principe de la description d'une rotation de matrice R avec les angles de Cardan est de décomposer R en trois matrices de rotations successives :

- R_3 , matrice de la rotation d'axe orienté et dirigé par Z_0 et d'angle φ qui est la matrice de passage de la base (X_0, Y_0, Z_0) à la base notée (U, V, Z_0) ;
- R_2 , matrice de la rotation d'axe orienté et dirigé par V et d'angle θ qui est la matrice de passage de la base (U, V, Z_0) à la base notée (X_3, V, W) ;
- R_1 , matrice de la rotation d'axe orienté et dirigé par X_3 et d'angle ψ qui est la matrice de passage de la base (X_3, V, W) à la base finale (X_3, Y_3, Z_3) .

Les angles des rotations successives vérifient : $\varphi \in [-\pi/2, \pi/2]$, $\theta \in]-\pi, \pi]$ et $\psi \in]-\pi, \pi]$.

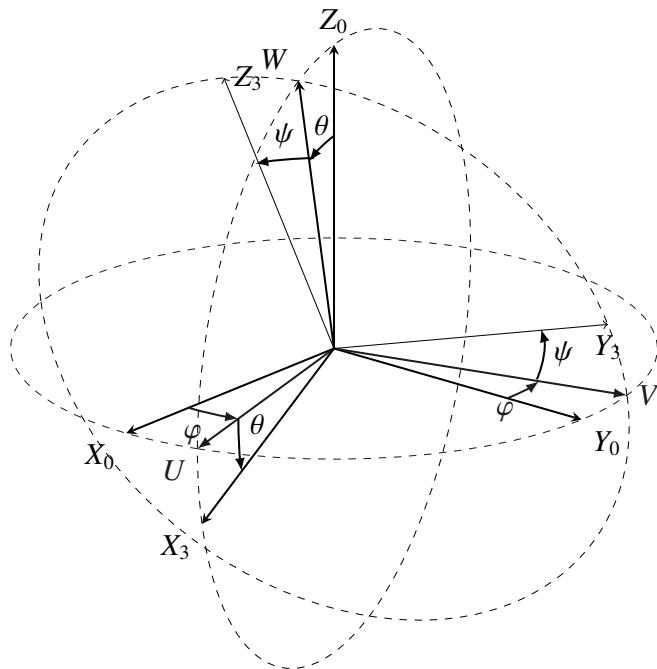


Figure 9 - Visualisation des angles de Cardan

Les formules donnent alors :

$$R = R_3 R_2 R_1 = \begin{pmatrix} \cos(\varphi) \cos(\theta) & \cos(\varphi) \sin(\theta) \sin(\psi) - \sin(\varphi) \cos(\psi) & \cos(\varphi) \sin(\theta) \cos(\psi) + \sin(\varphi) \sin(\psi) \\ \sin(\varphi) \cos(\theta) & \sin(\varphi) \sin(\theta) \sin(\psi) + \cos(\varphi) \cos(\psi) & \sin(\varphi) \sin(\theta) \cos(\psi) - \cos(\varphi) \sin(\psi) \\ -\sin(\theta) & \cos(\theta) \sin(\psi) & \cos(\theta) \cos(\psi) \end{pmatrix}.$$

Ainsi, on peut caractériser R par les trois angles φ , θ et ψ .

Q5. Exemple : on considère la matrice $A = \begin{pmatrix} \sqrt{2}/2 & 1/2 & 1/2 \\ 0 & \sqrt{2}/2 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 1/2 & 1/2 \end{pmatrix}$.

- a) Justifier que A est la matrice d'une rotation de \mathbb{R}^3 .
- b) Montrer l'existence et l'unicité des angles de Cardan pour cette matrice A et les déterminer.

Q6. Illustration du blocage de Cardan : on considère la matrice $B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix}$.

- a) Justifier que B est la matrice d'une rotation de \mathbb{R}^3 .
- b) Montrer que la détermination des angles de Cardan pour la matrice B équivaut à trouver $\varphi \in [-\pi/2, \pi/2]$, $\theta \in]-\pi, \pi]$ et $\psi \in]-\pi, \pi]$, tels que $\theta = \pi/2$, $\sin(\psi - \varphi) = 1$ et $\cos(\psi - \varphi) = 0$.

- c) Sachant que ces angles sont utilisés pour redresser la caméra par trois rotations successives, en quoi cela pose-t-il problème ?

II.2 - Les quaternions

Pour éviter le blocage de Cardan, on peut utiliser les quaternions pour représenter une rotation de \mathbb{R}^3 .

Une manière de représenter un quaternion est d'utiliser des matrices de l'espace vectoriel $\mathcal{E} = M_4(\mathbb{R})$.

La transposée d'une matrice M de \mathcal{E} sera notée M^T .

Pour a, b, c et d réels, on considère la matrice $M(a, b, c, d) = \begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix}$.

On note H l'ensemble des matrices $M(a, b, c, d)$ pour (a, b, c, d) dans \mathbb{R}^4 .

Ainsi $H = \{M(a, b, c, d) \mid (a, b, c, d) \in \mathbb{R}^4\}$.

Les éléments de H sont appelés quaternions.

On pose enfin $I_4 = M(1, 0, 0, 0)$ (matrice identité de \mathcal{E}), $E_1 = M(0, 1, 0, 0)$, $E_2 = M(0, 0, 1, 0)$ et $E_3 = M(0, 0, 0, 1)$.

Q7. Montrer que H est un sous-espace vectoriel de \mathcal{E} .

Q8. Montrer que la famille (I_4, E_1, E_2, E_3) est libre. En déduire une base et la dimension de H .

Q9. Produit de quaternions

- a) Calculer les produits suivants : $E_1 I_4$, E_1^2 , $E_1 E_2$ et $E_1 E_3$.

Pour la suite, on pourra utiliser directement le tableau suivant qui donne le résultat du produit AB :

$A \backslash B$	I_4	E_1	E_2	E_3
I_4	I_4	E_1	E_2	E_3
E_1	?	?	?	?
E_2	E_2	$-E_3$	$-I_4$	E_1
E_3	E_3	E_2	$-E_1$	$-I_4$

- b) Justifier rapidement que, pour tous quaternions Q_1 et Q_2 (c'est-à-dire des éléments de H), $Q_1 Q_2$ est aussi un quaternion.

Q10. Soit Q un quaternion (c'est-à-dire un élément de H), que l'on écrit sous la forme :

$Q = aI_4 + bE_1 + cE_2 + dE_3$ avec a, b, c et d des réels. On définit $N(Q) = \sqrt{a^2 + b^2 + c^2 + d^2}$.

- a) Montrer que la transposée Q^T de Q appartient à H et donner ses coordonnées dans la base (I_4, E_1, E_2, E_3) de H .
- b) Démontrer que $QQ^T = N(Q)^2 I_4$.
- c) Montrer qu'il existe un entier p (à préciser) tel que $|\det(Q)| = N(Q)^p$.
- d) Toutes les matrices de H sont-elles inversibles ?

Q11. On considère un quaternion Q que l'on suppose unitaire, c'est à dire qu'il vérifie $N(Q) = 1$.

On définit l'application u_Q par $u_Q(M) = QMQ^T$ pour tout M dans H .

- Justifier que u_Q définit un endomorphisme de H .
On pourra utiliser **Q9** et **Q10**.
- Montrer que I_4 et Q sont des vecteurs propres de u_Q associés à une même valeur propre que l'on précisera.
- Dans cette question, on suppose que Q n'est pas de la forme kI_4 avec $k \in \mathbb{R}$.
Déduire de la question précédente un vecteur propre Q' de u_Q appartenant à $\text{Vect}(E_1, E_2, E_3)$, sous-espace vectoriel de \mathcal{E} engendré par (E_1, E_2, E_3) .
(On pourra écrire Q sous la forme $Q = aI_4 + bE_1 + cE_2 + dE_3$ avec a, b, c et d des réels).

Q12. Étude d'un exemple

On définit $Q = \frac{1}{2}(I_4 + E_1 + E_2 - E_3)$ et on note toujours u_Q l'endomorphisme associé à Q : pour tout M dans H , on a $u_Q(M) = QMQ^T$.

On note A_Q la matrice de u_Q dans la base (I_4, E_1, E_2, E_3) de H .

- Vérifier que $N(Q) = 1$. En déduire, à l'aide de **Q11**, la première colonne de la matrice A_Q .
- Déterminer les termes manquants, notés $*$, de la matrice $A_Q = \begin{pmatrix} * & 0 & 0 & 0 \\ * & 0 & * & 0 \\ * & 0 & * & -1 \\ * & -1 & * & 0 \end{pmatrix}$.
- En déduire que le sous-espace vectoriel $F = \text{Vect}(E_1, E_2, E_3)$ de H est stable par u_Q et donner, sans aucun calcul, la matrice de l'endomorphisme f induit par u_Q sur F .
- À l'aide des **Q12c** et **Q11**, trouver un vecteur directeur de l'axe de la rotation de matrice B vue en **Q6**.

On pourrait démontrer qu'à toute matrice de rotation R de \mathbb{R}^3 , on peut associer un unique (au signe près) quaternion Q unitaire qui la représente.

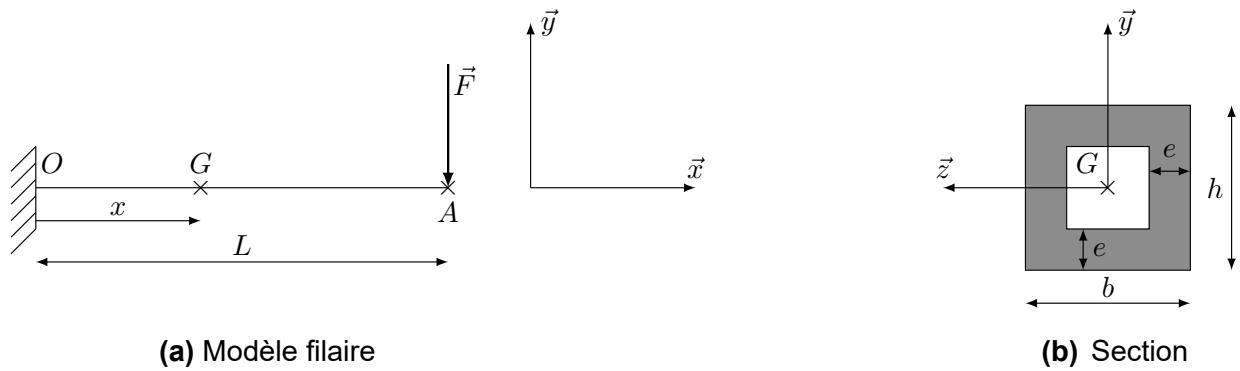
Les équations permettant le calcul du quaternion sont polynomiales de degré deux et sont plus facilement implémentables sur un calculateur embarqué.

Partie III - Optimisation du matériau du bras 1

La raideur du bras 1 est un paramètre important pour la qualité de l'image. En effet, une raideur trop faible entraînerait des vibrations au niveau de la caméra que le Slick aurait du mal à corriger si les fréquences sont grandes. De plus, la masse du Slick ne doit pas être trop importante pour faciliter son utilisation. C'est pourquoi, nous allons déterminer le matériau qui permet d'obtenir le meilleur compromis. D'autres critères interviennent dans le choix du matériau :

- la limite élastique doit être supérieure à 140 MPa afin d'éviter toute déformation plastique lors de "petites" chutes ;
- le matériau doit résister à l'eau (douce et salée) et aux UV.

Pour simplifier l'étude, on modélisera le bras 1 comme une poutre droite encastrée à une extrémité et soumise à une force $\vec{F} = -F\vec{y}$ à l'autre avec $F > 0$ (**figure 10**).

**Figure 10 - Modélisation du bras 1****Notations (figure 10)**

- L est la longueur de la poutre, $\overrightarrow{OA} = L\vec{x}$;
- G est le centre d'inertie d'une section, $\overrightarrow{OG} = x\vec{x}$, $x \in [0, L]$;
- la section est un tube rectangulaire de hauteur h , de largeur b et d'épaisseur e ;
- I est le moment quadratique de la section suivant l'axe (G, \vec{z});
- m est la masse du bras 1;
- ρ est la masse volumique du matériau du bras 1;
- E est le module de Young du matériau du bras 1.

Q13.

- Déterminer l'expression du moment fléchissant $M_f(x)$ en fonction de F , L et de x .
- On note $v(x)$ la déformée de la poutre. Rappeler la relation entre la dérivée seconde de $v(x)$, $M_f(x)$, E et I . Donner les deux conditions aux limites qu'imposent la liaison encastrement.
- Donner l'expression de la déformée $v(x)$ en fonction de F , L , E , I et de x . En déduire l'expression de $v(x = L)$.

On note K la raideur de la poutre : $K = \frac{F}{|v(x = L)|}$.

Q14. Donner l'expression de la raideur K en fonction de E , I et de L .**Q15.**

- Donner l'expression du moment quadratique I en fonction de b , h et de e .
- On a $e \ll b$ et $e \ll h$. En ne gardant que les termes prépondérants, montrer que I peut s'écrire :

$$I \approx \frac{(3b + h)h^2e}{6}. \quad (6)$$

De même, on peut exprimer la masse m en ne gardant que les termes prépondérants :

$$m \approx 2\rho e(b + h)L. \quad (7)$$

Le cahier des charges impose les paramètres géométriques b , h et L ainsi que la raideur K .

Vous allez exprimer la masse m en fonction des paramètres matériaux et de ceux imposés.

Q16. En utilisant les équations (6) et (7), donner des expressions des fonctions f et g et du réel α telle que la masse m s'écrive,

$$m = \alpha \times f(\rho, E) \times g(b, h, L, K). \quad (8)$$

On note I_p l'indice de performance du matériau : $I_p = \frac{E}{\rho}$.

La **figure 11** donne le diagramme (log-log) du module de Young en fonction de la masse volumique de matériaux compatibles avec le procédé de fabrication du bras 1.

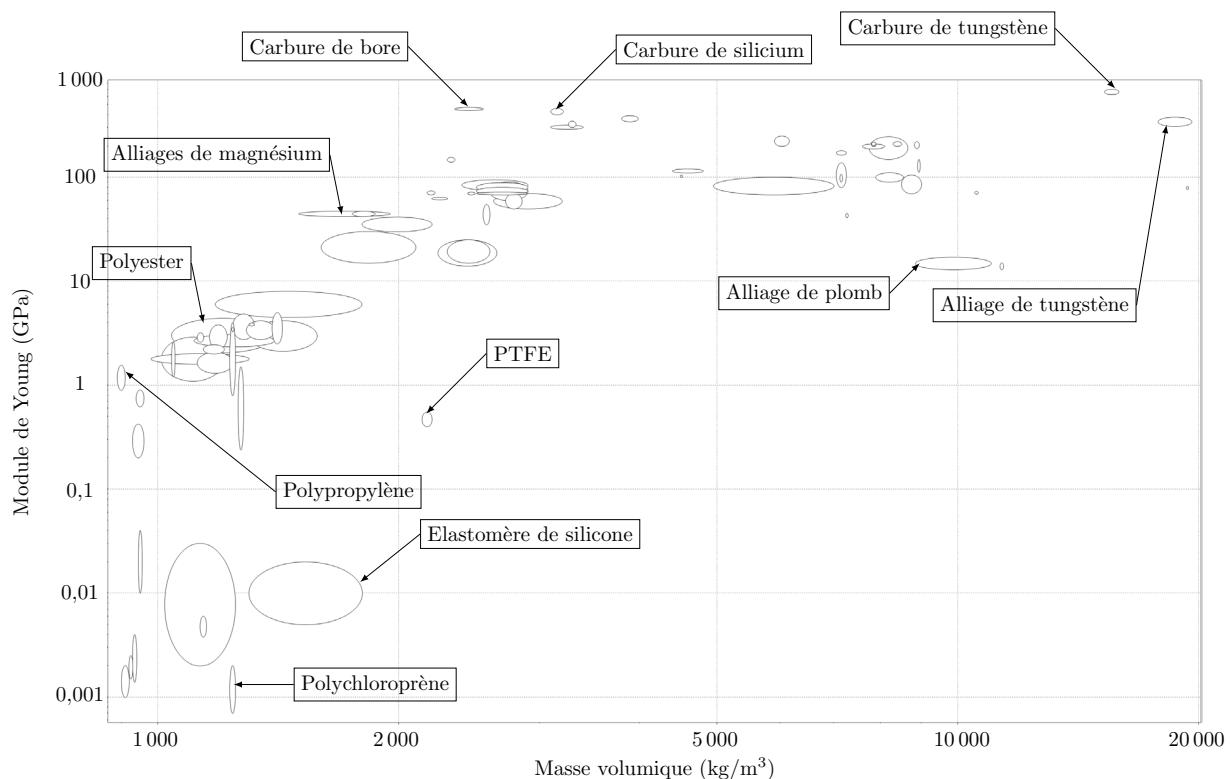


Figure 11 - Diagramme (log-log) du module de Young en fonction de la masse volumique

Q17. Expliquer et démontrer pourquoi il faut tracer une droite de pente 1 et choisir la plus grande ordonnée à l'origine pour avoir le meilleur indice de performance. Choisir un matériau.

FIN



ÉPREUVE SPÉCIFIQUE - FILIÈRE TPC

MODÉLISATION

Durée : 4 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

RAPPEL DES CONSIGNES

- Utiliser uniquement un stylo noir ou bleu foncé non effaçable pour la rédaction de votre composition ; d'autres couleurs, excepté le vert, peuvent être utilisées, mais exclusivement pour les schémas et la mise en évidence des résultats.
- Ne pas utiliser de correcteur.
- Écrire le mot FIN à la fin de votre composition.

Les calculatrices sont autorisées.

Le sujet est composé de trois parties et d'une Annexe en fin de sujet.

Étude d'une réaction à solide consommable

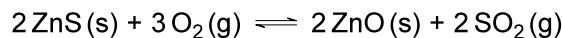
Présentation du problème

Les réactions à solide consommable sont courantes dans les procédés de transformation de la matière. On peut citer la réaction du dioxyde d'uranium (UO_2) avec l'acide fluorhydrique (HF) pour produire le tétrafluorure d'uranium (UF_4) selon la stœchiométrie suivante :



Cette réaction est impliquée dans la conversion du dioxyde d'uranium en hexafluorure d'uranium (UF_6) qui est utilisé dans les deux principaux procédés d'enrichissement de l'uranium (la diffusion gazeuse et l'ultracentrifugation) en raison de son point triple à 64°C (à 150 kPa).

Un autre exemple est la réaction d'oxydation du sulfure de zinc en présence d'air pour former l'oxyde de zinc, un matériau semi-conducteur qui pourrait remplacer le dioxyde de titane dans les cellules photovoltaïques. La réaction s'écrit :



Dans les deux cas, la réaction met en jeu une phase fluide et une phase solide. Une particularité de ce type de réactions est la formation d'un produit solide poreux qui remplace progressivement le réactif initial. Dans le cas de particules sphériques, on observe un déplacement du front de réaction (l'interface entre le réactif solide non consommé et le produit solide) en fonction du temps, dirigé de l'extérieur de la particule vers son centre (**figure 1**).

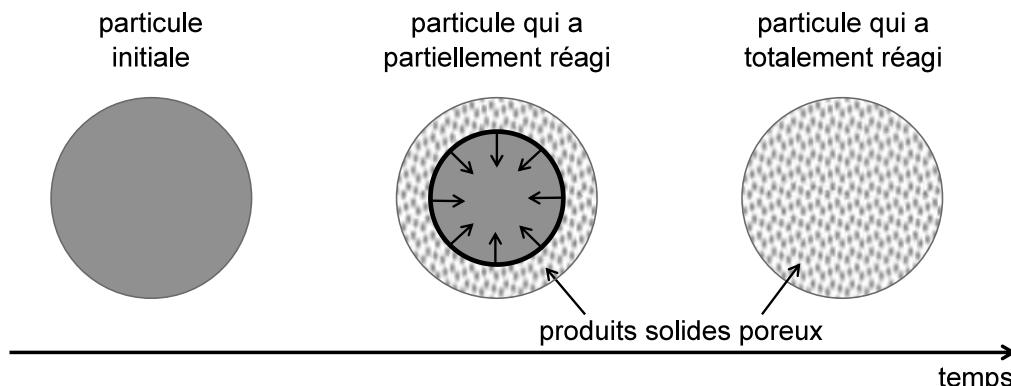


Figure 1 - Représentation schématique de l'évolution du front de réaction en fonction du temps

Les phénomènes physiques et chimiques mis en jeu lors de la réaction sont :

- le transport du fluide réactif à travers la couche limite externe des particules (généralement par convection) ;
- le transport du fluide réactif à travers la couche de produit solide poreux par diffusion dans les pores ;
- la réaction chimique qui est localisée au niveau du front de réaction ;
- le transport du produit fluide (s'il existe) dans les pores, puis à travers la couche limite externe.

Le modèle qui permet de représenter ces phénomènes porte le nom de *modèle à cœur rétrécissant*. Dans une version simple de ce modèle, on suppose que le produit poreux a une densité similaire à celle du réactif, si bien que le volume final de la particule qui a totalement réagi est le même que celui de la particule initiale. Dans le cas général, la consommation des particules est contrôlée par les trois premiers phénomènes cités précédemment. Cependant, dans certains cas limites, il se peut que l'un des trois phénomènes soit beaucoup plus lent que les deux autres et c'est celui-ci qui va imposer la vitesse de consommation des particules. On parle alors de *phénomène limitant*.

Dans la suite, la réaction entre les particules solides (B) et la phase fluide (A) sera écrite de la manière suivante où v est le nombre stœchiométrique relatif au solide :



Le but de cette épreuve est de mettre en équations le modèle à cœur rétrécissant permettant de prédire la consommation des particules en fonction du temps.

La première partie consiste, dans un premier temps, à établir l'équation aux dérivées partielles régissant l'évolution de la concentration du fluide A dans la couche de solide poreux B; puis, dans un second temps, à obtenir des résultats dans un cadre simplifié du modèle.

La deuxième partie est relative à la résolution numérique de l'équation aux dérivées partielles dans le cas général (sans hypothèses simplificatrices).

La dernière partie traite de la comparaison des résultats obtenus par les deux méthodes de résolution du problème.

Partie I - Mise en équation du modèle à cœur rétrécissant

L'objectif de cette première partie est d'établir l'équation différentielle qui régit l'évolution de la concentration du fluide A dans la couche de solide poreux B en fonction de l'espace et du temps. Une première version simplifiée du modèle à cœur rétrécissant sera alors déterminée.

I.1 - Étude de la diffusion du fluide dans la couche de produit poreux

Le transport du fluide dans la couche de produit de la réaction a lieu par diffusion dans les pores de ce matériau poreux. La porosité d'un matériau, notée ε , est définie comme le rapport entre le volume des pores et le volume total de matériau poreux. Le volume de matériau poreux inclut le volume de solide (appelé volume structurel) et le volume des pores. La porosité est donc la fraction volumique de vide dans le matériau.

Une étude au laboratoire a permis de déterminer les masses volumiques structurelle et apparente du produit poreux formé au cours de la réaction :

- la masse volumique structurelle, c'est-à-dire sans la porosité, $\rho_s = 2,50 \cdot 10^3 \text{ kg} \cdot \text{m}^{-3}$;
- la masse volumique apparente, tenant compte de la porosité, $\rho_{\text{app}} = 1,25 \cdot 10^3 \text{ kg} \cdot \text{m}^{-3}$.

Q1. Justifier que $\rho_s > \rho_{\text{app}}$.

Donner les valeurs théoriques minimale et maximale de la porosité.

Calculer la porosité ε du produit à partir des valeurs des masses volumiques structurelle et apparente mesurées au laboratoire.

On modélise la particule par un parallélépipède rectangle comme représenté sur le schéma de la **figure 2**. L'épaisseur de la particule est $2e$ et on suppose que la largeur L et la hauteur h sont très grandes devant l'épaisseur.

On note $S = L \times h$ la section latérale et on considère que la particule réagira avec le fluide uniquement par les deux faces latérales de surface S comme représenté sur le schéma de la **figure 2**.

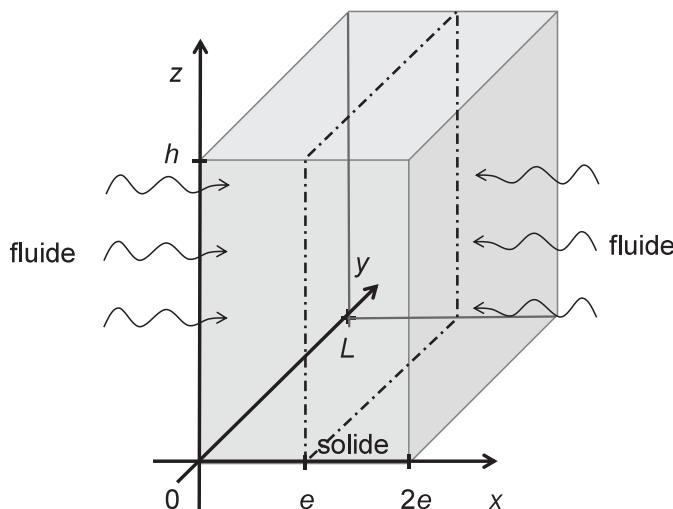


Figure 2 - Schéma de la particule de solide d'épaisseur $2e$

En représentation cartésienne, on note $\vec{j}_N(x, y, z, t)$ le vecteur densité de flux de particules associé aux phénomènes de diffusion que l'on étudie.

Q2. Expliquer pourquoi on peut supposer que $j_N(x, y, z, t) \approx j_N(x, t)$.

Expliquer pourquoi on peut restreindre cette étude à l'intervalle $[0, e]$.

I.2 - Établissement de l'équation de la diffusion de A dans la couche de produit poreux B

Q3. Les phénomènes de diffusion qui nous intéressent sont régis par les équations :

$$\vec{j}_N(x,t) = -D \overrightarrow{\text{grad}}(n(x,t)) \quad (\text{loi de Fick}) \quad \text{et} \quad \frac{\delta N}{dt} = \iint \vec{j}_N \cdot d\vec{S} \quad (\text{flux de } \vec{j}_N).$$

Rappeler la signification physique de la loi de Fick, ainsi que les noms et unités des grandeurs qui la constituent.

On considère un volume élémentaire de la couche de produit poreux de section S et d'épaisseur dx (**figure 3a**). La **figure 3b** est une vue de profil de la **figure 3a**. La **figure 3c** est également une vue de profil sur laquelle le volume structurel et le volume des pores ont été séparés de manière fictive.

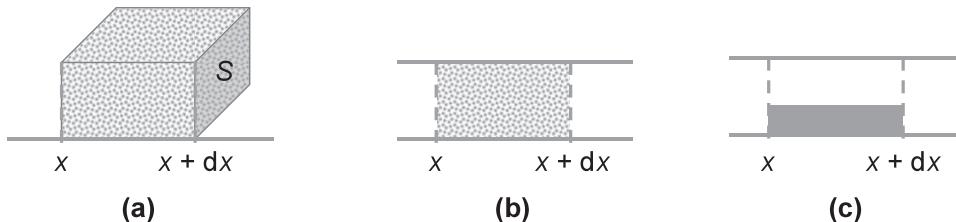


Figure 3 - Volume élémentaire de la couche de produit poreux de section S et d'épaisseur dx

(a) Vue en 3 dimensions (b) Vue de profil

(c) Vue de profil avec séparation fictive du volume structurel et du volume des pores

Q4. Reproduire sur votre copie le schéma du volume élémentaire de la **figure 3c**, en y faisant figurer le vecteur \vec{j}_N aux abscisses x et $x + dx$.

Q5. Réaliser un bilan de matière sur ce volume élémentaire traduisant la conservation du nombre de molécules et en déduire l'équation de la diffusion dans la couche de produit poreux :

$$\varepsilon \frac{\partial n}{\partial t} = D_e \frac{\partial^2 n}{\partial x^2}$$

où D_e est un coefficient de diffusion effectif qui tient compte de la porosité du matériau.

On utilise plus généralement l'équation de la diffusion de A dans la couche de produit poreux en considérant la concentration molaire $C(x,t)$ plutôt que $n(x,t)$:

$$\varepsilon \frac{\partial C}{\partial t} = D_e \frac{\partial^2 C}{\partial x^2}. \quad (1)$$

Cette équation s'accompagne des deux conditions aux limites suivantes :

$$- \text{en } x = 0, \quad D_e \left(\frac{\partial C}{\partial x} \right)_{x=0} = k_D (C_e - C_s) \quad (2)$$

$$- \text{en } x = x_f, \quad D_e \left(\frac{\partial C}{\partial x} \right)_{x=x_f} = k_C C_f \quad (3)$$

avec :

- k_D la constante de transfert du fluide dans la couche limite (en $\text{m} \cdot \text{s}^{-1}$);
- C_e la concentration molaire du fluide dans la phase gazeuse en dehors de la couche limite du fluide (donc $C(x = 0^-) = C_e$);
- C_s la concentration molaire en surface de la particule, en $x = 0$ (donc $C(x = 0^+) = C_s$);
- C_f la concentration molaire au niveau du front de réaction, en $x = x_f$ (donc $C(x = x_f) = C_f$);
- k_C la constante cinétique de la réaction de consommation du solide B par le fluide A (en $\text{m} \cdot \text{s}^{-1}$) (**figure 4**).

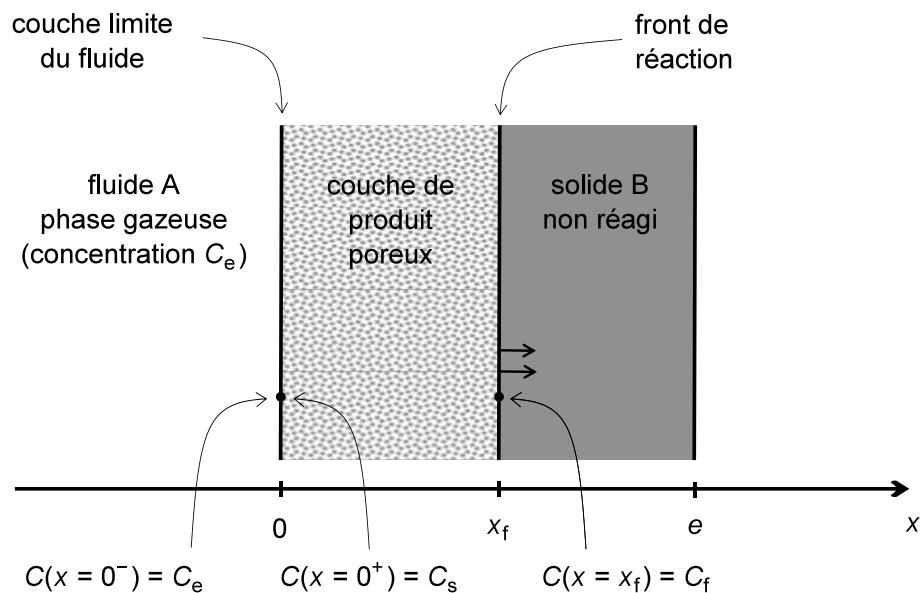


Figure 4 - Schéma de la particule de solide (sur l'intervalle $[0, e]$) partiellement consommée avec la représentation de la couche limite du fluide et du front de réaction

Q6. À l'aide d'une analyse dimensionnelle, vérifier l'homogénéité de l'équation (2).

I.3 - Modèle simplifié : hypothèse du régime stationnaire pour la phase fluide

Pour simplifier la résolution de l'équation aux dérivées partielles, nous supposons que la phase fluide évolue en régime stationnaire. La validité de cette hypothèse sera discutée dans la partie numérique.

Q7. Simplifier l'équation (1) de la diffusion de A dans le cas du régime stationnaire.

On note F_A le flux molaire de A, c'est-à-dire le flux du vecteur densité de flux de particules molaire à travers une section S de particule. Montrer que :

$$F_A = -D_e \frac{dC}{dx} S \quad (4)$$

et justifier que F_A ne dépend pas de x dans la couche de produit poreux.

Q8. On rappelle que $C(x = 0^+) = C_s$ et on note C_x la concentration du fluide A pour une abscisse x comprise dans l'intervalle $[0, x_f]$.

En intégrant l'équation (4) après séparation des variables, montrer que :

$$F_A = \frac{D_e S}{x} (C_s - C_x).$$

L'hypothèse du régime stationnaire pour la phase fluide entraîne l'égalité des flux molaires de A à travers la couche limite externe, dans la couche de produit poreux et consommé par la réaction.

De ce fait, la situation peut être modélisée électriquement, entre la couche limite en $x = 0$ et le front de réaction en $x = x_f$, par un ensemble de 3 résistances placées en série, traversées par le "courant commun" F_A et soumises aux "différences de potentiel" dues aux différences de concentration. Ces 3 résistances (figure 5) sont :

- en $x = 0$, la résistance R_{cl} de la couche limite ;
- entre $x = 0$ et $x = x_f$, la résistance R_{sp} du solide poreux ayant réagi ;
- en $x = x_f$, la résistance R_f due à la réaction chimique, située au niveau du front de la réaction.

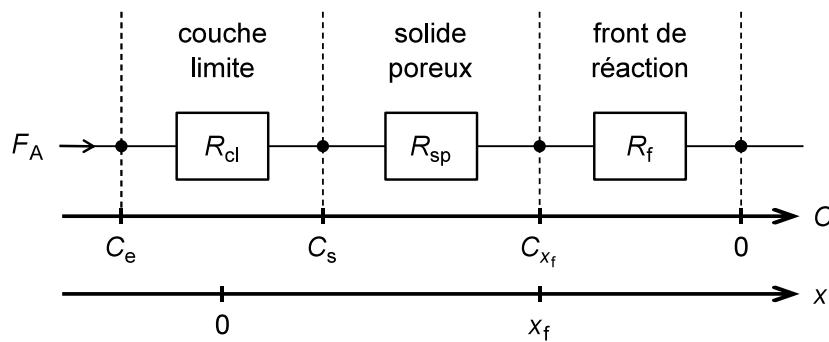


Figure 5 - Schéma du modèle électrique équivalent

- Q9.** On peut déduire de l'étude précédente la relation (5), qui traduit la circulation d'un courant commun aux 3 résistances évoquées :

$$F_A = \frac{C_e - C_s}{\frac{1}{k_D S}} = \frac{C_s - C_{x_f}}{\frac{x_f}{D_e S}} = \frac{C_{x_f} - 0}{\frac{1}{k_C S}} . \quad (5)$$

En déduire l'expression de la résistance équivalente de l'ensemble $R_{\text{éq}}$, telle que $F_A = \frac{C_e - 0}{R_{\text{éq}}}$, en fonction de D_e , k_C , k_D , S et de x_f .

- Q10.** On note N_B la quantité de matière de B.

Expliquer sans démonstration le sens physique de l'équation $F_A = -\frac{1}{v} \frac{dN_B}{dt}$, où v est le nombre stœchiométrique défini dans la présentation du problème.

- Q11.** Soit $N_B(x)$ la quantité de matière de B comprise entre les abscisses x et e .

Pour tout x compris entre x_f et e , exprimer $N_B(x)$ en fonction de ρ_B , M_B , S , x et de e , avec ρ_B la masse volumique de B et M_B sa masse molaire.

En déduire, à l'aide de la Q10, la relation $F_A = \frac{\rho_B S}{v M_B} \frac{dx}{dt}$.

- Q12.** Les expressions précédentes de F_A permettent d'obtenir l'équation à variables séparées :

$$dt = K \left(\frac{1}{k_D} + \frac{x}{D_e} + \frac{1}{k_C} \right) dx, \quad \text{avec } K = \frac{\rho_B}{v C_e M_B} .$$

Montrer que l'intégration de cette équation entre la date $t = 0$ (où le front est en $x = 0$) et la date courante $t = t_f$ (où le front est en x_f) permet d'obtenir la relation :

$$t_f = t_{0e} \frac{x_f}{e} + t_{0i} \left(\frac{x_f}{e} \right)^2 + t_{0c} \frac{x_f}{e}$$

avec :

- $t_{0e} = K \frac{e}{k_D}$, le temps caractéristique de transport dans la couche limite externe ;
- $t_{0i} = K \frac{e^2}{2 D_e}$, le temps caractéristique de transport interne par diffusion ;
- $t_{0c} = K \frac{e}{k_C}$, le temps caractéristique de la réaction chimique.

- Q13.** Sachant que les étapes précédentes sont successives, donner la relation entre la durée de consommation totale des particules, notée t_0 , et les trois temps caractéristiques t_{0e} , t_{0i} et t_{0c} .

Q14. Une expérience réalisée sur des particules d'épaisseur e_1 a permis de mesurer les valeurs suivantes : $t_{0e} = 60,0 \text{ s}$, $t_{0i} = 300 \text{ s}$ et $t_{0c} = 120 \text{ s}$.

Calculer la durée totale de consommation t_{01} .

En déduire la durée totale de consommation t_{02} pour une épaisseur $e_2 = 2e_1$.

L'analyse des valeurs des trois temps caractéristiques permet de mettre en évidence le poids de chaque phénomène dans la résistance au transfert de matière. Lorsque l'un de ces temps caractéristique est très supérieur aux autres, il impose son régime que l'on appelle alors le régime limitant.

Dans le cas de cette étude, le régime limitant peut donc être :

- le transport externe, si $t_{0e} \gg t_{0i}$ et $t_{0e} \gg t_{0c}$;
- la diffusion interne dans le produit poreux, si $t_{0i} \gg t_{0e}$ et $t_{0i} \gg t_{0c}$;
- la réaction chimique, si $t_{0c} \gg t_{0e}$ et $t_{0c} \gg t_{0i}$.

Q15. En utilisant la Q12, déterminer les 2 inégalités à satisfaire (l'une en fonction de D_e , k_D et de e et l'autre en fonction de D_e , k_C et de e) pour que le régime limitant soit le transport interne.

Partie II - Résolution numérique du problème

Une résolution numérique préalable de l'équation de diffusion du fluide A (équation (1)) a été faite en régime quelconque. Par comparaison, on souhaite valider la simplification de cette résolution dans le cadre suivant :

- hypothèse 1 : on suppose que le transport interne constitue le régime limitant ;
- hypothèse 2 : on se place en régime stationnaire pour le fluide A.

L'ensemble des paramètres physico-chimiques nécessaires à cette résolution est défini dans le **code Python 1**. Dans la suite, il est inutile de recopier ce préambule. Toutes les variables qui y sont définies sont supposées globales.

```

1 import numpy as np
2
3 # Paramètres du problème
4 e = 1.00e-3    # e demi-épaisseur [m]
5 M = 97.5e-3    # M_B masse molaire du solide [kg/mol]
6 rho = 4.13e3   # rho_B masse volumique du solide [kg/m^3]
7 eps = 0.500    # epsilon porosité du solide
8 De = 1.25e-6   # D_e coefficient de diffusion du fluide dans le solide [m^2/s]
9 kC = 100        # k_C constante cinétique de la réaction [m/s]
10 kD = 100       # k_D constante de transfert [m/s]
11 nu = 1         # nu nombre stœchiométrique
12 R = 8.31        # R constante des gaz parfaits [J/K/mol]
13
14 # Conditions opératoires
15 P = 2.50e5      # pression [Pa]
16 T = 900 + 273   # température [K]
17 xO2 = 0.210     # fraction molaire de O2
18 xN2 = 0.790     # fraction molaire de N2

```

Code Python 1 - Préambule Python

Le **tableau 1** contient une partie des grandeurs mathématiques et des objets Python utilisés dans la suite. Certaines grandeurs, moins importantes ou faisant l'objet d'une question spécifique, ne figurent volontairement pas dans ce tableau.

Symbol mathématique	Objet Python	Signification
$C(x,t)$	vec_C	concentration du fluide A
	vec_Cprec	vecteur des concentrations
	vec_Y	copie du vecteur vec_C
		vecteur des indices du front de réaction
x	vec_x	abscisse
	mat_x	vecteur des abscisses
N	N	nombre d'intervalles
Δx	Dx	pas d'espace
i	i	indice spatial
i_{fr}	ifr	indice du front de réaction
t	vec_t	date
	k	vecteur des dates
	Dt	indice temporel
	ItMax	pas temporel
	ItMax	nombre maximum d'itérations

Tableau 1 - Tableau de correspondance entre grandeurs mathématiques et objets Python

Q16. Sachant que le solide réagit uniquement avec le dioxygène, calculer la concentration C_e du fluide (en $\text{mol} \cdot \text{m}^{-3}$) dans la phase gazeuse à l'aide des valeurs numériques figurant dans le **code Python 1**.

Q17. Écrire une fonction $\text{Cgaz}(x, P, T)$ qui prend en arguments la fraction molaire x , la pression totale P et la température T d'un gaz et renvoie sa concentration molaire C (en $\text{mol} \cdot \text{m}^{-3}$).

Q18. On admet que le transport interne est le régime limitant si $e \gg \frac{2D_e}{k_D}$ et $e \gg \frac{2D_e}{k_C}$.

Vérifier que le transport interne est bien le régime limitant (et que l'hypothèse 1 est donc bien satisfaite).

Q19. Sachant que $K = 7,86 \cdot 10^3$ (définie à la **Q12**), déduire des expressions de la **Q12** et des données numériques du **code Python 1**, une estimation en secondes de la durée totale $t_0 \approx t_{0i}$ de consommation.

Pour résoudre numériquement l'équation aux dérivées partielles de diffusion, on utilise la méthode des différences finies avec un schéma explicite.

La première étape consiste à discréteriser l'équation (1) dans l'espace et dans le temps.

Pour discréteriser l'espace, on choisit de découper l'intervalle $[0, e]$ en 30 parties de même longueur (**figure 6**). On utilise l'indice i (compris entre 0 et N) pour identifier le i^{e} point de l'intervalle, tel que $x_i = i\Delta x$.

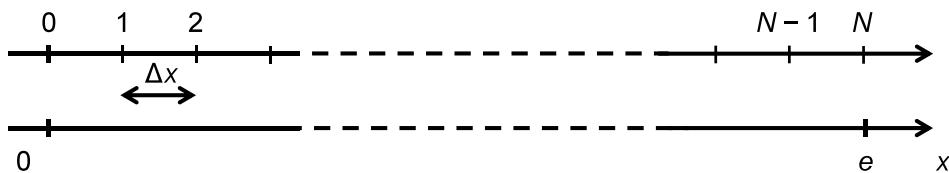


Figure 6 - Discréterisation spatiale dans la direction x

Q20. Préciser la valeur de N .

Donner le code permettant de calculer le pas d'espace Δx (variable Python : `Dx`).

Donner le code permettant de fabriquer le vecteur `vec_x` qui contiendra les abscisses x .

Pour obtenir l'évolution de la concentration en chaque point de l'espace en fonction du temps, on discréterise le temps en intervalles de durée Δt , appelé le pas de temps (variable Python : `Dt`). On utilise l'indice k pour identifier la date particulière t_k , telle que $t_k = k\Delta t$. L'indice k commence à 0 et peut prendre $ItMax$ valeurs (variable Python : `ItMax`). $ItMax$ est fixée par l'utilisateur et représente le nombre maximum d'itérations.

La discréterisation de l'équation aux dérivées partielles est obtenue en écrivant des développements limités. Dans la suite, on note $C_{i,k}$ la concentration au point d'abscisse x_i à la date t_k . Ainsi,

- $C_{i,k+1}$ représente la concentration au point d'abscisse x_i à la date $t_{k+1} = t_k + \Delta t$;
- $C_{i+1,k}$ représente la concentration au point d'abscisse $x_{i+1} = x_i + \Delta x$ à la date t_k .

Q21. À l'aide d'un développement limité, donner l'expression de $C(x, t + \Delta t)$ à l'ordre 1 en fonction de $C(x, t)$ et de $\left(\frac{\partial C}{\partial t}\right)_{x,t}$, la dérivée partielle de $C(x, t)$ par rapport au temps évaluée en x à la date t .

Q22. Donner alors l'expression de $\left(\frac{\partial C}{\partial t}\right)_{x_i,t_k}$, la dérivée partielle de $C(x, t)$ par rapport au temps évaluée à l'abscisse x_i à la date t_k , en fonction de $C_{i,k}$, $C_{i,k+1}$ et Δt .

L'utilisation de deux développements limités conduit à l'expression discrétisée (admise) suivante pour l'approximation de la dérivée seconde de C par rapport à x :

$$\left(\frac{\partial^2 C}{\partial x^2} \right)_{x_i, t_k} \approx \frac{C_{i+1,k} - 2C_{i,k} + C_{i-1,k}}{(\Delta x)^2}. \quad (6)$$

Q23. Préciser sans justification l'ordre auquel ont été faits les développements limités qui ont permis d'obtenir l'expression approchée donnée par l'équation (6).

Q24. Montrer que l'équation de diffusion (1) peut se mettre sous la forme :

$$C_{i,k+1} = C_{i,k} + r(C_{i+1,k} - 2C_{i,k} + C_{i-1,k}) \quad (7)$$

et donner l'expression de la variable r en fonction du coefficient de diffusion D_e , de la porosité ε , du pas de temps Δt et du pas d'espace Δx .

Q25. Pour des raisons de convergence numérique, on admet que la quantité $\frac{D_e}{\varepsilon} \frac{\Delta t}{(\Delta x)^2}$ doit être inférieure à 1 / 2.

En déduire la valeur maximale de Δt permettant la convergence numérique (on dispose des données du **code Python 1** et on admet que $\Delta x = 3,33 \cdot 10^{-5}$ m).

Dans la suite, on choisit le pas de temps $\Delta t = 1,00 \cdot 10^{-4}$ s.

Q26. Avec ce pas de temps et en supposant une durée totale de consommation $t_0 \approx 3,15 \cdot 10^3$ s, donner une estimation du nombre d'itérations nécessaires pour réaliser la simulation.

Au fur et à mesure de l'avancement de la réaction, le front de réaction se déplace de $x = 0$ à $x = e$. Lors de ce déplacement du front de réaction, la particule est constituée :

- pour $0 \leq x < x_f$, d'une partie qui a réagi et dans laquelle diffuse le réactif fluide ;
- pour $x_f \leq x < e$, d'une partie qui n'a pas encore réagi et dans laquelle la concentration en réactif fluide est nulle.

L'indice spatial correspondant à la position du front de réaction sera noté i_{fr} (variable Python : `ifr`).

L'équation (7) est valable uniquement pour les indices i compris entre 1 et $i_{fr} - 1$.

Pour les indices $i = 0$ et $i = i_{fr}$, il est nécessaire d'utiliser les conditions aux limites (équations (2) et (3)) pour calculer les valeurs particulières de $C_{0,k}$ et $C_{i_{fr},k}$. La discrétisation de ces conditions aux limites conduit aux relations de récurrence suivantes :

- en $x = 0$, $C_{0,k} = p_1 C_e + p_2 C_{1,k}$,
- en $x = x_f$, $C_{i_{fr},k} = p_3 C_{i_{fr}-1,k}$.

Q27. Traduire l'équation (3) en appliquant la méthode des différences finies.

En déduire l'expression du paramètre p_3 en fonction de Δx , k_C et D_e .

Pour résoudre numériquement l'équation (1), on utilise l'algorithme décrit par le **code Python 2** dont les étapes principales sont les suivantes :

- initialisation des variables ;
- boucle d'intégration :
 - calcul de l'indice i_{fr} correspondant à la position du front de réaction ;
 - calcul des concentrations à chaque pas d'espace à l'aide des relations de récurrence ;
 - stockage des concentrations dans une matrice.

Certaines portions du **code Python 2** sont volontairement absentes et sont signalées par « ».

```

1  # INITIALISATIONS
2  ItMax = 32000000
3  mat_C = np.zeros((N + 1, 1 + ItMax // 100000)) # Voir question Q32
4  vec_t = np.zeros((1 + ItMax // 100000))           # Voir question Q32
5  ---
6
7  # DÉFINITIONS
8  def Cgaz(x, P, T):                                # Voir question Q17
9      ---
10
11 def calcul_ifr():                                    # Voir questions Q30 et Q31
12     ---
13
14 def calcul_concentrations():                      # Voir question Q29
15     ---
16
17 def stockage(k):                                    # Voir question Q33
18     j = k // 100000
19     mat_C[:, j] = vec_C
20     vec_t[j] = k * Dt
21
22 # BOUCLE PRINCIPALE
23 for k in range(____):                             # Voir question Q28
24     # initialisations locales
25     ---
26
27     # calcul de l'indice du front de réaction
28     ifr = calcul_ifr()
29
30     # calcul des concentrations à la date k
31     calcul_concentrations(ifr)
32
33     # stockage partiel des données
34     if k % 10000 == 0:                            # Voir question Q33
35         stockage(k)

```

Code Python 2 - Structure du code

Q28. Sachant que l'on connaît le nombre maximal d'itérations, compléter la ligne 23 du **code Python 2**.

Q29. Le **code Python 3** permet de calculer les concentrations en tout point de l'espace à la date t_{k+1} , à partir des concentrations en tout point de l'espace à la date t_k . Les concentrations sont stockées dans le vecteur `vec_C` préalablement défini, tout comme les constantes `r`, `p1`, `p2`, `p3` et `ifr`.

Expliquer ce code.

```

1  vec_C[0] = p1 * Ce + p2 * vec_C[1]
2  for i in range(1, ifr):
3      vec_C[i] = r * vec_Cprec[i - 1] + (1 - 2 * r) * vec_Cprec[i] + r * vec_Cprec[i + 1]
4  vec_C[ifr] = p3 * vec_C[ifr - 1]
5  vec_Cprec = vec_C

```

Code Python 3 - Calcul de la concentration du fluide A

L'indice du front de réaction, i_{fr} , correspond à la partie entière du rapport entre l'épaisseur de la couche de produit poreux, notée Y , et le pas d'espace Δx . Pour l'obtenir, il faut calculer Y à chaque itération. Pour cela, on utilise l'expression discrétisée du bilan de matière en régime transitoire sur le réactif solide, qui s'écrit :

$$\frac{Y_{k+1} - Y_k}{\Delta t} = -D_e V_{mol,B} \frac{C_{i_{fr},k+1} - C_{i_{fr}-1,k+1}}{\Delta x} \quad (8)$$

avec Y_k l'épaisseur de la couche de produit poreux à la date t_k (c'est aussi la position du front de réaction) et $V_{mol,B}$ le volume molaire de B (variable Python : `VmolB`).

Q30. Toutes les variables nécessaires étant définies, donner le code correspondant au calcul de Y_{k+1} , valeur de Y à la date t_{k+1} .

Q31. Toutes les variables nécessaires étant définies, donner le code permettant de calculer la valeur de l'indice du front de réaction i_{fr} à partir de la valeur de Y_k et de Δx (cette valeur est renvoyée par la fonction `calcul_ifr()` du **code Python 2**).

Le nombre d'itérations étant très grand, on ne souhaite pas enregistrer les valeurs des concentrations à toutes les dates, mais seulement toutes les 100 000 itérations. Pour ces itérations, on enregistre :

- la valeur des concentrations dans la matrice `mat_C`;
- la date dans le vecteur `vec_t`.

Q32. Justifier les dimensions de la matrice `mat_C` et du vecteur `vec_t` qui sont définis dans les initialisations du **code Python 2**.

Donner les dimensions du vecteur `vec_Y`.

Q33. Expliquer la ligne 34 du **code Python 2** et donner les 2 premières valeurs de k pour lesquelles la fonction `stockage(k)` est appelée.

Partie III - Comparaison entre les deux modèles

Les données issues de la simulation faite sans approximation ont été stockées dans la variable `Dsimu`. Il s'agit d'une liste de tuples contenant les triplets de la position, de la date et de la concentration :

```
Dsimu = [(x0, t0, C0), (x1, t1, C1), ...].
```

On souhaite trier ces données en fonction de la date, qui est la 2^e composante de chaque tuple. Pour cela, on va modifier le code de tri suivant :

```
1 def tri(T):
2     "Tri en place le tableau T"
3     for i in range(1, len(T)):
4         x = T[i]
5         j = i
6         while j > 0 and T[j - 1] > x: # Voir question Q35
7             T[j] = T[j - 1]
8             j = j - 1
9         T[j] = x
```

Code Python 4 - Code de tri

Q34. Préciser sans justification si le **code Python 4** est un tri par insertion, un tri fusion ou un tri rapide.

Expliquer pourquoi il ne s'agit pas d'un tri récursif.

Q35. Proposer une modification de la ligne 6 du **code Python 4** pour trier les données stockées dans la variable `Dsimu` sur la 2^e composante (la date).

Afin de comparer les modèles, les taux de conversion du solide B ($X_B = x_f / e$), obtenus par les deux méthodes déjà évoquées, ont été tracés en fonction du temps (**figure 7**) :

- en trait pointillé : résolution numérique sans approximation ;
- en trait plein : résolution numérique avec les approximations de la partie II.

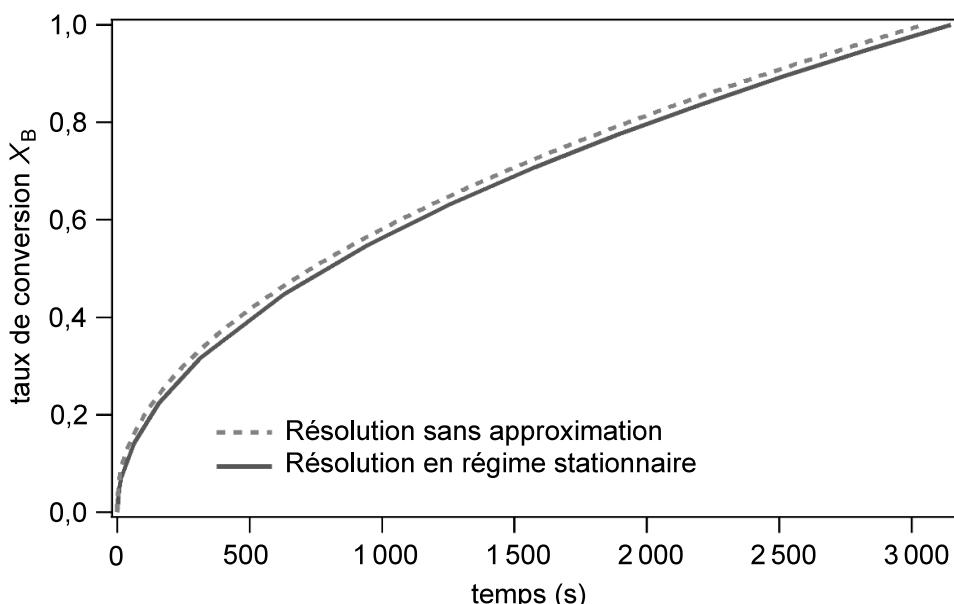


Figure 7 - Comparaison des taux de conversion obtenus par les deux modèles

La résolution sans approximation de l'équation aux dérivées partielles prédit une consommation totale du solide B après 30 417 663 itérations.

Q36. À partir de la donnée du nombre réel d'itérations, déterminer la durée totale de consommation.

Comparer cette valeur à celle obtenue en faisant l'hypothèse du régime stationnaire pour le fluide, $t_0 = 3,15 \cdot 10^3$ s.

Conclure sur l'accord entre les deux modèles.

Q37. En considérant que l'hypothèse du régime stationnaire pour le gaz est acceptable et que le phénomène limitant est la diffusion du réactif fluide dans la couche de produit poreux, dessiner sur un même graphe l'allure des profils de concentration (les courbes $C(x)$) du réactif fluide entre $x = 0$ et $x = e$ pour trois configurations différentes :

- une particule qui n'a pas réagi ;
- une particule qui a totalement réagi ;
- une particule qui a partiellement réagi.

On fera figurer les valeurs des concentrations et les abscisses caractéristiques sur le graphe.

ANNEXE

Bibliothèque numpy de Python

Import de la bibliothèque **numpy** :

```
>>> import numpy as np
```

Création d'un tableau **numpy** à 2 lignes et 3 colonnes, appelé **M** :

```
>>> M = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(M)
[[1.5 2. 3.]
 [4. 5. 6.]]
```

Accès à un élément de **M** :

```
>>> M = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(M[0, 0])
1.5
>>> M[1, 2] = -7
>>> print(M)
[[ 1.5 2. 3.]
 [ 4. 5. -7.]]
```

Sélection d'une ligne ou d'une colonne de **M** :

```
>>> M = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> print(M[:, 1]) # sélection de la colonne 1
[2. 5.]
>>> print(M[1, :]) # sélection de la ligne 1
[4. 5. 6.]
```

Création d'un vecteur nul ou d'une matrice nulle :

```
>>> vec = np.zeros(5)
>>> print(vec)
[0. 0. 0. 0. 0.]
>>> mat = np.zeros((3, 2))
>>> print(mat)
[[0. 0.]
 [0. 0.]
 [0. 0.]]
```

FIN



Epreuve d'Informatique et Modélisation de Systèmes Physiques

Durée 4 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

AVERTISSEMENT

La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction, la clarté et la précision** des raisonnements entreront pour une **part importante** dans l'**appréciation des copies**. En particulier, les résultats non justifiés ne seront pas pris en compte. Les candidats sont invités à encadrer les résultats de leurs calculs.

CONSIGNES :

- Composer lisiblement sur les copies avec un stylo à bille à encre foncée : bleue ou noire.
- L'usage de stylo à friction, stylo plume, stylo feutre, liquide de correction et dérouleur de ruban correcteur est interdit.
- Remplir sur chaque copie en MAJUSCULES toutes vos informations d'identification : nom, prénom, numéro inscription, date de naissance, le libellé du concours, le libellé de l'épreuve et la session.
- Une feuille, dont l'entête n'a pas été intégralement renseigné, ne sera pas prise en compte.
- Il est interdit aux candidats de signer leur composition ou d'y mettre un signe quelconque pouvant indiquer sa provenance.

Présentation de la problématique SUIVI DE LA CONSOMMATION D'EAU D'UNE HABITATION

La consommation d'eau, au même titre que la consommation d'électricité ou de gaz, représente une part non négligeable du budget de nombreux ménages, et contribue de façon significative à leur empreinte environnementale. La facturation de ces consommations étant souvent annuelle, leur suivi tout au long de l'année présente un intérêt aussi bien économique qu'écologique.

1. Principe du suivi

Le suivi de la consommation d'une habitation est réalisé à l'aide d'un compteur d'eau (Figure 1). Il s'agit d'un dispositif monté en série sur l'arrivée d'eau d'une habitation, qui comporte un dispositif mécanique mis en rotation par le passage de l'eau. Si le compteur a été correctement posé et étalonné, à chaque tour de la partie mobile correspond un volume d'eau connu. Un ensemble d'engrenages et de roues à cliquet permet d'afficher le volume consommé sur un cadran.

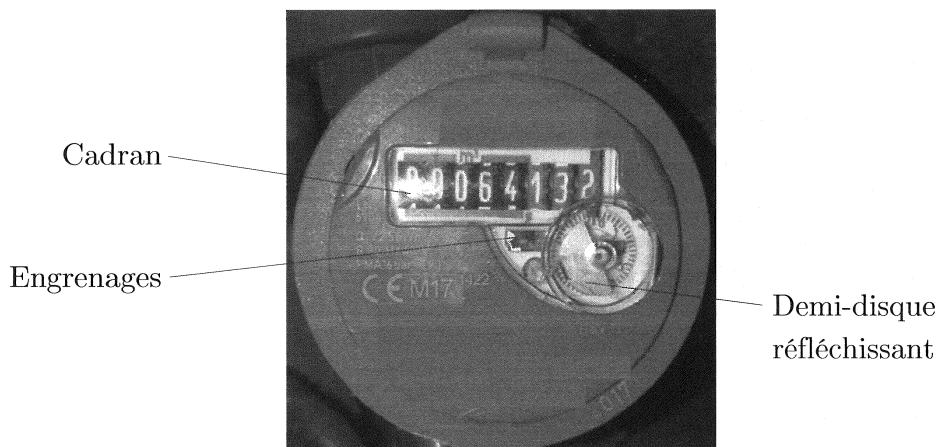


FIGURE 1 – Compteur d'eau installé dans une habitation

Le relevé fréquent d'un compteur étant une opération fastidieuse, son automatisation est d'un intérêt certain pour faciliter le suivi. On peut pour cela utiliser des compteurs dits "intelligents" ou "communicants", ou encore instrumenter un compteur d'eau classique. Dans les deux cas, le principe est le même : il s'agit de compter les tours d'un demi-disque métallique tournant avec la partie mobile au moyen d'un détecteur optique (phototransistor) ou magnétique (interrupteur à lame souple). Ces dispositifs, alimentés sous une tension continue, fournissent l'un comme l'autre une tension présentant une impulsion par tour de la partie mobile (Figure 2).

La mesure en temps réel de la consommation d'eau d'une habitation peut ainsi être réalisée en comptant les impulsions issues du capteur au moyen, par exemple, d'un microcontrôleur. En

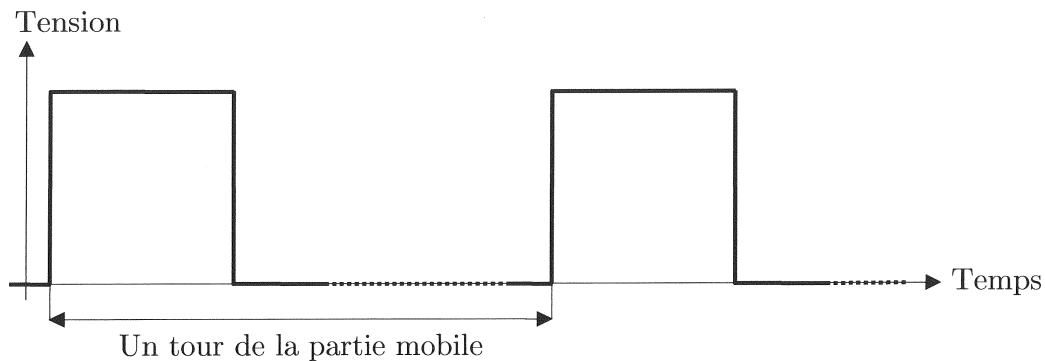


FIGURE 2 – Schématisation du signal en sortie du détecteur

pratique, le signal obtenu peut être affecté par divers phénomènes indésirables (rebond des interrupteurs à lame souple, saturation progressive des phototransistors) et il est donc nécessaire de mettre préalablement en forme les impulsions pour éviter les erreurs de comptage.

2. Travail demandé

Ce sujet comporte deux parties.

La première partie (durée conseillée : 1h30) porte sur l'étude d'un montage électronique, utilisant un amplificateur opérationnel, destiné à mettre en forme les impulsions avant comptage.

La seconde partie (durée conseillée : 2h30) porte sur le stockage des impulsions dans le microcontrôleur, leur traitement informatique (sur un ordinateur classique) pour reconstruire les débits et volumes consommés, la transmission cryptée des mesures vers un serveur distant mettant en oeuvre une base de données, et l'exploitation de celle-ci afin d'accéder à distance au suivi de la consommation.

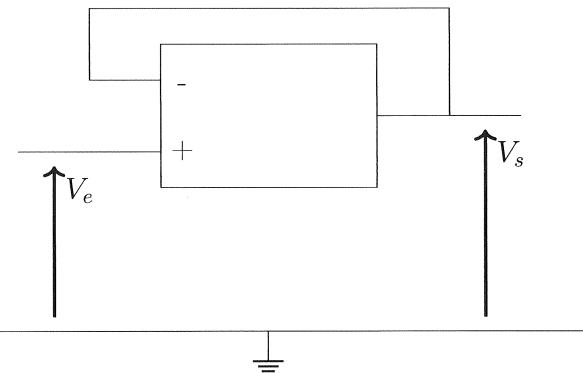
Ces deux parties sont indépendantes.

PREMIERE PARTIE
COMPTEUR D'IMPULSIONS
ANALOGIQUE

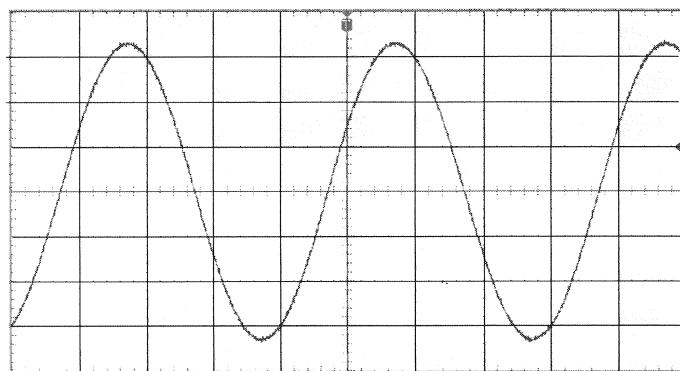
A. GENERALITES SUR LES ALI (15%)

Dans cette partie, on considère un ALI alimenté en +15/ - 15 Volts par une alimentation à point milieu. On admettra que les tensions de saturation haute et basse sont +/ - 15 Volts.

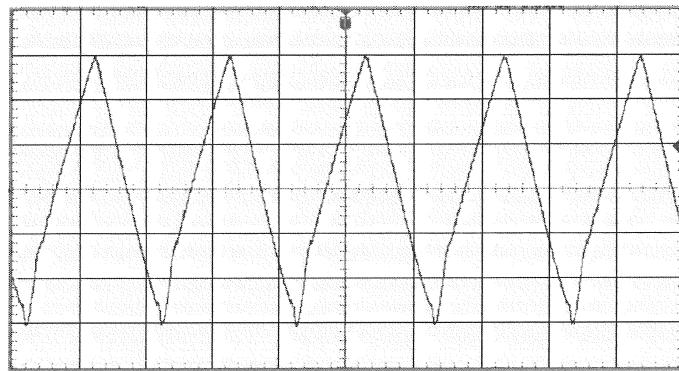
1. Représenter la tension de sortie en fonction de la tension différentielle d'entrée, en indiquant clairement les ordres de grandeur considérés (on indiquera la partie correspondant au régime linéaire et celle correspondant au régime saturé).
2. On s'intéresse au montage représenté ci-dessous. Montrer que $V_s = V_e$. Comment s'appelle ce montage ? Quel est son intérêt ? (on considérera le gain de l'ALI comme infini)



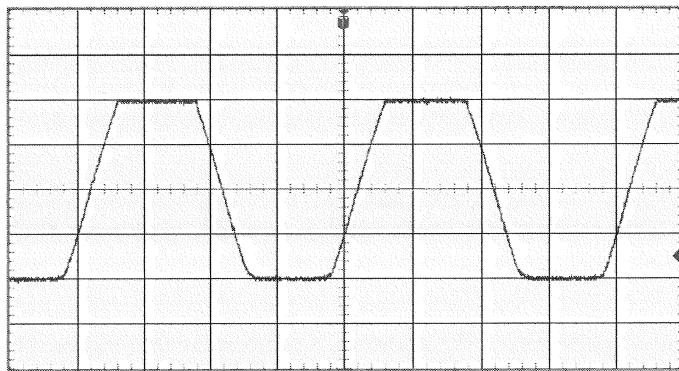
3. On alimente ce montage avec en entrée la tension dont l'oscillogramme est donné ci-dessous. Les réglages sont $2V/div$ et $100\mu s/div$, quelles sont les caractéristiques de cette tension ? Peut-on raisonnablement penser observer la même chose en sortie ?



4. Toutes choses égales par ailleurs, on augmente la fréquence et on observe en sortie la tension ci-dessous. Les réglages sont $2V/div$ et $1\mu s/div$. Quelle caractéristique de l'ALI est ainsi mise en évidence ? Evaluer sa valeur numérique.



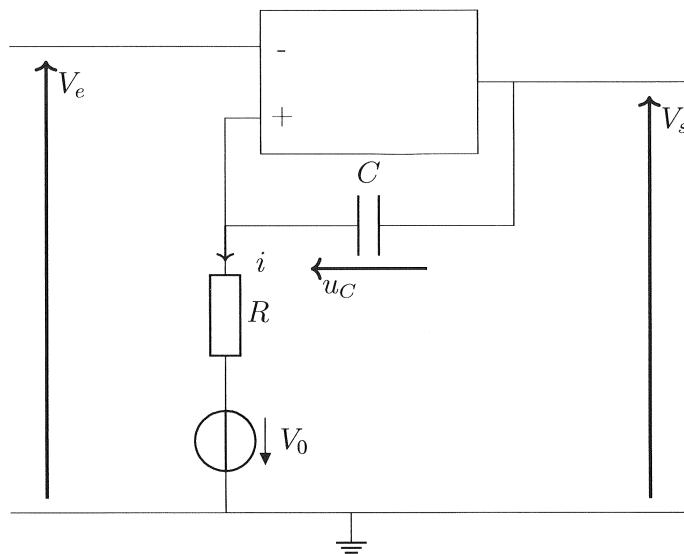
5. On revient à la fréquence de la question 3, et on ajoute une résistance de charge $R_0 = 50\Omega$ entre la sortie et la masse. Les réglages sont $2V/div$ et $100\mu s/div$. Quelle caractéristique de l'ALI est ainsi mise en évidence ? Evaluer sa valeur numérique.



6. Donner le schéma d'un montage amplificateur non inverseur utilisant un ALI et 2 résistances.
Etablir l'expression du gain de ce montage.
7. Proposer des valeurs pour les résistances pour avoir des gains de 10, 100 et 1000. Jusqu'à quelle valeur de gain peut-on aller avec un tel montage avec une tension d'entrée continue ? Avec une tension d'entrée sinusoïdale de fréquence $10kHz$? (de simples ordres de grandeur sont attendus)
8. On alimente ce montage, en prenant un gain de 10, avec la tension d'entrée de la question 3. Dessiner l'allure de la tension attendue en sortie.

B. COMPTEUR D'IMPULSIONS (25%)

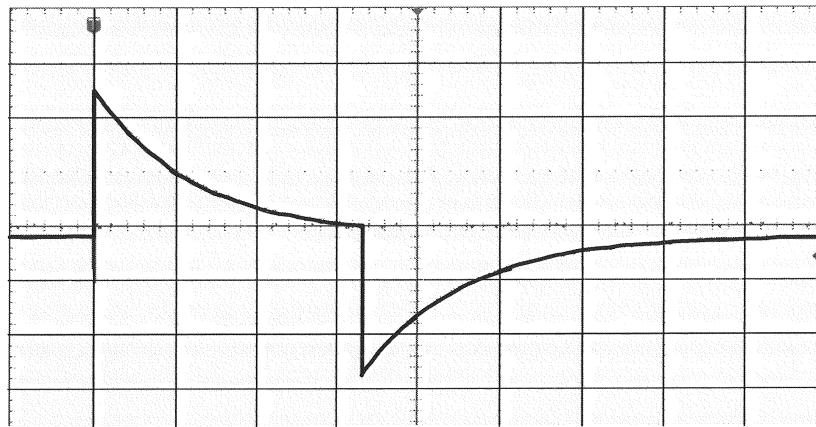
Le montage ci-dessous permet de réaliser un compteur d'impulsions analogique. L'ALI est alimenté en $+V_{cc}$ / $-V_{cc}$ avec $V_{cc} = 7$ Volts par une alimentation à point milieu. Dans toute cette partie, il fonctionne en régime saturé et les tensions de saturation $+V_{sat}$ et $-V_{sat}$ sont considérées comme égales aux tensions d'alimentation $+V_{cc}$ et $-V_{cc}$. On considérera que le temps de réponse de l'ALI est négligeable (on bascule de $+/-V_{sat}$ à son opposé de manière instantanée). On prend pour ce montage $R = 10\text{k}\Omega$, $C = 650\text{nF}$ et $V_0 = 1\text{V}$.



9. Quelle est la valeur de i en régime stationnaire ? De V_+ ? Justifier le fait que $V_s = -V_{sat}$ en régime stationnaire si $V_e = 0$. Quelle est alors la valeur de u_C ?
10. A $t = 0$, on envoie en entrée une impulsion très brève de durée Δt : V_e passe instantanément de 0 à -5 volts puis, un temps Δt plus tard, repasse à 0 (toujours instantanément). Représenter cette impulsion.
11. Que signifie très brève pour Δt ? (Avec quelle grandeur caractéristique du circuit faut-il comparer). En déduire une condition sur Δt . Expliquer pourquoi la valeur de u_C ne varie quasiment pas entre $t = 0$ et $t = \Delta t$.
12. Expliquer pourquoi le passage de V_e de 0 à -5 volts à $t = 0$ fait basculer la sortie à $+V_{sat}$, et pourquoi le retour à 0 à $t = \Delta t$ ne provoque pas un autre basculement.
13. Montrer que, suite au basculement (on prend $t = 0$ au moment du basculement), u_C évolue de la manière suivante :

$$u_C = 2V_{sat} e^{(-t/\tau)} - (V_{sat} + V_0)$$
- Donner l'expression de τ ainsi que son sens physique.
14. A quel instant t_1 la sortie va-t-elle repasser en saturation basse ? Donner l'expression en fonction de R , C , V_{sat} et V_0 et faire l'application numérique. On donne $\ln(14) \simeq 2,6$.

On donne ci-dessous un enregistrement de la tension V_+ suite à une impulsion. L'instant $t = 0$ est décalé à une division après la gauche de l'écran. Les réglages sont $5V/div$ et $5ms/div$.



15. Expliquer les deux phases observées dans l'évolution de cette tension (expliquer en particulier la valeur minimale prise par V_+).
16. Sur la même échelle de temps que l'enregistrement précédent, représenter l'évolution des tensions u_C et V_s .

Le montage reçoit en entrée des impulsions périodiques, toujours de largeur Δt et avec une période T , dans le but de pouvoir mesurer la fréquence f de ces impulsions.

17. Quelle condition doit respecter T vis-à-vis des caractéristiques du montage ?
18. Représenter sur un même graphe, sans échelle mais en respectant les conditions des questions 11 et 17, les tensions V_e et V_s (sur au moins une période T et au plus deux) . Indiquer clairement les trois temps caractéristiques.
19. Donner l'expression de la valeur moyenne $\langle V_s \rangle$ de V_s en fonction de V_{sat} , f et t_1 .
20. Quel type de filtre pourrait-on utiliser en aval du montage précédent pour obtenir $\langle V_s \rangle$? Préciser comment il doit être branché (faire un schéma) et proposer des valeurs pour ses composants si $f = 10 \text{ Hz}$.
21. On souhaite obtenir avec un voltmètre une tension directement proportionnelle à f . Expliquer comment compléter le montage en aval du filtre pour obtenir ce résultat.

SECONDE PARTIE EXPLOITATION INFORMATIQUE DES MESURES

Dans cette partie, les fonctions et programmes attendus devront être écrits en langage Python (sauf, bien entendu, lorsqu'une requête SQL est demandée). On prendra garde à la lisibilité du code, et notamment aux indentations. Le code devra être documenté. Toute fonction définie dans le sujet pourra être utilisée, même sans avoir été codée.

C. ESTIMATION DES DEBITS ET VOLUMES (30%)

La sortie du dispositif précédent est raccordée à une entrée logique d'un microcontrôleur. À chaque fois qu'une impulsion se produit, le microcontrôleur mémorise l'instant auquel l'impulsion a été détectée. Cette information est ensuite utilisée pour reconstituer l'évolution temporelle du volume d'eau et du débit consommés. On se propose ici d'implanter partiellement cette reconstitution.

C.1. Justification du format des données

Si l'on fait l'hypothèse que le débit volumique Q traversant le compteur est constant dans l'intervalle de temps séparant deux impulsions, alors on l'obtient par la relation :

$$Q = \frac{c}{\tau} \quad (1)$$

où $c = 7 \times 10^{-5} \text{ m}^3$ est le volume d'eau consommé à chaque impulsion, et τ la durée séparant deux impulsions. Le débit maximal pouvant traverser le compteur ne dépasse pas $Q_{\max} = 2,5 \text{ m}^3 \cdot \text{h}^{-1}$.

22. Estimer la plus petite valeur numérique que prend τ .

Le microcontrôleur effectue le chronométrage et le stockage en mémoire des instants avec une résolution limitée. On suppose donc que la durée τ est connue avec une incertitude $\delta\tau$. Par suite, le débit Q est déterminé avec une incertitude δQ . On souhaite que l'incertitude relative $\delta Q/Q$ soit d'au plus 1% en valeur absolue (toujours sous l'hypothèse simplificatrice d'un débit constant).

23. Exprimer, en valeur absolue, $\delta Q/Q$ en fonction de $\delta\tau/\tau$. Estimer alors la valeur maximale que doit avoir $\delta\tau$ pour que la condition ci-dessus soit satisfaite au débit Q_{\max} .

Les instants des impulsions sont représentés au format *timestamp*, défini comme le nombre de secondes écoulées depuis le 1^{er} janvier 1970 à minuit. Par exemple, lors d'une acquisition effectuée début 2021, des impulsions ont été détectées aux *timestamp* suivants, exprimés en secondes :

1609825201,145 1609825202,948 1609825206,532 ...

Le microcontrôleur utilisé pour l'acquisition stocke les nombres sur au plus 32 bits. Il propose des types entier et flottant définis comme suit :

- les entiers peuvent prendre des valeurs comprises entre -2^{31} et $2^{31} - 1$;
- les flottants sont au format simple précision de la norme IEEE 754, qui prévoit une mantisse codée sur 23 bits et un exposant codé sur 8 bits.

Quelle que soit la valeur obtenue précédemment, on admet que les instants des impulsions doivent être mémorisés à 1 ms près. Pour les applications numériques, on pourra considérer que $2^{10} \approx 1000$.

- 24.** Justifier qu'il n'est pas possible de stocker directement les instants des impulsions dans le microcontrôleur avec la résolution demandée, que ce soit avec le stockage entier (en exprimant les instants en millisecondes) ou avec le stockage flottant.

C.2. Récupération des données

Compte tenu de la limitation mise en évidence dans la partie précédente, on décide de représenter les instants des impulsions de la façon suivante :

- on synchronise le microcontrôleur à l'aide d'une horloge extérieure : on repère un instant où le *timestamp* est un nombre *entier de secondes*, et on relève la valeur entière correspondante ;
- on relève ensuite, à chaque impulsion, le nombre entier de *millisecondes* écoulées depuis la synchronisation.

Ces valeurs sont stockées en mémoire et, à intervalles de temps réguliers, transmises à un ordinateur où elles sont écrites dans un fichier texte. Par exemple, les *timestamp* donnés dans la partie précédente :

1609825201, 145 1609825202, 948 1609825206, 532 ...

pourraient être représentés par le fichier texte suivant :

```
1609825200
1145
2948
6532
...
```

On souhaite, à partir de ce fichier, construire en Python une liste de flottants contenant les *timestamp*. Supposons par exemple que le fichier précédent s'appelle `log.txt` et se limite aux quatre premières lignes ci-dessus. On exécute alors les instructions suivantes, dont on donne le résultat affiché dans l'interpréteur :

```
>>> f = open("log.txt", "r")
>>> lignes = f.readlines()
```

```
>>> f.close()
>>> print(lignes)
['1620115200\n', '1145\n', '2948\n', '6532\n']
```

On précise que la séquence de caractères \n indique le passage à la ligne et ne perturbe pas les fonctions de conversion. On précise également que le type flottant de Python est au format double précision sur la plupart des ordinateurs, ce qui suffit largement à stocker les instants des impulsions avec la résolution exigée.

- 25.** À l'aide de l'exemple ci-dessus, écrire une fonction `recup(nomFichier)` qui prend pour argument une chaîne de caractères `nomFichier` donnant le nom du fichier texte contenant les instants stockés au format décrit dans cette partie, et renvoie une liste de flottants contenant les instants des impulsions exprimés en seconde.

C.3. Première reconstruction : débit constant par morceaux

On suppose dans cette partie que l'on dispose de la liste des instants des impulsions, notés T_i pour $0 \leq i \leq n$ (il y en a donc $n + 1$). Par construction, cette liste est croissante au sens strict, deux impulsions ne pouvant pas survenir simultanément.

On envisage tout d'abord l'hypothèse d'un débit constant par morceaux. On pose alors, pour tout i tel que $0 \leq i \leq n - 1$:

$$q_i = \frac{c}{T_{i+1} - T_i} \quad (2)$$

où c est le volume consommé entre deux impulsions.

- 26.** Écrire une fonction `débits1(1Imp, c)` qui prend pour arguments la liste `1Imp` des instants des impulsions et un flottant `c`, et renvoie la liste des débits définis par l'équation (2).

À partir de ces deux listes, on souhaite reconstruire l'évolution temporelle du débit volumique consommé sur l'intervalle de temps correspondant à l'enregistrement, de la façon suivante :

$$q(t) = \begin{cases} 0 & \text{si } t < T_0 \text{ ou } t \geq T_{n-1} \\ q_i & \text{si } T_i \leq t < T_{i+1}, 0 \leq i < n - 1 \end{cases} \quad (3)$$

- 27.** Écrire une fonction `indice(1Imp, t)` qui prend pour arguments la liste `1Imp` des instants T_i des impulsions et un flottant `t`, tels que `t` est compris entre le premier et le dernier élément de `1Imp` (premier inclus, dernier exclu), et recherche l'entier i tel que $T_i \leq t < T_{i+1}$. On ne demande pas de vérifier la condition précédente (ce sera fait lors de l'appel de la fonction).

L'évaluation précédente est menée en m instants de calcul utilisés pour le tracé et l'interpolation, stockés dans une liste `1Temps`.

28. Écrire une fonction `temporel(1Q,1Imp,1Temps)` qui prend pour arguments trois listes `1Q` (débits moyens calculés par l'équation (2)), `1Imp` (instants des impulsions) et `1Temps` (instants de calcul), et renvoie la liste des débits moyens aux instants de `1Temps`. Ceux-ci ne sont pas forcément compris entre le premier et le dernier élément de `1Imp`. Lorsqu'ils le sont, on appellera la fonction `indice`.
29. Donner et justifier la complexité asymptotique de la fonction `temporel` en fonction du nombre m d'instants de calcul et du nombre n d'impulsions dans le pire des cas.

En pratique, la liste des impulsions et celle des instants de calcul sont toutes deux triées par ordre croissant. En optimisant le parcours des deux listes, il est possible de réduire la complexité des calculs.

30. Justifier qu'il n'est pas possible de réduire la complexité tout en conservant l'appel à la fonction `indice`. Expliquer, en deux ou trois phrases éventuellement complétées par un schéma, comment passer en revue les éléments des deux listes pour minimiser le nombre d'opérations si l'on suppose que les deux tailles m et n sont voisines. Donner la complexité obtenue pour $m = n$.

En plus du débit volumique, on souhaite pouvoir visualiser le volume consommé, défini comme l'intégrale temporelle du débit volumique. Pour tout j tel que $0 \leq j < m$, on note t_j les instants de calcul, $q_j = q(t_j)$ les débits et $v_j = v(t_j)$ les volumes consommés. Les instants de calcul étant supposés régulièrement espacés, on notera $\Delta t = t_{j+1} - t_j$.

31. Donner la relation de récurrence permettant d'exprimer, par la méthode des trapèzes, v_j en fonction de v_{j-1} , q_j , q_{j-1} et Δt .
32. Écrire une fonction `volumes(1Temps,1QT,v0)` qui prend pour arguments deux liste `1Temps` (instants de calcul) et `1QT` (débits aux instants de `1Temps`) et un flottant `v0`, et renvoie la liste des volumes consommés, calculés par la méthode des trapèzes. Le volume initial (avant le premier instant de `1Temps`) sera pris égal au paramètre `v0`.

On dispose d'un relevé d'impulsions contenu dans un fichier nommé `2021-01-31.txt`. On souhaite faire appel aux fonctions précédentes pour extraire les impulsions stockées dans ce fichier, déterminer les débits et préparer les deux listes "temporelles" nécessaires au tracé.

33. Écrire une suite d'instructions appelant entre autres les fonctions précédentes de sorte à :
- placer le contenu du fichier `2021-01-31.txt` dans une liste `1Imp` ;
 - construire la liste des débits `1Q` associés, en litres par seconde, si à chaque impulsion correspond 0,07 litre ;
 - générer une liste `1Temps` de 1000 flottants régulièrement espacés, allant de la première à la dernière valeur de `1Imp`, tous deux inclus ;
 - construire la liste des débits `1QT` (tirés de `1Q`) aux instants de `1Temps` ;
 - en déduire la liste `1VT` des volumes exprimés en litres, toujours aux instants de `1Temps`, pour un volume initial nul.

A partir des listes venant d'être construites, on peut tracer l'évolution temporelle du débit et du volume, donnée Figure 3.

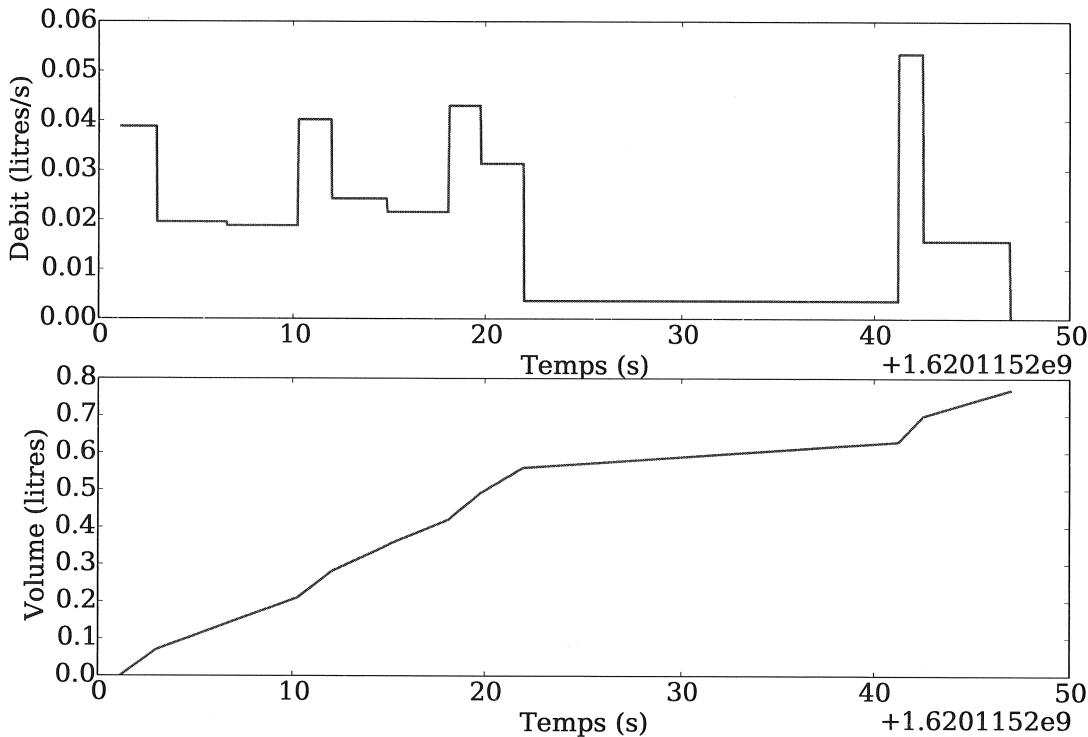


FIGURE 3 – Évolution temporelle du débit et du volume consommé

Une limitation de cet algorithme est qu'il ne permet pas, en l'état, de détecter l'absence de consommation d'eau pendant un laps de temps donné. En effet, un tel cas de figure se manifeste par la présence de deux impulsions très espacées, et l'algorithme estime alors, de façon logique, un débit volumique constant et très faible entre ces deux impulsions. On se propose donc de modifier le mode d'évaluation des débits pour permettre la détection des périodes sans consommation, selon le principe schématisé Figure 4.

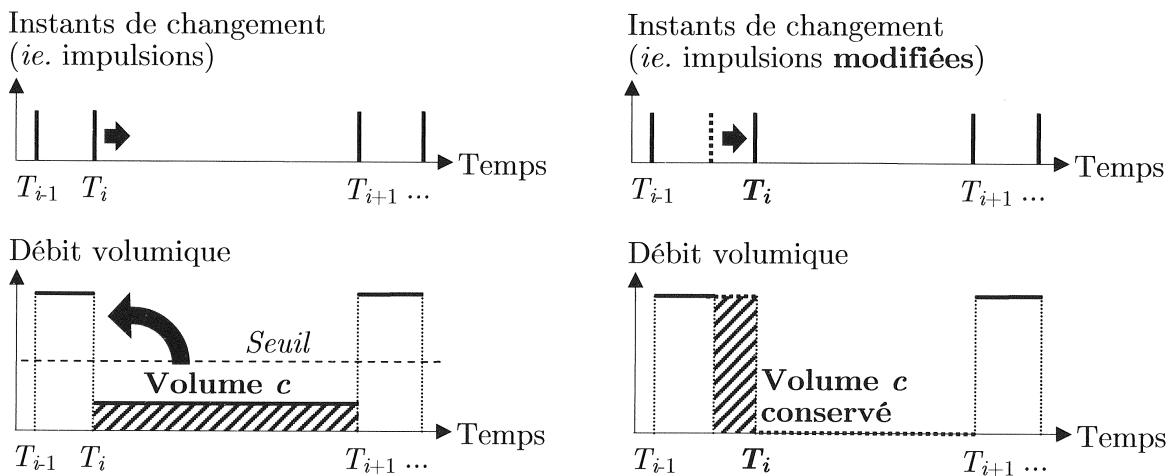


FIGURE 4 – Principe de la détection des périodes sans consommation

Pour cela, on se donne un seuil, ou débit minimal admissible ; le débit q_i calculé par l'équation (2) ne sera retenu que s'il est supérieur à ce seuil. Si le débit q_i est inférieur au seuil, on le met à zéro, et on retardé l'instant T_i de l'impulsion précédente de sorte à conserver le volume total consommé. On remarquera que le volume consommé entre deux impulsions est toujours égal à c .

Enfin, on n'effectuera ce processus que sur *un échantillon à la fois* : si plusieurs débits vérifiant le critère précédent se suivent, on ne décalera que la *première* impulsion intervenant dans le calcul de ces débits, sans modifier les suivantes. Un tel cas de figure signifie en effet qu'il existe soit un faible débit continu, soit une consommation intermittente de faibles volumes ; il n'est pas possible de distinguer ces deux cas en l'état, mais il n'est pas non plus souhaitable de les négliger.

La fonction ne devra pas modifier la liste des impulsions. On rappelle que lorsque l'argument d'une fonction est une liste, alors la fonction peut modifier son argument. Pour éviter cela, il est possible de copier une liste avec une instruction du type `Lcopie = L[:]`.

34. Écrire une fonction `debits2(Limp, c, seuil)` prenant pour arguments la liste `Limp` des instants des impulsions et deux flottants `c` et `seuil`, qui renvoie un tuple contenant :

- la liste des instants des impulsions modifiés comme indiqué ci-dessus (l'argument `Limp` ne doit pas être modifié),
- et la liste des débits calculés comme indiqué ci-dessus.

Comme indiqué précédemment, on ne modifiera pas plus d'une impulsion d'affilée (mais on pourra modifier plusieurs impulsions dans la liste).

Pour l'exemple précédent, le résultat obtenu est donné Figure 5.

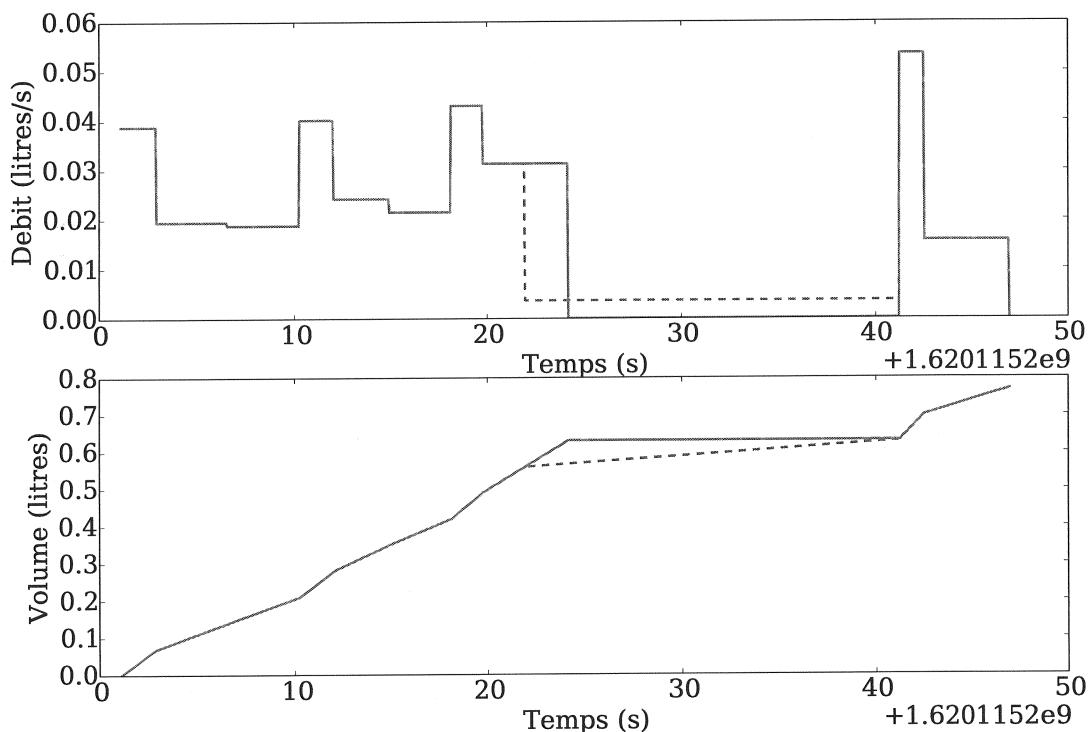


FIGURE 5 – Évolution temporelle avec prise en compte des périodes sans consommation

D. TRANSMISSION DES DONNÉES ET CONSULTATION A DISTANCE (30%)

Les données sur la consommation élaborées dans la partie précédente font ensuite l'objet d'un chiffrement, puis sont transmises à un serveur dans le but de permettre un suivi à distance de la consommation. Ces opérations sont l'objet de cette partie.

D.1. CHIFFREMENT ET DÉCHIFFREMENT DES DONNEES

Les données sur la consommation d'eau ayant un caractère confidentiel, elles font l'objet d'un chiffrement avant d'être transmises au serveur distant. L'algorithme utilisé est le chiffrement XOR, ou "OU exclusif".

On rappelle que le OU exclusif, noté \oplus et représenté par l'instruction \wedge en Python, est un opérateur logique binaire qui renvoie 1 (ou Vrai) si les deux opérandes sont différents, 0 (ou Faux) s'ils sont identiques. Ainsi, $0 \wedge 0$ et $1 \wedge 1$ renvoient tous deux 0, tandis que $1 \wedge 0$ et $0 \wedge 1$ renvoient tous deux 1.

L'opérateur \wedge permet d'appliquer un OU exclusif *bit à bit* entre deux entiers : le OU exclusif est alors appliqué sur les écritures binaires respectives des deux entiers, entre chaque paire de bits de même rang. Considérons par exemple l'instruction $6 \wedge 3$. Si l'on écrit les bits de poids fort en premier (le résultat ne dépend pas de la convention choisie) :

- 6 a pour écriture binaire $(110)_2$;
- 3 a pour écriture binaire $(011)_2$;
- le OU exclusif bit à bit donne alors $1 \wedge 0$ (poids fort) puis $1 \wedge 1$ puis $0 \wedge 1$ (poids faible), soit $(101)_2$, soit 5.

35. Déterminer le résultat renvoyé par l'instruction $13 \wedge 7$.

On considère dans cette partie que les données à transmettre consistent en une liste d'entiers, et on se donne un autre entier nommé *clé*. Le cryptage XOR consiste à appliquer un OU exclusif bit à bit entre les données à transmettre et la clé.

La façon la plus simple de procéder consiste à appliquer un OU exclusif bit à bit entre chacun des entiers à transmettre et la clé.

- 36.** Écrire une fonction `code1(L, cle)` prenant pour arguments une liste L d'entiers et un entier *cle*, et renvoyant la liste des éléments de L chiffrés à l'aide de *cle*.
- 37.** Justifier que, si *a* et *b* sont deux bits valant 0 ou 1, $(a \oplus b) \oplus b = a$ (on utilisera par exemple une table de vérité). Indiquer alors comment décoder une liste codée par l'instruction `Lc = code1(L, cle)` et préciser pourquoi cet algorithme de chiffrement est dit *symétrique*.

En pratique, la fonction `code1` n'est pas un moyen de chiffrement sûr car tous les éléments de la liste sont cryptés de la même façon. S'il existe une ressemblance entre ces éléments (par exemple s'ils sont proches les uns des autres) et si l'on dispose d'informations sur les données, il est

possible d'identifier la clé puis d'en déduire la totalité du message. Pour y remédier, on se propose d'appliquer le chiffrement non pas à chaque élément de la liste (ce que l'on appelle le chiffrement *par blocs*) mais à la totalité des données mises bout à bout, en répétant pour cela la clé autant de fois que nécessaire (ce que l'on appelle le chiffrement *par flux*).

Considérons par exemple la suite d'entiers positifs (12,3,5,8) codés sur 4 bits, avec la clé 2 codée sur 3 bits. Écrivons ces entiers en binaire les uns à la suite des autres, puis la clé (010) répétée tout au long du message (la dernière répétition est ici partielle), et enfin le résultat du OU exclusif bit à bit :

```
1100 0011 0101 1000  
0100 1001 0010 0100  
1000 1010 0111 1100
```

Enfin, on interprète le résultat binaire final comme une suite d'entiers codés sur 4 bits (comme la liste de départ). On obtient donc (8,10,7,12).

Dans l'exemple ci-dessus, la clé étant écrite sur 3 bits et les données sur 4 bits, chaque entier est codé de façon différente. Ce principe permet d'obtenir un chiffrage relativement sûr si les nombres de bits utilisés sont premiers entre eux et suffisamment élevés.

On souhaite programmer cet algorithme pour transmettre des entiers positifs codés sur 64 bits. Les représentations binaires seront codées sous la forme de chaînes de caractères contenant des 0 et des 1. On rappelle que :

- la fonction `bin` permet de convertir un entier en binaire : `bin(10)` renvoie '`'0b1010'`', le préfixe '`'0b'`' indiquant une écriture binaire. On remarquera que l'écriture utilise le nombre minimal possible de bits, de sorte que le bit de poids fort est toujours 1 ;
- la fonction `int` permet de convertir une représentation binaire en entier, si l'on indique la base de numération (ici 2) en second argument : `int('0b1010',2)` et `int('1010',2)` renvoient tous deux 10.

On suppose que tous les entiers dont il est question dans cette partie sont positifs et représentables sur 64 bits.

38. Écrire une fonction `bin64(n)` prenant pour argument un entier positif `n`, et renvoyant une chaîne de caractères contenant son écriture binaire sur 64 bits (c'est-à-dire complétée par autant de zéros que nécessaire), sans le préfixe '`'0b'`'. On utilisera la fonction `bin`.
39. Écrire une fonction `bits(L)` prenant pour argument une liste `L` d'entiers positifs, et renvoyant une chaîne de caractères constituée des écritures binaires des éléments de `L` sur 64 bits, concaténées les unes aux autres. On utilisera la fonction `bin64`.
40. Écrire une fonction `entiers(flux)` prenant pour argument une chaîne `flux` au format précédent (constituée des caractères '`'0'`' et '`'1'`' et de longueur multiple de 64), et renvoyant la liste des entiers dont les codages respectifs sur 64 bits se trouvent bout à bout dans la chaîne `flux`. On utilisera la fonction `int`.

- 41.** Si l'on appelle l la longueur de la clé, quelle est la position du bit de la clé (entre 0 et $l - 1$) correspondant au bit de rang i (compté à partir de zéro) de la liste à chiffrer ? On donnera la réponse sous la forme d'une opération arithmétique simple.

Pour que l'algorithme précédent réalise un chiffrement sûr, on souhaite que le nombre de bits sur lequel est écrit la clé soit premier avec 64, c'est-à-dire impair.

- 42.** Écrire une fonction `code2(L, cle)` prenant pour arguments une liste `L` d'entiers et un entier `cle`, et renvoyant une liste contenant le résultat du chiffrement par flux de `L` à l'aide de `cle` selon l'algorithme décrit ci-dessus. L'entier `cle` sera écrit en binaire sur un nombre de bits impair (on ajoutera donc un 0 en tête de son écriture binaire si cela est nécessaire). On utilisera entre autres les fonctions `bits` et `entiers`.

Une limitation importante du chiffrement par OU exclusif est que si l'on dispose d'un message *et* de sa version chiffrée, alors il est très facile de retrouver la clé de chiffrement. Pour cette raison, sur les applications présentant un risque particulier (ce qui n'est pas le cas ici), la clé de chiffrement est à *usage unique* (l'algorithme est alors dit à *masque jetable*).

- 43.** À partir des résultats précédents et en vous aidant notamment de la réponse à la question 37, expliquer simplement comment il est possible de reconstruire la clé si l'on dispose d'un message et de son chiffrement par la fonction `code1`. Faire de même pour la fonction `code2`.

D.2. CONSULTATION DES DONNÉES

Un particulier souhaite suivre précisément sa consommation d'eau et enregistre à cet effet chaque jour la quantité consommée. Les données ainsi recueillies sont enregistrées dans une base de données. On donne ci-dessous un extrait de la table `eau` utilisée.

Table <code>eau</code> (extrait)		
<code>id</code>	<code>date</code>	<code>conso</code>
...
36	2017-11-11	497
37	2017-11-12	156
38	2017-11-13	392
39	2017-11-14	274
40	2017-11-15	343
41	2017-11-16	296
42	2017-11-17	153
...

Cette table contient :

- un identifiant associé à chaque mesure, de type entier,
- la date, au format `aaaa-mm-jj`, de type chaîne de caractères,
- la consommation (en litres), de type flottant.

- 44.** Écrire en SQL les requêtes permettant d'obtenir :
1. la consommation le 17 novembre 2017,
 2. les dates des consommations supérieures à 500 litres.
- 45.** L'opérateur LIKE permet de faire des comparaisons partielles entre chaînes de caractères, et en particulier de remplacer une partie d'une chaîne par un %, la comparaison étant vraie quel que soit la partie de chaîne comparée avec le % (par exemple 'informatique' LIKE 'info%' est vrai). Écrire les requêtes permettant d'obtenir :
1. les consommations de tous les jours du mois de novembre 2017,
 2. les consommations totale et moyenne du mois de novembre 2017.
- 46.** On envisage maintenant de gérer les consommations de tout un ensemble d'habitations (par exemple, la mairie enregistre les informations de chaque logement de la commune). Expliquer comment organiser les données en utilisant, en plus de la table **eau**, une table **habitations** dont on précisera les attributs les plus importants. Quel attribut faudra-t-il ajouter à la table **eau**? Expliquer soigneusement quelle sera la condition de jointure utilisée pour faire des requêtes sur ces deux tables.

Fin de l'épreuve

