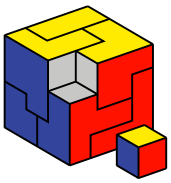
Prologin 2021



Concours national d'informatique

Épreuve écrite d'algorithmique
En ligne 2

Dimanche 27 février 2021

Déco Dico

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, une épreuve de programmation sur machine (3 heures 30).

Conseils

— Lisez bien tout le sujet avant de commencer.
— **Soignez la présentation** de votre copie.
— N'hésitez pas à poser des questions.
— Si vous avez fini en avance, relisez bien, ou préparez votre présentation pour l'entretien.
— N'oubliez pas de passer une bonne journée.

Remarques

— Le barème est donné à titre indicatif uniquement.
— Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
— Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
— Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe, sinon ça va barder.
— Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
— Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 Exercice 1 : Un peu d'ordre

Joseph Marchand, archéologue de profession, a découvert au cours de ses recherches deux livres anciens. Il s'avère rapidement que le premier ouvrage est un dictionnaire d'un autre temps, porteur de mots inconnus. Les caractères même de ces mots sont étrangers à Joseph.

Curieux, il décide de chercher l'ordre alphabétique de cette langue perdue.

Les questions qui suivent vous permettrons d'aider Joseph à résoudre son problème.

2.1 A, B, D

Question 1 (1 point)

> Ranger les mots suivants d'après l'alphabet utilisé en français. Accents et majuscules n'ont pas d'importance.
>
> ["A", "Prologin,", "une", "crêpe", "sera", "toujours", "apportée", "à", "ceux", "qui", "en", "ont", "besoin"]

Question 2 (1 point)

> Écrire une fonction $sort\_words(words)$, prenant comme argument une liste de mots $words$. La fonction devra trier les mots de $words$ dans l'ordre lexicographique [1] (sans utiliser les fonctions de tri de votre language).

Question 3 (1 point)

> Ranger les mots suivants dans l'ordre défini par leur alphabet.
>
> 1. $"aaa","bbb","ccc","aba","acc","bac" -> ['a','c','b']$,
> 2. $"x8q","841","wqw","xxx","118","4x4" -> ['8','x','w','1','4','q']$

Question 4 (1 point)

> Ecrire une fonction $word\_diff(a, b)$ qui renvoie pour deux mots a et b, le premier couple de lettres qui diffèrent. Par exemple :
>
> — word_diff("graphe", "greffe") renverra ('a', 'e'),
> — word_diff("chat", "chats") renverra (", 's')
> — word_diff("trier", "trier") renverra (", '')

Question 5 (1 point)

> Écrire une fonction $sort\_words\_from\_alphabet(words, alpha)$, prenant comme arguments une liste de mots $words$ et une liste de caractères indiquant l'ordre alphabétique $alpha$ (sans utiliser les fonctions de tri de votre language).
>
> Par exemple, pour notre alphabet lexicographique, la fonction prendrait comme deuxième argument [a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]

Question 6 (1 point)

> Trouver un alphabet valide [2] pour le dictionnaire ["ABC", "BCB", "BCDE", "EF", "CCE", "CF"]

Question 7 (1 point)

> Peut-il exister plusieurs alphabets pour un dictionnaire ?

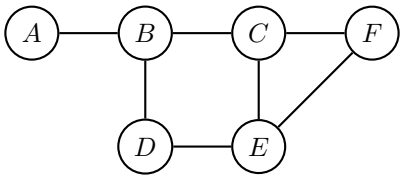
1. A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
2. Qui conserve l'ordre du dictionnaire

Question 8 (2 points)

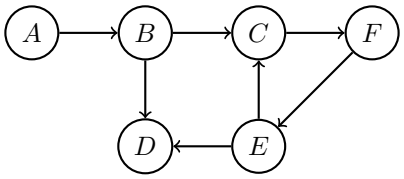
> Prouver votre réponse à la question précédente.

2.2 Un mot sur les DAG

Un graphe est une structure de données permettant de représenter des relations dans un ensemble d'éléments. Un graphe est composé de sommets qui représentent les éléments, et d'arêtes qui représentent les relations entre les différents éléments.

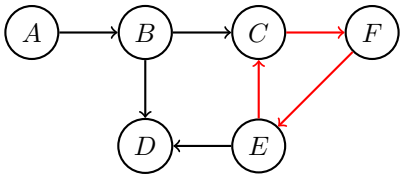


Un graphe est orienté si ses arêtes sont fléchées.



Le graphe est orienté acyclique s'il ne possède pas de circuit. Un circuit dans un graphe orienté est une suite d'arcs consécutifs (chemin) dont les deux sommets extrémités sont identiques.

Dans notre exemple précédent, le graphe possède un cycle en CFE.



Question 9 (2 points)

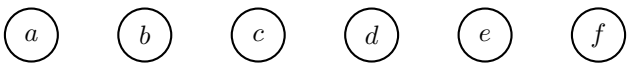
> Proposer une structure de données pour exprimer un graphe orienté.

Question 10 (2 points)

> Créer un graphe des différences. Vous pouvez utiliser des fonctions créées précédemment dans le sujet.

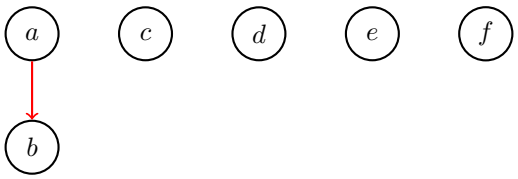
L'idée est d'itérer à travers le dictionnaire et de comparer les mots adjacents pour obtenir une différence de caractère. On souhaite construire un graphe des précédences de lettres. Pour cela, on va souhaiter comparer tous les mots à leur suivant et enregistrer les premières différences dans le graphe.

On crée un graphe composé uniquement de sommets, les lettres. Par exemple, reprenons {"ABC", "BCB", "BCDE", "EF", "CCE", "CF"}, le dictionnaire vu précédemment.

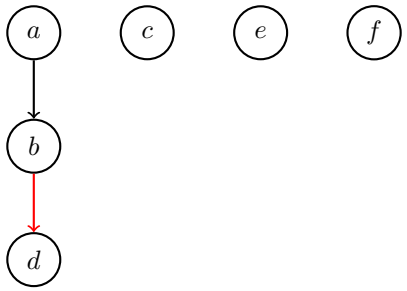


S'il y a une différence, on insère la paire de lettres qui diffèrent dans le graphe sous la forme d'un arc.

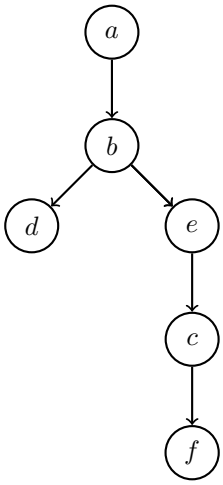
Dans notre exemple, la première différence de "ABC" et "BCB" est ("A", "B"), on exprime donc cette différence dans le graphe en ajoutant un arc de A vers B.



On compare les mots suivants, “BCB” et “BCDE” : la différence est (“B”, “D”), qu'on ajoute au graphe.

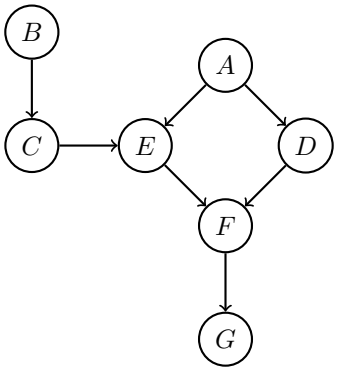


Après avoir répété cela pour tous les mots, notre graphe est forcément acyclique puisque les mots suivent un ordre alphabétique, une lettre ne pouvant pas précéder et succéder une autre simultanément. Le fléchage en fait par conséquent un graphe acyclique orienté.

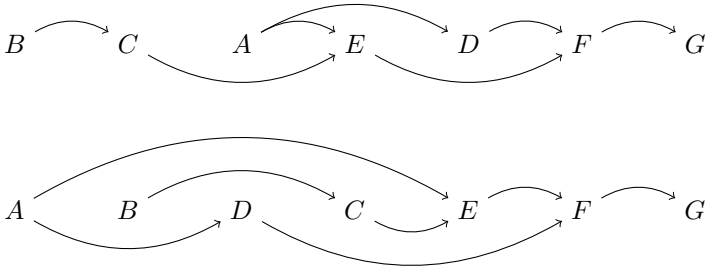


2.3 Le tri topologique

Un tri topologique d'un graphe orienté acyclique G=(S,A) est un ordre linéaire des sommets de G tel que si G contient l'arc (u,v), u apparaît avant v. Le tri topologique d'un graphe peut être vu comme un alignement de ses sommets le long d'une ligne horizontale, tel que tous les arcs soient orientés de gauche à droite. Attention ! Le tri topologique d'un graphe orienté acyclique n'est pas forcément unique.

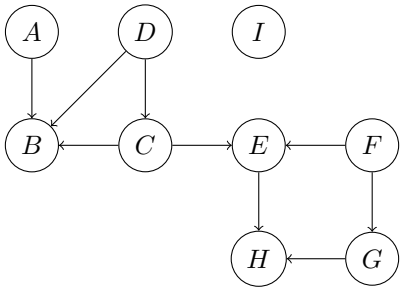


Voici par exemple 2 tris topologiques pour le graphe ci-dessus :



Question 11 (2 points)

> Proposer un tri topologique du graphe ci-dessous.



Question 12 (2 points)

> Pourquoi le graphe doit-il forcément être acyclique pour trouver un tri topologique ?

Question 13 (3 points)

> Proposer un algorithme de tri topologique. Quelle est sa complexité au mieux ?

Question 14 (5 points)

> Implémenter un algorithme qui résoud le problème principal : à partir d'un dictionnaire de mots, déterminer un alphabet.

3 Exercice 2 : Un autre Trie

Après avoir retrouvé l'ordre de l'alphabet de cette langue, Joseph décide d'ouvrir le second livre, probablement une compilation de recettes de cuisine. Joseph remarque que certains mots reviennent souvent, indiquant surement des aliments communs. Il décide de comparer le nombre d'occurences de chaque aliment par rapport à un livre de cuisine moderne, pour tenter d'identifier des équivalences.

Il doit donc premièrement trouver le nombre d'occurences des différents mots. Joseph étant affamé, sa solution de recherche doit être efficace dans le temps.

Les questions qui suivent vous permettrons d'aider Joseph à résoudre son problème.

3.1 String matching

Dans un premier temps, nous souhaitons trouver pour une chaîne de caractères S si un pattern [3] P non nul est inclus dans S. Un pattern est ici une suite de caractères du même alphabet que S.

Par exemple, dans la chaîne "MAPEEJNDMQ DUUPEDME PEQW", le pattern "PE" est répété 3 fois, dans la chaîne "BLABLABLEBLIBLOBLOBLABLABLABLU", le pattern "BLABLA" est aussi répété 3 fois (attention aux superpositions !)

Question 15 (1 point)

> Proposer une manière naïve d'afficher toutes les occurences de P dans S.

Question 16 (1 point)

> Vérifier un pattern c'est bien, mais en vérifier plusieurs à la fois, c'est mieux.
> Adapter la proposition précédente pour prendre en compte plusieurs patterns.

3. Motif

Question 17 (1 point)

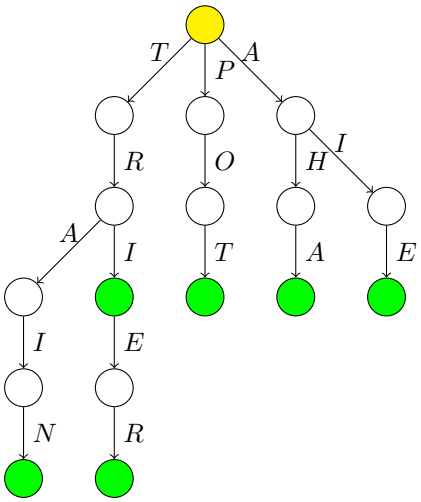
> Discuter de la complexité de votre réponse précédente.

3.2 Un nouvel arbre

Avant de poursuivre, introduisons une nouvelle structure, le trie (prononcer *traille*).

Un trie est un arbre pour lequel chaque noeud possède un indicateur booléen et jusqu'à $|Z|$ [4] enfants, avec Z notre alphabet. En effet, les trie servent à stocker un ensemble de mots, en épellant à chaque sommet une nouvelle lettre du mot. L'indicateur d'un noeud permet de savoir si la suite de lettres de la racine à ce noeud forme un mot.

Par exemple, le trie pour le dictionnaire ["TRAIN", "TRI", "AIE", "POT", "AHA", "TRIER"] est :



La racine est représentée en jaune, les noeuds qui indiquent la fin d'un mot en vert. Chaque noeud représente une chaîne de caractères donnée par le chemin de la racine jusqu'à ce noeud. Par exemple, le noeud vert en bas à gauche représente la chaîne de caractères "TRAIN".

Question 18 (1 point)

> Proposer une structure de données pour représenter un trie.

Question 19 (1 point)

> Il est souhaité de vérifier si le mot M appartient à un trie, quelle est la complexité de la recherche ?

Question 20 (2 points)

> Décrire une procédure efficace [5] d'insertion dans un trie.

Question 21 (2 points)

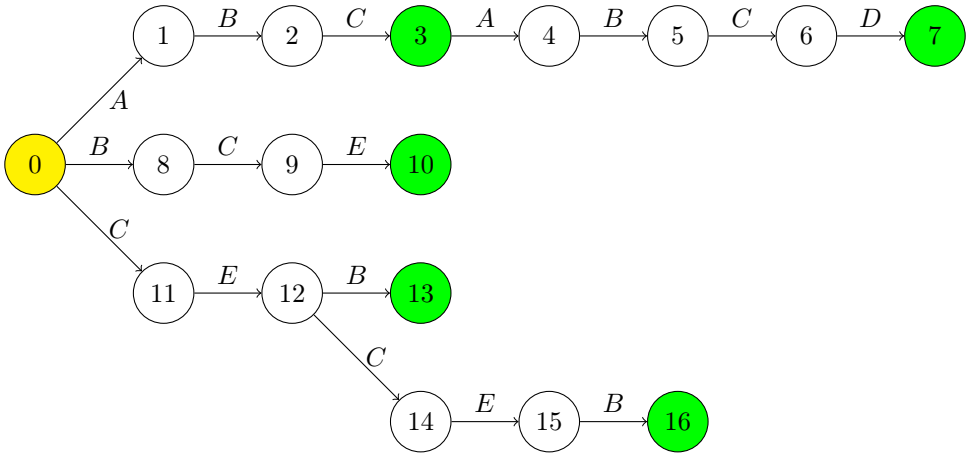
> Décrire une procédure efficace [6] de suppression dans un trie.

4. L'opérateur | | permet de récupérer la taille d'un objet. Pour $x =$ "*Prologin*", $|x| = 8$
5. Mesurée par la complexité et l'espace mémoire nécessaire
6. Même chose

3.3 Matcher par trie

Maintenant que nous maitrisons les trie, utilisons les pour stocker tous les patterns à trouver dans notre texte.

Prenons par exemple les patterns ["ABCABCD", "BCE", "CEB", "CECEB", "ABC"]



Ici, les noeuds ont des numéros uniquement pour les identifier dans les explications. En réalité, ils ne contiennent pas de valeur numérique.

Si on souhaite valider la chaîne de caractères "ABCEB" avec les patterns du trie ci-dessus :
— Avec A, on atteint le noeud 1 en débutant à 0,
— Avec B, on atteint le noeud 2, mais aussi le 8 en repartant de 0,
— Avec C, on atteint le noeud 3 (qui valide le pattern "ABC"), 9 et 11,
— Avec E, le noeud 3 n'a pas d'enfant en E, et on atteint le noeud 10 (qui valide le pattern "BCE") et 12,
— Avec B, le noeud 10 n'a pas d'enfant en B, et on atteint le noeud 13 (qui valide le pattern "CEB").

Question 22 (2 points)

Proposer une nouvelle réponse à la question 16, en utilisant cette fois un simple trie.

Question 23 (4 points)

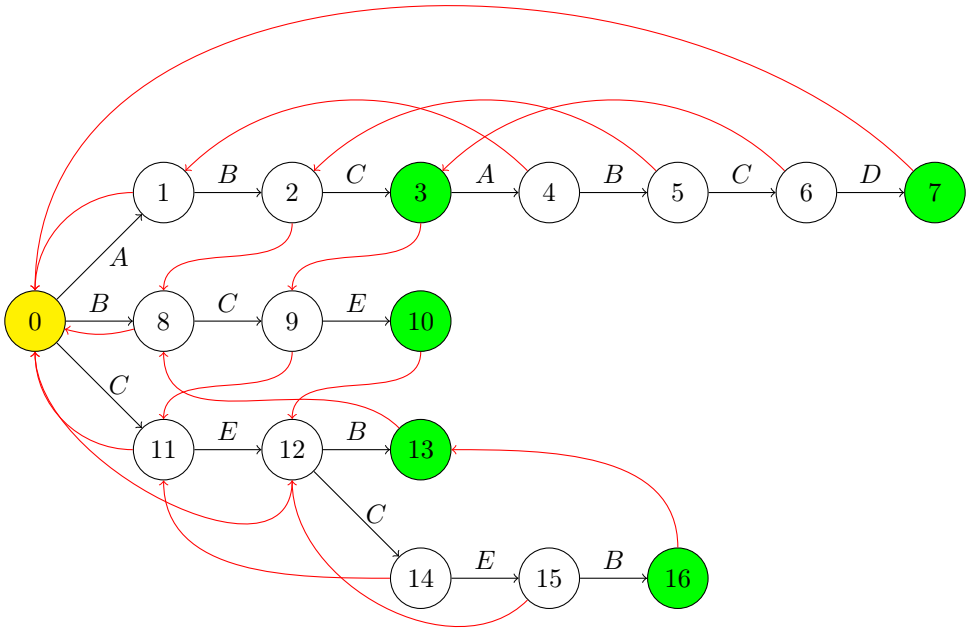
Proposer un algorithme qui construit les liens suffixes sur tous les noeuds (en dehors de la racine) d'un trie. Modifier si nécessaire votre structure de trie. La solution attendue est en O(N), N le nombre de noeud dans le tri.

Avec N une node représentant le mot w, et x le plus long suffixe de w appartenant au trie, un **lien de suffixe** dans un trie est un pointeur de N au noeud correspondant au plus long suffixe de w appartenant au trie, c'est à dire au mot x.

Le **suffixe** d'un mot w représente entre 0 et $|w|$ caractères qui composent la fin de w.

Pour le lien suffixe du mot "ABCABC" :
— Les suffixes de "ABCABC" sont ["", "C", "BC", "ABC", "CABC", "BCABC", "ABCABC"],
— Parmi eux, seuls ["", "C", "BC", "ABC", "ABCABC"] appartiennent au tri,
— Le suffixe "ABCABC" est exclu puisqu'un lien suffixe vers soit-même n'a pas de sens,
— Le plus long suffixe restant est donc "ABC"
Ainsi, dans notre graphe, le plus long suffixe du noeud 6 est le noeud 3.

Ci-dessous l'ensemble des liens suffixes de notre exemple.



Question 24 (2 points)

> L'algorithme ci-dessous à vocation à répondre au problème principal. Toutefois, il ne permet pas vraiment de prendre en compte tous les patterns qui apparaissent dans le texte. Dans quel cas un pattern sera ignoré ?

Avec *patterns* la liste des n patterns, S la chaîne de caractères. Considérer l'algorithme suivant :

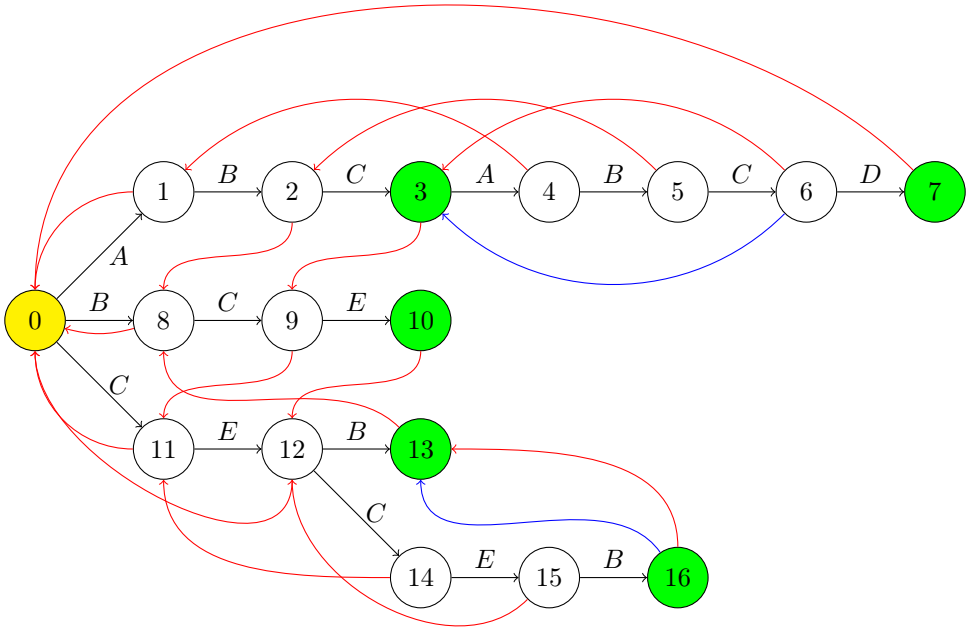
```
# Pre-calculations
trie = Trie(patterns)
trie.construire_lien_suffix()
state = trie.racine

# Run
Pour i = 0, tant que i < |S|:
    Tant que state != trie.racine et S[i] n'appartient pas a state.children:
        state = state.lien_suffix
    Si S[i] appartient a state.children:
        state = state.children[S[i]]
    Si state est un mot:
        Afficher le mot
```

Question 25 (4 points)

> Proposer un moyen de résoudre ce problème en utilisant ce qui a déja été développé. Ci-dessous quelques pistes.

Le graphe précédent avec de mystérieux nouveaux liens :



Quelques remarques utiles :
— Si a est une sub-string de b, alors a est un suffixe d'un prefix de b.
— Avec P_1 et P_2 des patterns tel que $|P_1| < |P_2|$ et P_1 un suffixe de P_2, alors quand le pattern $|P_1|$ est reconnu, $|P_2|$ est reconnu également.

Question 26 (2 points)

> Modifier l'algorithme de la question 24 pour prendre en compte cette solution.

Question 27 (2 points)

> Comparer la complexité de cet algorithme avec celle que la question 16.

> L'algorithme que vous venez de développer se nomme Aho-Corasick string matching. Il peut trouver sa place dans la recherche d'occurences de termes spécifiques dans un document ou encore de recherche d'empreinte de virus dans un programme.

Question 28 (5 points)

> Nous souhaitons maintenant que nos patterns prennent en compte les caractères spéciaux [7] "?" et "*". Comment peut on modifier notre algorithme pour gérer ces nouveaux caractères ? L'usage d'un trie est-il adapté à ce matching ?

7. '?' est utilisé pour désigner n'importe quel caractère et '*' désigne un nombre de caractère compris entre 0 inclus et l'infini.

4 Partie bonus

Les questions suivantes sont plus dures que le reste du sujet. Il est fortement conseillé de finir les autres questions avant.

Question bonus 29 (1 point)

> Proposer un ASCII-ART amusant.

Question bonus 30 (1 point)

> Décrire le plus fidèlement possible le bruit d'un stylo plastique vert sur une vitre humide.

Question bonus 31 (1 point)

> Présenter une option de compilation du kernel Linux.

Question bonus 32 (10 points)

> Proposer et rédiger le plan d'un sujet écrit pour la prochaîne épreuve régionale.

Question bonus 33 (1 point)

> Quel est votre organisateur Prologin favori ?

Question bonus 34 (1 point)

> Quelle est votre organisatrice Prologin favorite ?

Question bonus 35 (1 point)

> Quelle est la bonne écriture du pseudo du président de l'association Prologin ?