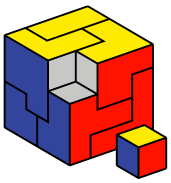
Prologin 2024



Concours national d'informatique

Épreuve écrite d'algorithmique
Lausanne, Louvain-la-Neuve, Paris

Dimanche 25 février 2024

La Bibliothèque de Babel

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. Pendant cette épreuve, vous passerez un entretien (10 minutes) avec un organisateur. Ensuite, vous aurez une épreuve de programmation sur machine (3 heures et 30 minutes) l'après-midi.

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.

— 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.

— Lorsqu'un algorithme est demandé, vous pouvez le décrire avec suffisamment de précision, le pseudo-coder ou l'implémenter dans le langage de votre choix. Dans le dernier cas, veuillez néanmoins préciser le langage 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.

— 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 Préambule

Jøsëf Marchand, en quête de la réponse universelle, se rend à la bibliothèque de Babel. La quantité de livres étant phénoménale, il a besoin de votre aide pour y trouver sa réponse.

Le sujet est composé de 3 parties indépendantes :

1. Automates de recherche (20 points)
2. Knuth-Morris-Pratt (20 points)
3. Aho-Corasick (20 points)

3 L'Automate de Babel

Afin d'aider Josëf Marchand à rechercher la réponse dans la Bibliothèque de Babel, nous allons commencer par lui proposer un algorithme de recherche de mot-clé. Dans cette section, on construira un automate spécialement pour un mot, afin de rechercher rapidement toutes ses occurrences dans la Bibliothèque de Babel.

Notations

- Dans ce sujet, on appelle un *mot* ou *chaîne de caractères* une séquence (potentiellement vide) de lettres majuscules parmi les 26 lettres de l'alphabet français. On appelle Σ l'ensemble des 26 lettres de l'alphabet, c'est à dire $\Sigma = \{\text{«A»}, \text{«B»}, \ldots, \text{«Z»}\}$.
- On écrira $|w|$ pour représenter la longueur d'une chaîne de caractères w. Par exemple, avec $w =$ «PROLOGIN», on a $|w| = 8$. La chaîne de caractères vide $\varepsilon =$ «» a une longueur de $|\varepsilon| = 0$.
- Les chaînes de caractères sont basées à 0, ainsi dans cette section $w[0]$ représente le premier caractère de w, en « position 0 », et $w[|w|-1]$ représente le dernier caractère de w. Par exemple, avec $w =$ «PROLOGIN», on a $w[0] =$ «P», $w[1] =$ «R», $\ldots$, $w[7] =$ «N».
- On note $w[i..j]$ la sous-chaîne de caractères allant du caractère en position i inclus au caractère en position j exclus. Par exemple, avec $w =$ «PROLOGIN», on a $w[3..6] =$ «LOG».
- On note $u \mid\mid v$ la concaténation des chaînes de caractères u et v. Par exemple, si $u =$ «PRO» et $v =$ «LOGIN», on a $u \mid\mid v =$ «PROLOGIN».
- Dans les blocs de pseudo-code donnés, les deux bornes des boucles `pour` sont toujours incluses.

3.1 Algorithme Naïf

Question 1 (3 points)

On considère l'algorithme de recherche de sous-chaîne suivant :

Algorithme 1 : Recherche de sous-chaîne ?

```
Entrées : Une chaîne de caractères w, une potentielle sous-chaîne de caractères u
Résultat : Vrai si u est inclus dans w, Faux sinon
pour i ← 0 à |w| − |u| faire
    j ⟵ 0 ;
    tant que j < |u| et w[i + j] = u[j] faire
        j ⟵ j + 1 // instruction goulot
    fin
    si j = |u| alors
        retourner Vrai
    fin
fin
retourner Faux
```

> Indiquez tout d'abord si l'algorithme est correct. Si l'algorithme est incorrect, expliquez l'erreur et proposez un correctif en appliquant les moindres modifications.

Question 2 (2 points)

> Pour une chaîne de caractères w de taille $|w| = 1000$, estimez le nombre de fois que l'instruction goulot est exécutée, dans le pire cas et dans le cas moyen.
>
> Si vous le souhaitez, vous pouvez apporter une réponse avec les opérateurs de Landau appropriés, en fonction de $|w|$ et $|u|$

Question 3 (2 points)

> Donner un exemple d'entrée qui **maximise** le nombre de fois que l'instruction goulot de l'algorithme naïf est exécutée, avec $|w| = 10$ au maximum. L'objectif est de trouver une entrée qui augmente au maximum le temps d'exécution de l'algorithme.

3.2 Automates

Dans ce sujet, un *automate* est un graphe orienté composé :

— d'un ensemble S de N états allant de 0 à $N-1$, les sommets du graphe

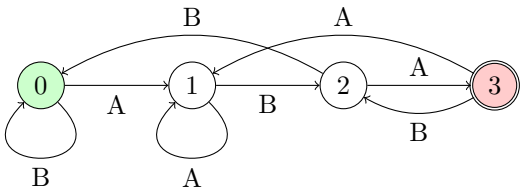
— d'un état initial, l'état 0, coloré en vert,

— d'un état final, l'état $N-1$, coloré en rouge et doublement entouré,

— d'une fonction de transition $\delta : S \times \Sigma \mapsto S$, qui représente les arcs étiquetés du graphe.

Par exemple, en considérant uniquement les lettres A et B, le graphe suivant représente un automate à 4 états, avec la fonction de transition suivante :

— $\delta(0, A) = 1$,
— $\delta(0, B) = 0$,
— $\delta(1, A) = 1$,
— $\delta(1, B) = 2$,
— ...
— $\delta(3, B) = 2$.



Dans notre définition simplifiée d'un automate, le *chemin* associé à un mot w est la suite d'états obtenus en partant de l'état initial (0), et en se déplaçant d'état en état en suivant les arcs étiquetés par les lettres du mot.

Par exemple, dans l'automate précédent, le chemin associé au mot «AABAB» est [0, 1, 1, 2, 3, 2].

Question 4 (2 points)

> Listez au moins 5 mots dont le chemin associé dans l'automate donné en exemple finit sur l'état 2. Donnez une condition nécessaire et suffisante pour qu'un mot finisse sur l'état 2.

Question 5 (2 points)

> Listez au moins 5 mots dont le chemin associé dans l'automate donné en exemple finit sur l'état 3. Donnez une condition nécessaire et suffisante pour qu'un mot finisse sur l'état 3.

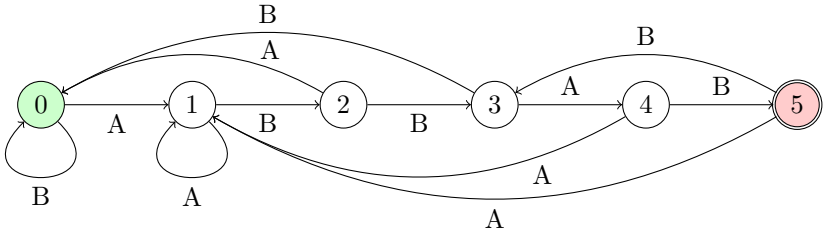
Question 6 (2 points)

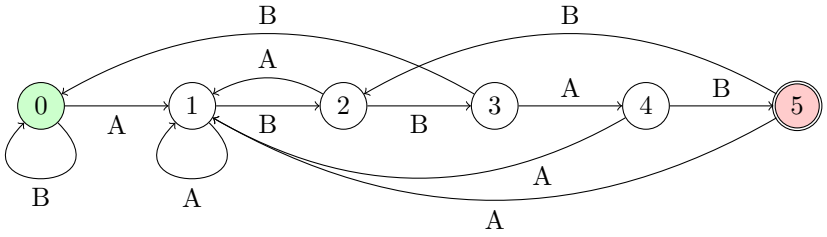
> Considérez le chemin associé au mot w = «AAABABABBABAA». Combien de fois le chemin passe-t-il par l'état 3 ? Comparez le résultat obtenu au nombre de fois que la sous-chaîne «ABA» est présente dans w.

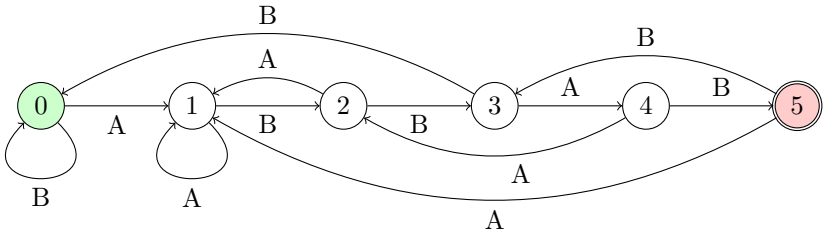
Question 7 (3 points)

On appelle *automate de recherche* de u le plus petit automate qui permet, de cette manière, de détecter toutes les occurrences de u dans w.

> Parmi les trois automates suivants, indiquez s'il s'agit d'un automate de recherche de u = «ABBAB». Si vous pensez qu'il ne s'agit pas d'un automate de recherche, justifiez :
>
> — Soit en trouvant un mot w dont une occurrence de u n'est pas détectée dans le chemin associé à w,
>
> — Soit en trouvant un mot w dont une occurrence de u est détectée là où il ne devrait pas y en avoir.







Question 8 (2 points)

> Étant donné un mot u, un état s et une lettre x, quelle doit être la valeur de $\delta(s, x)$ afin de construire un automate de recherche ? En clair, comment déterminer vers quel état doit mener l'arc partant de s étiqueté par un x ?

Question 9 (2 points)

> Dessinez l'automate de recherche du mot «AAABAB».

4 Knuth-Morris-Pratt

Dans cette section, on va construire l'algorithme de Knuth-Morris-Pratt pas à pas, afin de rechercher n'importe quel mot dans la bibliothèque de Babel en un temps linéaire. Certaines questions peuvent être difficiles, n'hésitez pas à les passer.

Question 10 (2 points)

— On dit qu'un mot u est un *préfixe* d'un mot w si w commence par u. Autrement dit, u est un préfixe de w s'il existe un i tel que $w[0..i] = u$.

— Similairement, on dit qu'un mot u est un *suffixe* d'un mot w si w se finit par u. Autrement dit, u est un suffixe de w s'il existe un j tel que $w[j..|w|] = v$.

— Finalement, on dit qu'un mot u est un *facteur* d'un mot w si u est inclus dans w. Autrement dit, u est un facteur de w s'il existe un i et un j tel que $w[i..j] = u$.

Considérez la chaîne de caractères $w =$ «AAABA». On a alors «AAAB» qui est un des préfixes de w, «ABA» un des suffixes de w, et «AAB» un des facteurs de w.

On appelle Pref(w) l'ensemble des préfixes de w, Suff(w) l'ensemble des suffixes de w et Fact(w) l'ensemble des facteurs de w.

Par exemple, on a Pref(w) = {ε, «A», «AA», «AAA», «AAAB», «AAABA»}.

> Combien existe-t-il de facteurs de w ? Indiquez l'ensemble des facteurs de w.

Question 11 (2 points)

> Considérez la chaîne de caractères $w =$ «AAABAACAAAABAA». Indiquez tous les préfixes de w qui sont également des suffixes de w.

Question 12 (3 points)

Pour une chaîne de caractères w, on appelle *indice LPS* la taille du plus long préfixe de w différent de w qui est également un suffixe de w. Par exemple, l'indice LPS de $w =$ «AABAABA» est 4, car le plus long préfixe différent de w qui est également un suffixe est $u =$ «AABA», de longueur 4.

> Pour chaque préfixe $u \neq \varepsilon$ de $w =$ «AAABAACAAAABAAA», indiquez l'indice LPS de u.

Question 13 (2 points)

> Soit une chaîne u dont le préfixe de taille k est un suffixe de u. Soit $w = u \mid\mid u[k]$. Justifiez que le préfixe de taille $k + 1$ de w est un suffixe de w.

Question 14 (2 points)

> Soient une chaîne u dont l'indice LPS est égal à k, et x n'importe quel caractère. Justifiez que l'indice LPS de $w = u \mid\mid x$ est inférieur ou égal à l'indice LPS de $u + 1$.

Question 15 (2 points)

> Supposons que nous ayons une chaîne u d'indice LPS égal à k. Quel est l'indice LPS de $w = u \mid\mid u[k]$? Justifiez.

Question 16 (2 points)

> On appelle le tableau LPS de u, noté LPS[i], le tableau indiquant pour tout i allant de 1 à $|u|$ l'indice LPS de $u[0..i]$. La question précédente nous permet de trouver l'indice LPS de $u \mid\mid u[\text{LPS}[|u|]]$. Qu'en est-il de l'indice LPS de $w = u \mid\mid u[\text{LPS}[\text{LPS}[|u|]]]$?

Question 17 (2 points)

> En étendant ce principe, proposez un algorithme qui calcule le tableau LPS d'un mot w, avec une complexité temporelle de $O(|w|)$. Prouvez la complexité de votre algorithme.

Question 18 (3 points)

> En supposant que vous pouvez calculer le tableau LPS de n'importe quelle chaîne w avec une complexité temporelle de $O(|w|)$, proposez un algorithme qui permet de trouver toutes les occurrences d'une sous-chaîne u dans une plus grande chaîne v avec une complexité temporelle de $O(|v|)$.

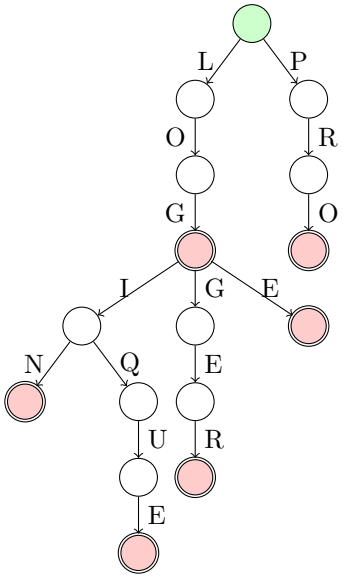
5 Aho-Corasick

On va désormais étendre l'automate vu dans la première section pour supporter la recherche de plusieurs mots-clés en simultané.

On appelle une *trie*[a] un arbre enraciné permettant de stocker un ensemble de mots de la manière suivante : Soit W un ensemble de mots.

- L'arbre possède un noeud pour chaque préfixe de chaque mot dans W.
- La racine représente le mot vide, ε.
- Le noeud représentant le mot w est relié par une arête étiquetée x au noeud représentant $w \mid\mid x$.
- Les noeuds représentant un mot appartenant à W sont colorés et doublement entourés.

Par exemple, la trie suivante représente l'ensemble de mots {«LOG», «LOGE», «LOGGER», «LOGIN», «LOGIQUE», «PRO»}



a. Aussi couramment appelé *arbre préfixe*

Question 19 (2 points)

> Représentez la trie correspondant à l'ensemble de mots
> $W = \{$«A», «AB», «BAB», «BC», «BCA», «C», «CAA»$\}$

Question 20 (2 points)

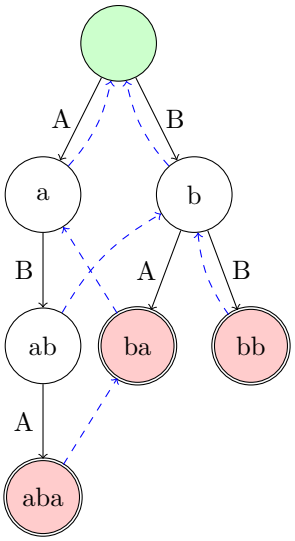
> Soit W un ensemble de mots dont nous connaissons uniquement la trie T associée. Étant donné un mot w, proposez un algorithme pour vérifier si w appartient à W. Quelle est la complexité de votre algorithme ?

On va rajouter des arêtes à la trie afin de pouvoir effectuer une recherche de sous-chaîne dans un grand texte, pour toutes les sous-chaînes dans W en même temps.

Question 21 (2 points)

Voici une première tentative d'adaptation d'une trie pour rechercher les occurrences des trois mots-clés $W = \{$«ABA», «BA», «BB»$\}$ dans un texte.

On commence à la racine de l'arbre, et pour chaque lettre du texte, on tente de parcourir la trie. Lorsque aucun arc ne dispose de la lettre désirée depuis le noeud courant, alors on emprunte un arc en bleu pointillé, appelés *arcs suffixes*, et on retente.



Par exemple, supposons que nous recherchions les occurrences des mots-clés {«ABA», «BA», «BB»} dans la chaîne w = «BABABB» :

— On part de la racine de l'arbre. On lit $w[0] = B$, alors on descend dans le fils droit de la racine, b ;

— On lit $w[1] = A$, alors on descend dans le fils gauche du noeud courant, ba.

— On lit $w[2] = B$. Comme aucun fils du noeud courant n'est étiqueté par un B, on emprunte l'arc suffixe, on arrive sur le noeud a, et on retente de lire B. Cette fois-ci, un fils correspondant existe, alors on visite son fils, ab.

— On lit $w[3] = A$. On visite alors la feuille de la branche gauche, aba.

— On lit $w[4] = B$. Comme la feuille ne possède pas de fils étiqueté B, on remonte par l'arc suffixe, on arrive sur ba. Nous sommes toujours sur une feuille, qui n'a donc pas de fils étiqueté B, on remonte une nouvelle fois par l'arc suffixe. On arrive alors au fils gauche de la racine, a, qui possède enfin un fils étiqueté B. On réussit finalement à lire le B par cette arête, on arrive sur l'état ab.

— On lit enfin $w[5] = B$. Comme aucun fils n'est étiqueté par un B, on commence par emprunter l'arc suffixe jusqu'à b, puis on descend par le fils droit dans le noeud bb.

> Lors de ce parcours, quels noeuds colorés ont été visités, et combien de fois ? Comparez le résultat obtenu avec le nombre d'occurrences des mots-clés correspondants.

Question 22 (1 point)

> Donnez une chaîne de caractères plus courte dont le chemin dans la trie passe une unique fois par les trois feuilles.

Question 23 (2 points)

> Effectuez le parcours du chemin associé à la chaîne «BAABA». Comparez le nombre de fois que vous visitez un noeud coloré au nombre d'occurrences des trois mots-clés.

Question 24 (3 points)

Afin de remédier au problème que soulève la question précédente, dès que nous réussissons à lire une lettre, on parcourt tous les arcs suffixes depuis le noeud d'arrivée jusqu'à la racine afin de ne manquer aucune occurrence d'un préfixe.
Par exemple, l'arbre suivant montre la trie obtenue pour l'ensemble de mots-clés {«A», «ABAB», «B», «BABA»}. Si nous parcourons, par exemple, la chaîne w = «ABABA», on va dorénavant :

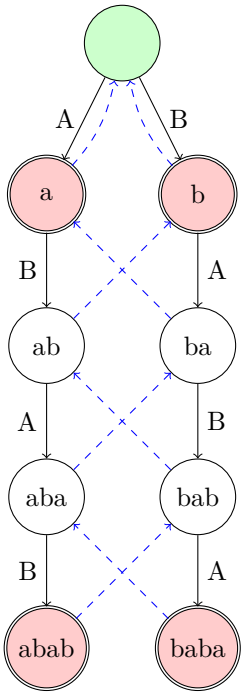
— Lire $w[0]$ = «A», aller sur a, et reconnaître le mot-clé «A»,

— Lire $w[1]$ = «B», aller sur ab, parcourir la chaîne d'arcs suffixes jusqu'à la racine et reconnaître le mot-clé «B»,

— Lire $w[2]$ = «A», aller sur aba, parcourir la chaîne d'arcs suffixes depuis aba jusqu'à la racine et reconnaître le mot-clé «A»,

— Lire $w[3]$ = «B», aller sur $abab$, reconnaître le mot-clé «ABAB», parcourir la chaîne d'arcs suffixes jusqu'à la racine et reconnaître le mot-clé «B» également.

— Lire $w[4]$ = «A», prendre l'arc suffixe jusqu'à bab puis descendre sur $baba$, reconnaître le mot-clé «BABA», puis finalement parcourir la chaîne d'arcs suffixes jusqu'à reconnaître une dernière fois le mot-clé «A».



> Quelle est maintenant la complexité temporelle pire cas du traitement d'un texte par une trie contenant 4 noeuds colorés ?

Question 25 (5 points)

Proposez une solution pour améliorer la complexité pire-cas de la recherche des mots-clés dans l'arbre. Vous pouvez apporter des modifications à la trie si vous le souhaitez.

Question 26 (3 points)

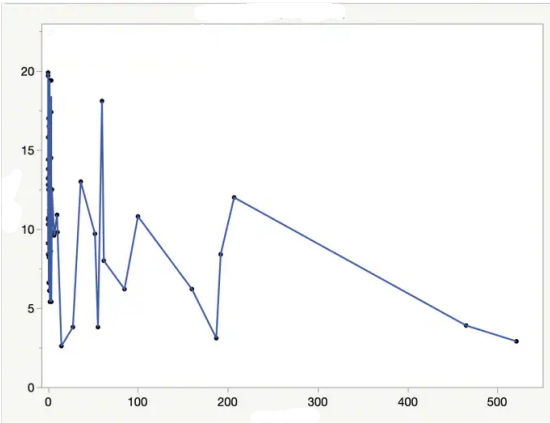
Rajoutez les arcs suffixes à votre trie de la question 19.

6 Questions bonus

N'abordez ces questions que si vous vous êtes relu 42 fois.

Question bonus 27 (1 point)

> Légendez et donnez un titre au graphique suivant :



Question bonus 28 (1 point)

> Coloriez les cases d'une page entière de votre copie comme une grille de puissance 4 qui fait égalité, sans motif répété.

Question bonus 29 (1 point)

> Indiquez le prénom de l'organisateur vous ayant fait passer l'entretien.

Question bonus 30 (1 point)

> Répondez à la question 25 sans utiliser la lettre «e».

Question bonus 31 (1 point)

> Devinez une manière dont le rédacteur du sujet a pu écorcher "Aho-Corasick". Si votre tentative apparaît dans son historique de recherche, vous remportez le point.