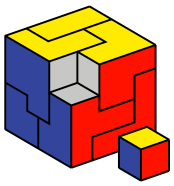
Prologin 2019



Concours national d'informatique

Épreuve écrite d'algorithmique
Angers, Lille et Marseille

9 février 2019

ProloPorte

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, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (4 heures).

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.

Après avoir récupéré un local associatif digne de ce nom, les membres de Prologin veulent permettre l'accès à celui-ci à un maximum de membres sans pour autant compromettre la sécurité de ce dernier [1]. Pour cela, Haruhi Suzumiya a une idée, elle veut installer un digicode sur la porte. Pour des raisons de sécurité, chaque membre de l'association doit avoir un code d'accès différent [2].

2 Introduction

Question 1 (1 point)

> Décrivez une structure de données qui permet de représenter la liste des mots de passe enregistrés sur le digicode.

Question 2 (1 point)

> Écrivez une fonction `authentification(mdp, liste_mdp)` qui retourne «vrai» si le mot de passe donné fait partie de la liste des mots de passe donnée en entrée et «faux» sinon. t

Question 3 (1 point)

> Estimez le nombre d'instructions exécutées par l'algorithme écrit pour la question 2 en fonction du nombre de mots de passe enregistrés sur le digicode.

Au fur et à mesure qu'Haruhi enregistre des mots de passe sur le digicode, elle commence à recevoir des plaintes des autres organisateurs qui trouvent la porte de plus en plus lente à s'ouvrir.

Question 4 (2 points)

Haruhi décide qu'elle va maintenant transformer la liste des mots de passe en une liste triée avant de l'enregistrer sur le digicode. Cette opération peut se faire depuis son ordinateur, qui a un processeur plus puissant que le digicode [3], et ce en environs $n \log n$ étapes de calcul, où n est le nombre de mots de passes à enregistrer.

> Écrivez une nouvelle fonction `authentification_rapide(mdp, liste_mdp)` pour le digicode qui prend en compte le fait que la liste des mots de passe est maintenant triée.

Question 5 (2 points)

> Estimez le nombre d'instructions exécutées par l'algorithme écrit pour la question 4 en fonction du nombre de mots de passe enregistrés sur le digicode.

3 Automates

Malheureusement, il n'y a plus assez d'espace mémoire sur le digicode pour enregistrer tous les codes. Après quelques recherche rapide sur internet, Haruhi pense que la meilleure solution pour pallier à ce grand problème est d'utiliser des **automates** [4].

Un automate déterministe est un algorithme qui peut entrer dans un nombre fini d'**états** q_0, q_1, ..., q_n.

- Avant de commencer à rentrer un mot de passe, il est dans l'**état initial** q_0.
- Lorsque l'on appuie sur un chiffre x du digicode, il peut changer d'état vers un nouvel état q_v qui peut être déterminé en connaissant seulement l'état actuel q_u et le chiffre appuyé x. On peut alors définir un ensemble de **transitions** δ tel que $(q_u, x, q_v) \in \delta$ si l'automate peut passer de l'état q_u à l'état q_v lorsque l'on appuie sur la touche x. Si l'automate est dans l'état q_u et qu'il n'y a pas de transition permettant de changer d'état en lisant x depuis l'état q_u, alors le mot de passe est **refusé**.
- On considère que le mot de passe entré est **accepté** si l'état de l'automate après avoir rentré celui-ci sur le digicode est l'**état final**, q_n.

1. Il est important que les membres puissent rentrer dans le local pour travailler, mais moins important qu'ils ne puissent ressortir.
2. Qui sera différent de son prénom, de "0000" et "fmocnqiufscjqsgk" pour éviter les mots de passe trop évidents.
3. On suppose que les digicode de la NASA ne sont pas concernés par ce sujet.
4. Indépendemment de son envie de manger des crêpes aux tomates.

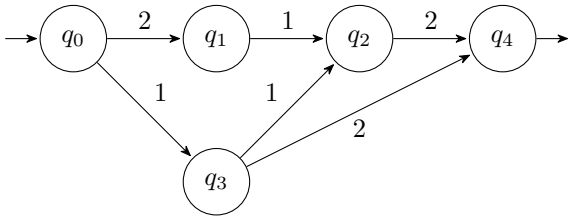


FIGURE 1 – Un automate, qui contient 5 états, q_0 est l'état initial et q_4 est l'état final

L'exemple sur la figure 1 représente un automate qui a pour ensemble de transitions $\delta = \{(q_0, 2, q_1), (q_1, 1, q_2), (q_2, 2, q_4), (q_0, 1, q_3), (q_3, 1, q_2), (q_3, 2, q_4)\}$. En pratique on utilisa ce type de dessins dans le reste du sujet pour représenter les automates, il suffit d'ajouter une flèche qui relie deux états en l'étiquetant avec un chiffre x si l'automate passe d'un état à l'autre en appuyant sur x. L'état initial est identifié par une flèche entrante, l'état final est identifié par une flèche sortante. L'automate de la figure 1 accepte les mots de passe 212, 112, et 12.

- 212 est accepté en allant de l'état q_0 à q_1 à q_2 et enfin à l'état q_4.
- 112 est accepté en allant de l'état q_0 à q_3 à q_2 et enfin à l'état q_4.
- 12 est accepté en allant de l'état q_0 à q_3 et enfin à l'état q_4.

On remarque qu'on peut définir de façon équivalente les mots acceptés par un automate en utilisant leur représentation par un graphe : un mot de passe est accepté par l'automate s'il existe un chemin qui va d'un état initial à un état final en passant par des transitions (les flèches) étiquetées par les chiffres successifs du mot de passe.

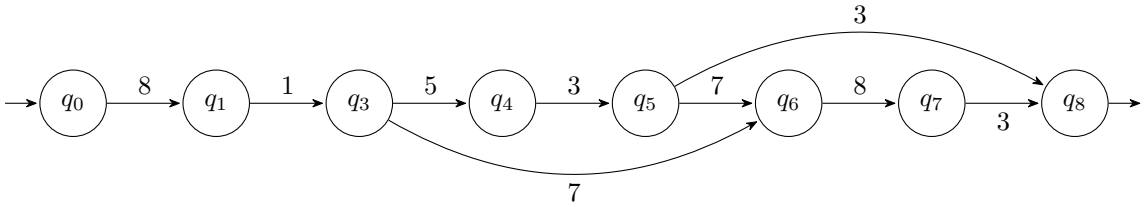


FIGURE 2 – Un automate, qui contient 9 états, q_0 est l'état initial et q_8 est l'état final

Question 6 (1 point)

> Listez les codes acceptés pour le digicode donné sur la figure 2.

Question 7 (2 points)

> Dessinez un automate permettant d'obtenir un digicode qui accepte uniquement les codes 4242 et 4372. Cet automate devra avoir au plus 6 d'états pour obtenir tous les points à la question.

Ce qui intéresse Haruhi, c'est qu'en utilisant des automates, elle pense pouvoir pouvoir accepter un nombre infini [5] de mots de passe en utilisant le peu de mémoire du digicode.

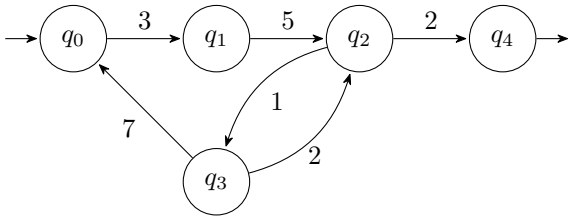


FIGURE 3 – Un automate, qui contient 5 états, q_0 est l'état initial et q_4 est l'état final

Question 8 (2 points)

5. En prévision du jour où l'association aura un nombre infini de membres.

> Trouvez 5 mots de passe acceptés par le digicode décrit sur la figure 3. Sauriez-vous décrire l'ensemble de tous les mots de passes acceptés ?

Lorsque quelqu'un fait une faute de frappe en inscrivant son mot de passe, il doit appuyer sur le bouton "réinitialiser" sur le digicode puis réécrire le mot de passe en entier. Ce n'est pas pratique, on aimerait avoir un système plus souple.

Question 9 (2 points)

Le mot de passe de Haruhi est 4242 [6], elle aimerait ne pas avoir à réinitialiser le digicode à chaque fois qu'elle fait une faute de frappe en écrivant son mot de passe. Par exemple, si elle écrit 44242, 3924242 ou bien 4244242, etc... elle aimerait que le digicode ouvre la porte.

Astuce : lorsque vous dessinez un automate, s'il y a plusieurs transition entre deux états q_u et q_v qui correspondent à la lecture des chiffres x_1, x_2, etc... vous pouvez ne dessiner qu'une seule flèche étiquetée par "x_1, x_2, ...".

> Dessinez un automate qui accepte n'importe quelle séquence de chiffres qui termine par 4242. Cet automate devra avoir 5 états pour obtenir tous les points à la question.

Question 10 (2 points)

Il est conseillé de prendre le temps de regarder les autres questions du sujet avant répondre à celle-ci [7], en effet votre réponse sera utilisée pendant tous le reste du sujet.

> Décrivez une structure de données qui permet de représenter un automate.

Question 11 (2 points)

> Écrivez une fonction `exec_deterministe(automate, mdp)` qui exécute l'automate déterministe donné, c'est-à-dire qui retourne «vrai» si le mot de passe donné est accepté par l'automate, et «faux» sinon.

Question 12 (2 points)

> Estimez le nombre d'instructions exécutées par l'algorithme écrit pour la question 11 en fonction de la taille de l'automate et du mot de passe.

4 Non déterminisme

Maintenant que Haruhi sait ce qu'elle va faire de son digicode, elle aimerait bien implémenter des algorithmes qui lui permetraient de manipuler plus facilement les mots de passes acceptés par le digicode.

Elle se rend compte qu'elle va maintenant avoir besoin de travailler avec des automates plus compliqués, qui ne seront plus forcément déterministe [8] :

- Ils peuvent avoir plusieurs états initiaux, on notera I l'ensemble des états initiaux.
- Ils peuvent avoir plusieurs états finaux, on notera F l'ensemble des états finaux.
- Lorsque l'on rentre un chiffre x depuis un état q_u, il est possible qu'il y ai des états q_v et q'_v distincts qui soient tous deux des nouveaux états possible de l'automate (autrement dit il est possible que $(q_u, x, q_v) \in \delta$ et $(q_u, x, q'_v) \in \delta$).

Un mot de passe $x_1x_2...x_n$ est alors accepté par l'automate s'il y a des états $q_0q_1...q_n$ tels que $q_0 \in I$, $q_n \in F$ et pour tout $i \in \{1...n\}$, $(q_{i-1}, x_i, q_i) \in \delta$. Autrement dit, s'il existe un chemin dans le graphe représentant l'automate qui commence à un état initial, termine à un état final et tel que les flèches successives de ce chemin sont étiquetées par les lettres successives du mot de passe entré.

Par exemple, l'automate donné sur la figure 4 accepte, entre autres, les mots de passe suivants :

- 351, en passant par les états q_0, q_1, q_2 puis q_4
- 351751, en passant par les états q_0, q_1, q_2, q_3, q_1, q_2 puis q_4
- 7351, en passant par les états q_3, q_0, q_1, q_2 puis q_4

6. Qui ne fait étrangement pas partie des mots de passe évidents interdits.
7. Il n'y a pas de blague nulle ici, c'est un vrai conseil !
8. On peut très originalement appeler ça "automate non déterministe".

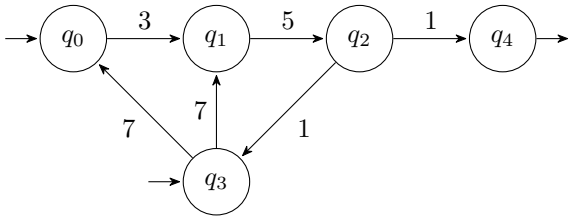


FIGURE 4 – Un automate non déterministe, qui contient 5 états, q_0 et q_3 sont des états initiaux et q_4 est le seul état final

Question 13 (1 point)

> Trouvez 5 mots de passe acceptés par le digicode décrit sur la figure 4. Sauriez-vous décrire l'ensemble de tous les mots de passes acceptés ?

Question 14 (1 point)

> Si c'est nécessaire, adaptez la structure de données décrite à la question 10 pour qu'elle puisse représenter aussi des automates non déterministe.

Question 15 (3 points)

> Écrivez une fonction `exec(automate, mdp)` qui exécute l'automate donné, c'est-à-dire qui retourne «vrai» si le mot de passe donné est accepté par l'automate, et «faux» sinon.

Question 16 (2 points)

> Estimez le nombre d'instructions exécutées par l'algorithme écrit pour la question 15 en fonction de la taille de l'automate et du mot de passe.

5 Features

Quand de nouveaux membres rejoignent les organisateurs de Prologin, Haruhi aimerait pouvoir rajouter de nouveaux mots de passes disponibles.

Question 17 (2 points)

> Étant donné deux automates, comment construire un nouvel automate qui accepte un mot de passe s'il est accepté par l'un de ces deux automates ?

Question 18 (2 points)

> Écrivez une fonction `union(automate1, automate2)` qui retourne un automate qui accepte un mot de passe s'il aurait été accepté par l'un des deux automates donnés en entrée.

Lorsqu'un membre quitte l'association, Haruhi aimerait révoquer son accès. Pour cela, il suffit de faire en sorte que l'automate n'accepte plus le mot de passe de cette personne.

Question 19 (2 points)

Dans un premier temps, on va construire un automate qui accepte tous les mots de passe, sauf celui de ce membre. On aura ensuite plus qu'à vérifier qu'un mot de passe est valide avec l'ancienne version du digicode, mais pas valide pour l'automate qui accepte tous les mots de passe sauf celui de la personne qui quitte l'association.

> Dessinez un automate qui accepte tous les mots de passe, sauf 4242.
> Comment généraliser cette construction pour n'importe quel mot de passe $x_1x_2...x_n$?

Question 20 (2 points)

> Écrivez une fonction `automate_refus(mdp)` qui retourne un automate qui accepte tous les mots de passe sauf celui donné en entrée.

Question 21 (3 points)

> Étant donné deux automates, comment construire un automate qui accepte un mot de passe s'il aurait été accepté par ces deux automates ?

Question 22 (3 points)

> Écrivez une fonction `intersection(automate1, automate2)` qui retourne un automate qui accepte un mot de passe s'il aurait été accepté par les deux automates donnés en entrée.

Question 23 (3 points)

L'intérêt de générer des automates est que le plus gros temps de calcul peut être fait une seule fois et sur un ordinateur, une fois l'automate généré il peut être envoyé sur le digicode qui n'a plus qu'à exécuter l'algorithme que vous avez écrit pour la question 15 à chaque fois qu'un mot de passe est rentré.

> Estimez le nombre d'états et de transitions des automates qui sont générés par les fonctions que vous avez écrit pour les questions 18, 20 et 22.

6 Section Bonus : Optimisation

Cette section est facultative et comporte des questions plus difficiles, il est déconseillé de commencer à y répondre avant d'avoir bien fini les questions qui précèdent.

Question bonus 24 (1 point)

Haruhi vous demande s'il elle pourra représenter n'importe quel ensemble de mots de passe avec des automates. Vous êtes persuadés que non mais vous devez la convaincre avec un contre-exemple.

> Montrer qu'il n'est pas possible d'avoir un digicode qui accepte seulement les mots de passe 01, 0011, 000111, ..., 00000001111111, etc... Cet ensemble de mots de passe peut se définir comme l'ensemble des 0^n1^n.

Au fil des opérations sur le digicode, l'automate stocké grossi, et Haruhi aimerait réduire autant que possible la taille de celui-ci afin de s'assurer qu'il ne risque pas de ne plus tenir dans la mémoire du digicode.

En analysant les automates créés par ses opérations, Haruhi constate que certains états de l'automate sont redondants, en effet il arrive qu'il existe des états q_u et q_v distincts, tels que les séquences de chiffres acceptés en partant de ces deux états soient les mêmes.

Question bonus 25 (1 point)

Haruhi veut aussi en profiter pour accélérer le digicode, elle veut transformer l'automate en un automate déterministe afin de pouvoir utiliser l'algorithme décrit à la question 11 qui est plus rapide.

> Écrivez une fonction `déterminise(automate)` qui retourne un automate déterministe qui accepte les même mots de passe que l'automate donné en entrée.
>
> Une façon de faire est de retourner un automate dont les états représentent les combinaisons d'états dans lequel l'automate donné en entrée peut être à un instant de la lecture du mot de passe.

Question bonus 26 (1 point)

> Écrivez une fonction `indistinguables(automate, q_u, q_v)`, qui retourne «vrai» si les séquences de chiffres à entrer pour que l'automate déterministe donné accepte depuis les états q_u et q_v sont les mêmes et retourne «faux» sinon.

> Il est possible que vous ayez besoin de pré-calculer une structure ne dépendant pas de q_u et q_v avant d'exécuter cette fonction, vous pouvez donner l'algorithme qui calcule cette structure et supposer que vous y avez accès.
>
> Quelle est la complexité de votre algorithme ?

Grâce à votre algorithme, Haruhi peut maintenant identifier et éliminer les états redondants !

7 Bonus

Question bonus 27 (0 point)

> Pensez-vous qu'utiliser des automates soit une bonne façon de résoudre le problème de mémoire du digicode ?

Question bonus 28 (0 point)

> Dessinez un automate mignon.

Question bonus 29 (0 point)

> Pensez-vous que les automates mignons vont détrôner les chatons sur internet ?

Question bonus 30 (0 point)

> Dessinez des tomates-auto [9].

Le sujet comporte 7 pages (sans compter la page de garde), 30 questions, et 3 questions bonus. Les questions normales sont notées sur 44 points, et les questions bonus rapportent au total 3 points, plus 1 point de présentation.

9. Nous non plus nous ne savons pas...