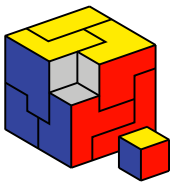
Prologin 2019



Concours national d'informatique

Épreuve écrite d'algorithmique
École Polytechnique

17 février 2019

Sorcellerie algorithmique

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.

2 Préface

Le 9 mai 1969, à deux kilomètres de la frontière nord du Laos, la 101ème Division d'Infanterie de l'armée américaine progresse vers la colline 937.

Pour eux, c'est une simple reconnaissance. Pour le Viêt-Cong, la colline 937 est une zone stratégique.

La dizaine d'appelés de la 101ème Division, peu formée au combat, devait effectuer cette mission de routine en moins de deux heures.

Ils résisteront héroïquement pendant neuf jours.

Ce sujet ne raconte pas leur histoire. [1]

3 Introduction

La rentrée scolaire magique approche à grand pas, et comme chaque année, les sorciers doivent choisir leur baguette magique. Pourtant, ce n'est pas toujours une mince affaire : cœur en ventricule de dragon, poil de rougarou, bois d'érable, de cèdre, rigide, flexible, …. Des centaines de possibilités, rendant chaque baguette unique, avec son propre caractère. De plus, tout le monde sait que ce n'est pas nécessairement le sorcier qui choisit la baguette, mais parfois l'inverse !

Pour éviter des émeutes et disputes entre sorciers et baguettes, le marchand [3] local souhaite développer un algorithme pour assigner les paires.

On supposera qu'il y a n sorciers et n baguettes, de telle sorte que chaque étudiant en magie se voit attribuer une seule baguette et qu'aucune baguette ne reste sans maître. Tous les sorciers ont une liste triée regroupant toutes les baguettes et indiquant leur ordre de préférence. De même, chaque baguette tient une liste triée des n sorciers, dans l'ordre d'affection.

4 Un couplage pas comme les autres

Toute la difficulté d'un système est de déterminer un **couplage** entre les sorciers et les baguettes. Évidemment, on ne peut pas choisir n'importe quel couplage.

Prenons l'exemple ci-dessous avec $n = 3$:
Voici l'ordre de préférence des sorciers en termes de baguette (de la plus souhaitée jusqu'à la moins désirée) :
— Joseph Marchand : 3 2 1
— Haruhi Suzumiya : 3 1 2
— Arie Pauteur : 1 2 3

De même, voici l'ordre de préférence des baguettes concernant les sorciers :
— Baguette 1 : Joseph, Arie, Haruhi
— Baguette 2 : Arie, Joseph, Haruhi
— Baguette 3 : Arie, Haruhi, Joseph

On définit un couplage comme étant **instable** s'il existe un sorcier s et une baguette b telle que s préfère b à sa baguette déjà assignée, et de même b préfère s à son sorcier actuellement associé. Autrement dit, une paire est instable si un sorcier et une baguette préfère être ensemble tous les deux, plutôt qu'avec leurs choix actuels. Un couplage où cela ne se produit jamais est donc **stable**.

1. Mais si vous avez la référence, 200 points pour Gryffondor [2] !
2. Pas pour vous malheureusement
3. Non il ne s'appelle pas Joseph, désolé.

Question 1 (2 points)

> En reprenant l'exemple précédent, déterminez un couplage stable attribuant à chaque sorcier une baguette. Donnez un autre couplage, non stable cette fois-ci.

Nous cherchons dans cette partie à developper un algorithme qui part d'une assignation initiale et la modifie jusqu'à ce qu'elle soit stable.

On représentera par la suite les préférences de chacun avec des tableaux [4], par exemple : `sorciers[i][0]` contient le numéro de la baguette préférée (rang 0) du $i$ème sorcier.

Question 2 (2 points)

> Écrire une fonction `couplage_aleatoire(n)` qui assigne de manière quelconque les n sorciers aux n baguettes. Cette fonction doit renvoyer une liste de couples de la forme `(sorcier, baguette)`.

Question 3 (3 points)

> Écrire une fonction `couples_non_stables(sorciers, baguettes, couplage)` qui prend en argument les préférences de chacun ainsi qu'un couplage déjà formé, et renvoie la liste des paires instables.

Question 4 (4 points)

> Écrire une fonction `assigner_iteratif(sorciers, baguettes, paires)` qui forme des paires stables à partir des paires instables, jusqu'à ce qu'il n'y en ait plus.

Question 5 (3 points)

> Justifier succinctement que l'algorithme précédent renvoie un couplage stable.

Question 6 (4 points)

> Montrer que cet algorithme n'est pas garanti de toujours terminer.

5 Mener l'algorithme à la baguette

Le pauvre dirigeant du magasin est constamment en train de changer les paires de sorciers et de baguettes dans l'espoir de satisfaire tout le monde, mais sans cesse des paires instables apparaissent. Fatigué, de voir une telle inefficacité, vous décidez de ~~lancer un sort~~ rédiger votre propre programme. [5]

Question 7 (3 points)

> Écrire une fonction `proposer(sorciers, baguettes, paires, i, j)` qui permet au sorcier i de proposer à la baguette j d'être son maître. Cette dernière n'accepte que si elle n'est pas encore assignée à un sorcier, ou bien qu'elle préfère i au sorcier qui lui est assigné. La fonction devra renvoyer la nouvelle liste de paires formées.

Question 8 (2 points)

> Pourquoi un sorcier n'a aucun intérêt à proposer à une baguette à qui il a déjà proposé dans le passé ?

Question 9 (2 points)

> À quelle baguette de sa liste de préférences, un sorcier a-t-il intérêt de proposer pour optimiser ses chances de former une paire stable ?

Question 10 (5 points)

4. Si vous souhaitez utiliser une autre structure de données, précisez le avant de répondre à la question.
5. Eh oui, pour lancer des sorts il faudrait déjà avoir une baguette !

Écrire une fonction `assigner_glouton(sorciers, baguettes)` qui prend en argument les listes de préférences et retourne un couplage stable.

Question 11 (4 points)

Combien d'appels à la fonction `proposer` effectue-t-on dans le pire des cas ?

Question 12 (3 points)

Montrer que l'algorithme renvoie bien une assignation stable.

Question 13 (3 points)

En déduire qu'il existe bien une assignation stable, peu importe les préférences des sorciers et des baguettes.

Question 14 (2 points)

La solution d'assignation stable est-elle toujours unique ?

Question 15 (4 points)

Pensez-vous que l'algorithme joue en la faveur des sorciers ou des baguettes ? Justifier.

Question 16 (3 points)

Suivant votre réponse à la question précédente, montrer que les sorciers (ou baguettes) ne peuvent pas avoir une meilleure assignation, sans que l'assignation formée soit instable.

Question 17 (3 points)

Comment modifier l'algorithme pour qu'il favorise l'autre classe que celle qu'il favorise en sa version actuelle ?

6 Bonus

Attention, merci d'aborder les questions suivantes uniquement si vous avez un niveau en magie suffisant [6], et que vous avez terminé le reste du sujet.

Question 18 (8 points)

Généraliser l'algorithme pour prendre en compte les sorciers pouvant choisir plusieurs baguettes. On pourra se contenter de décrire l'algorithme avec du pseudo-code, ou de l'expliquer sans considérations techniques.

Question 19 (3 points)

Votre algorithme favorise-t-il les sorciers ou les baguettes ? Justifier.

Question 20 (4 points)

Modifier votre algorithme pour qu'il favorise l'autre classe.

Question 21 (6 points)

6. Le test est plutôt simple : à quelle carte est-ce que je pense ?

> Dans la partie 4, le marchand doit échanger des baguettes de main de sorciers une infinité de fois. Comment peut-il s'y prendre pour réaliser une infinité de ces opérations en un temps fini ? On s'affranchira de toute considération relativiste et quantique [7].

7 Bonus (les vrais)

Attention, merci de ne pas aborder les questions suivantes, elles n'ont pas vraiment d'intérêt (ou bien c'est en réalité un piège pour vous empêcher d'avoir tous les points ? [8]).

Question 22 (42 points)

> Racontez l'histoire entamée en préface, en remplaçant dans vos phrases le mot «bombardement» par «magie noire».

Question 23 (5 points)

> Inventez une formule magique (nom du sort et effet).

Question 24 (0 point)

> Dessinez un penguin [9].

8 Conclusion

« Odi panem quid meliora ». Ça veut rien dire, mais je trouve que ça boucle bien.

Roi Loth, Kaamelott, Livre V

7. Bien entendu, une réponse en lien avec la magie n'est pas celle attendue.
8. Non.
9. Certains organisateurs préfèrent les manchots [10]
10. Certains organisateurs préfèrent les penguins [9]