Prologin 2013

Concours national d'informatique

Épreuve écrite d'algorithmique
Lyon

Samedi 23 février 2013

MinisterMind

Extrait de *Quai d'Orsay*, par Blain & Lanzac

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 (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 Sujet

Introduction

L'alinéa 2 de l'article 8 de la Constitution du 4 octobre 1958, texte fondateur de la V^{e} République, énonce :

> « Sur la proposition du Premier Ministre, [le Président de la République] nomme les autres membres du Gouvernement et met fin à leurs fonctions. »

En d'autres termes, le Premier ministre propose au président la liste des ministres qui composeront le nouveau gouvernement. Mais, même en période de cohabitation, le président de la République peut refuser de nommer certaines personnes : ainsi, mardi 18 mars 1986, lorsque le Premier ministre arriva à l'Élysée pour présenter la liste des 13 ministres du gouvernement, le président de la République y jeta un rapide coup d'œil et lui répondit :

« 9 bien placés, 1 mal placé. »

La coutume veut que le secrétaire général de la présidence communique la liste des membres du gouvernement à la presse sur le perron de l'Élysée. Récemment employé au gouvernement, vous devez aider le Premier ministre à trouver la bonne combinaison de ministres qui recevra l'approbation du président de la République.

Les ministres potentiels étant numérotés de 1 à $n \geqslant 13$, une combinaison sera représentée en mémoire comme une liste d'entiers $[m_1, \ldots, m_{13}]$.

Question 1 (2 points)

> Écrire une fonction qui retourne 1 si la combinaison est correcte, c'est-à-dire qu'elle ne contient aucun ministre en double, 0 sinon.

Question 2 (3 points)

> Écrire une fonction qui prend en argument :
> – une liste ℓ_1 que vous avez précédemment remise au président ;
> – deux entiers x et y correspondant à la réponse du président :
> « x bien placé(s), y mal placé(s) » ;
> – une liste ℓ_2 que vous souhaitez remettre au président.
> et retourne 1 si la liste ℓ_2 concorde avec la réponse du président sur la liste ℓ_1, 0 sinon.

Après quelques recherches sur Internet, vous tombez sur le sujet de l'épreuve régionale de Lyon du concours national d'informatique Prologin 2013, *MinisterMind*, mentionnant l'existence d'un algorithme de Donald Knuth qui résout le MasterMind un nombre de coups minimum dans le pire cas, ainsi que d'autres approches qui minimisent l'entropie. Après en avoir fait part au Premier ministre, celui-ci vous répond : « Vous avez raison. Il faut à tout prix minimiser l'entropie. » et vous charge d'écrire son prochain discours.

L'écriture de discours ministériels n'est pas chose aisée. Un *discours* est une liste de mots, dont certains plaisent au Premier ministre et d'autres non. On suppose qu'il existe une fonction d'appréciation (connue du Premier ministre seul) qui attribue à chaque mot un entier, possiblement négatif, d'autant plus grand que le mot lui plaît.

Index	0	1	2	3	4	5	6	7
Discours	`J'`	`ai`	`décidé`	`de`	`dissoudre`	`l'`	`Assemblée`	`nationale.`
Appréciation	10	20	42	0	-10	0	-10	20
Sous-discours	`J'`	`ai`	`décidé`	`de`		`l'`	`Assemblée`	
Extrait						`l'`	`Assemblée`	`nationale.`

TABLE 1 – Exemples de discours, sous-discours d'appréciation 62, extrait d'appréciation 10.

On appellera *sous-discours* d'un discours un nouveau discours obtenu en retirant des mots à ce discours et on appellera *extrait* d'un discours une suite de mots consécutifs dans ce discours [1].

On dit qu'un discours *plaît* au ministre si et seulement si la somme des appréciations de ses mots est strictement positive. Pour plus de commodité, un discours sera représenté en mémoire par un tableau d'entiers (qui représenteront par exemple les indices des mots dans d'un dictionnaire).

L'algorithme `verdict` qu'effectue le Premier ministre sur un discours que vous lui remettez est donc le suivant :

```
Prendre le discours
Faire la somme des appréciations de tous les mots du discours
Si cette somme est strictement positive
  Retourner Vrai # Pas mal.
Sinon
  Retourner Faux # On n'y est pas du tout. Il y a encore beaucoup de travail.
```

Dans toutes vos fonctions, vous pourrez appeler la fonction `verdict`, qui prend en argument un discours [2] et retourne un booléen [3]. Évidemment, vous essaierez de déranger le Premier ministre le moins possible.

Question 3 (2 points)

> Écrire une fonction qui prend en argument un discours et en retourne un sous-discours qui plaît au Premier ministre s'il existe et une liste vide sinon.

Question 4 (3 points)

> Écrire une fonction qui prend en argument un discours et en retourne le sous-discours (éventuellement vide) qui plaira le plus au Premier ministre.

Question 5 ($(8 - 3\log_n n_{\texttt{verdict}})$ points)

> Aujourd'hui, le Premier ministre est particulièrement occupé, et vous ne pouvez pas lui demander son `verdict` plus de n fois où n est le nombre de mots du discours initial [4].
> Écrire une fonction qui prend en argument un discours et en retourne le plus long extrait qui plaise au Premier ministre s'il existe et une liste vide sinon.

1. Un extrait d'un discours est donc un sous-discours, mais un sous-discours n'en est pas forcément un extrait.
2. Soit une liste de mots, soit un intervalle ou un ensemble de positions de mots dans le discours ; faites au plus simple selon votre langage.
3. Un entier si vous codez en C.

Il peut arriver que le Premier ministre *stabilosse*[5] des mots qui lui plaisent beaucoup dans la version finale et qu'il veut voir apparaître dans son discours, et qu'il *rature*[6] ceux qu'il ne veut voir apparaître sous aucun prétexte. Par exemple, si le mot `dissoudre` est raturé dans l'exemple page précédente, les seuls extraits possibles de longueur 3 seront `J'ai décidé`, `ai décidé de` et `l'assemblée nationale`.

Question 6 (3 points)

> On considère le cas où il n'y a que des ratures, aucun *stabilossage*. Écrire une fonction qui prend en argument un entier n ainsi que les positions des ratures et retourne le nombre d'extraits de longueur n obéissant aux contraintes du Premier ministre.

Question 7 (5 points)

> Écrire une fonction qui prend en argument un entier n ainsi que les positions des mots *stabilossés* et celles des mots raturés et retourne le nombre d'extraits de longueur n obéissant aux contraintes du Premier ministre.

Question 8 (4 points)

> À présent, le Premier ministre n'hésite maintenant pas à vous préciser que votre discours est « complètement nul[7] » lorsque c'est le cas. Comment déterminer efficacement, à partir d'un discours complètement nul, un extrait pouvant boucler (c'est-à-dire, en s'autorisant à passer au premier mot du discours après le dernier mot) ?

Vous pouvez attaquer les questions suivantes si et seulement si votre appréciation du sujet est pour l'instant strictement positive.

Question bonus 9 (3 points)

On considère maintenant une grille $m \times n$ de mots sur une feuille. On dit qu'un sous-discours est *rectangulaire* s'il correspond à un rectangle de mots inclus dans la feuille. Si le Premier ministre est très occupé, il ne prend pas même pas le temps de lire et vous demande de lui indiquer un sous-discours rectangulaire sur lequel il effectue son algorithme. Comment déterminer un sous-discours rectangulaire qui plaît au Premier ministre en allant le moins de fois possible le solliciter dans son bureau ?

Question bonus 10 (5 points)

Que se passe-t-il si on considère des mots stabilossés et raturés dans une grille $m \times n$? Combien y a-t-il de sous-discours rectangulaires $p \times q$ respectant les contraintes ?

Le sujet est sur 27 points, et les questions bonus rapportent au total 8 points, plus 1 point de présentation.

4. Si vraiment vous insistez, sa secrétaire vous laissera rentrer dans son bureau n^2 fois, mais vous obtiendrez moins de points à cette question. De même, il est déconseillé de déranger le Premier ministre n^3 fois.
5. Subjonctif présent du verbe *stabilosser*, signifiant « surligner au Stabilo Boss® ».
6. Subjonctif présent du verbe *raturer*, signifiant « annuler un mot en tirant un trait dessus ».
7. Comprendre : d'appréciation zéro.