Retour sur trace

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve: 3 heures 30 minutes

Juin/Juillet 2014

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Sur votre table est indiqué un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion!). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures!
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple: $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***. Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

1 Introduction

Le but de ce sujet est d'étudier des algorithmes de recherche de solutions pour divers problèmes (le problème des n dames, le problème du cavalier, ou la recherche de règles de Golomb optimales). Tous les algorithmes vus ici appartiennent à la famille des algorithmes de retour sur trace (ou *backtracking* en anglais).
Les différents problèmes abordés étant indépendants, les parties correspondantes du sujet le sont aussi et peuvent donc être traitées dans n'importe quel ordre. Il est à noter cependant que certaines instances des problèmes à résoudre peuvent demander de **longs temps de calcul**. Ces instances sont identifiées par des mentions « difficile » ou « très difficile ». Il est recommandé aux candidat·e·s de **traiter d'abord les autres questions du sujet** avant de s'attaquer à ces instances difficiles qui requièrent une certaine finesse tant dans la conception algorithmique que dans l'optimisation de l'implantation.
Enfin, puisque certaines questions peuvent demander un long temps de calcul (minutes ou dizaines de minutes, selon la qualité de votre implantation), **n'hésitez pas à laisser votre programme s'exécuter pendant que vous commencez à préparer les questions suivantes**. Vous pouvez aussi profiter des deux cœurs disponibles sur le processeur de la machine sur laquelle vous travaillez pour **lancer jusqu'à deux programmes en parallèle** du moment que ceux-ci tiennent en mémoire.

Dans la suite du sujet, pour tous a, $b \in \mathbb{Z}$, $b > 0$, on désigne par $a \bmod b$ l'unique représentant dans $\{0, \dots, b-1\}$ de la classe d'équivalence de a modulo b.
On définit alors par récurrence les suites

$$u_{k+1} = 7951 \times u_k \bmod 123457 \text{ et}$$
$$v_{k+1} = 2971 \times v_k \bmod 345679,$$

avec $v_0 = u_0$ pour initier la récurrence. La valeur de u_0 vous est donnée sur votre table ; vous devez impérativement la reporter sur votre fiche réponse. Afin de faciliter les vérifications de vos programmes, les résultats attendus vous sont donnés pour un autre u_0, noté $\widetilde{u_0}$.
On définit de plus la signature d'un m-uplet $s = (s_1, \dots, s_m) \in \mathbb{Z}^m$ comme suit :

$$\mathsf{Sig}(s) = \sum_{i=1}^{m} (u_i \times s_i \bmod v_i)\,.$$

Question 1 *Calculez la signature de*
a) $(1, 2, 3, 4, 5)$, ***b)*** $(6, 7, 8, 9, 10)$, ***c)*** $(11, 12, 13, 14, 15)$.

2 Recherche de solutions

2.1 Algorithme de backtracking

Soit S un ensemble, dont les éléments sont appelés états. On se donne un certain $s^{(0)} \in S$. On suppose de plus donnés deux prédicats, $\mathsf{EstComplet}$ et $\mathsf{EstCorrect}$, tous deux fonctions de S dans $\{\mathsf{vrai}, \mathsf{faux}\}$:

— un état $s \in S$ est dit complet lorsque $\mathsf{EstComplet}(s) = \mathsf{vrai}$, et incomplet sinon ;

— un état $s \in S$ est dit correct lorsque $\mathsf{EstCorrect}(s) = \mathsf{vrai}$, et incorrect sinon ; et
— un état $s \in S$ est dit solution lorsqu'il est à la fois complet et correct, c'est-à-dire lorsque $\mathsf{EstComplet}(s) = \mathsf{EstCorrect}(s) = \mathsf{vrai}$.

Typiquement (mais pas forcément), un état complet est dans $\mathbb{Z}^m$ et un état incomplet est dans $(\{\bot\} \cup \mathbb{Z})^m \setminus \mathbb{Z}^m$. En particulier, tout état solution est complet. Par contre, un état incomplet peut être incorrect.

On se donne aussi f, une application de S dans les parties de S, qui à chaque état s associe un ensemble d'états $f(s)$. Un état $s' \in f(s)$ diffère usuellement de s par le fait qu'un élément (disons le i-ième) de s est passé de $\bot$ à $s'_i \in \mathbb{Z}$; on dit que la i-ième variable de s a été instanciée.

Lorsqu'il existe une suite $s^{(1)}, \ldots, s^{(k)}$ d'éléments de S telle que $s^{(i+1)} \in f(s^{(i)})$ pour tout $1 \leq i < k$, on dit que $s^{(1)}$ est un ancêtre de $s^{(k)}$. Cette suite, appelée chemin de $s^{(1)}$ à $s^{(k)}$, n'est pas nécessairement unique. Lorsque k est minimal parmi tous les chemins de $s^{(1)}$ à $s^{(k)}$, on l'appelle longueur du chemin de $s^{(1)}$ à $s^{(k)}$. On parlera enfin de condition d'unicité pour se référer au fait que, pour tous s et $s' \in S$ tels que s est un ancêtre de s', il existe un unique chemin de s à s'.

On supposera aussi que l'application f est consistante, c'est-à-dire que
— si s est incorrect, alors tout $s' \in f(s)$ est aussi incorrect ; et
— pour tout $s \in S$, $s^{(0)}$ est un ancêtre de s.

On appellera profondeur de f la longueur maximale d'un chemin de $s^{(0)}$ à s, pour $s \in S$. Enfin, les états complets de S sont les $s \in S$ tels que $f(s)$ est vide.

On se donne alors la fonction récursive Backtrack suivante :

```
fonction Backtrack(s ∈ S) :
    si EstCorrect(s) = vrai alors :
        si EstComplet(s) = vrai alors :
            afficher s
        sinon :
            pour tout s' ∈ f(s) faire :
                Backtrack(s')
```

Question à développer pendant l'oral : Que fait $\mathsf{Backtrack}(s^{(0)})$ lorsque f est bien consistante ? Toutes les hypothèses faites dans la définition de la consistance sont-elles nécessaires pour cela ?

Question à développer pendant l'oral : Quelle est la complexité en temps comme en mémoire de cette fonction ? La condition d'unicité a-t-elle un impact ? Si oui, lequel ?

2.2 Heuristiques

De nombreuses heuristiques permettent de rendre l'algorithme plus rapide. Ainsi :
— Modifier l'application f afin d'avoir la condition d'unicité.
— Si le programme est sûr qu'un état n'est l'ancêtre d'aucune solution, il est préférable de modifier EstCorrect de sorte à déclarer cet état incorrect.
— Souvent, $f(s)$ contient des états s' qui diffèrent de s par le fait qu'une variable a été instanciée. Il est parfois préférable d'instancier une variable qui a peu de choix

possibles. Il est parfois aussi préférable de l'instancier de sorte à avoir le moins possible de variables ayant plusieurs choix possibles.

Afin de résoudre les problèmes énoncés dans les parties suivantes du sujet, il vous faudra très probablement adapter et mettre en œuvre certaines de ces heuristiques. Vous veillerez à justifier vos choix et leur impact sur la complexité de l'algorithme de recherche de solutions.

3 Cas avec contraintes

3.1 Problème des n dames

Nous considérons ici le problème des n dames qui, pour un entier $n > 0$ donné, revient à placer n dames sur un échiquier de $n \times n$ cases sans que les dames ne puissent s'attaquer mutuellement, conformément aux règles de déplacement de cette pièce au jeu d'échecs. En d'autres termes, il ne peut jamais y avoir deux dames sur une même ligne horizontale, verticale ou diagonale.

Plus formellement, les états complets de S forment l'ensemble des n-uplets $(s_1, \ldots, s_n) \in \{0, 1, \ldots, n-1\}^n$. Un état complet $s \in S$ est solution si et seulement si, pour tous i, $j \in \{1, \ldots, n\}$,

- $i \neq j \Rightarrow s_i \neq s_j$,
- $i \neq j \Rightarrow s_i + j \neq s_j + i$, et
- $i \neq j \Rightarrow s_i - j \neq s_j - i$.

Question à développer pendant l'oral : Décrivez un algorithme permettant de trouver, pour un n donné, toutes les solutions du problème des n dames. Détaillez sa complexité en temps et en mémoire, ainsi que les heuristiques que vous avez éventuellement utilisées.

Question 2 *Pour chacune des valeurs de n suivantes, quel est le nombre de solutions s du problème des n dames dont la signature $\mathsf{Sig}(s)$ est paire ?*
a) $n = 4$, ***b)*** $n = 5$, ***c)*** $n = 6$, ***d)*** $n = 7$, ***e)*** $n = 8$, ***f)*** $n = 9$.

Question 3 *Pour chacune des valeurs de n suivantes, quel est le nombre de solutions s du problème des n dames dont la signature $\mathsf{Sig}(s)$ est paire ?*
a) $n = 10$, ***b)*** $n = 11$, ***c)*** $n = 12$, ***d)*** $n = 13$, ***e)*** $n = 14$, ***f)*** $n = 15$.

Question 4 *Pour chacune des valeurs de n suivantes, quel est le nombre de solutions s du problème des n dames dont la signature $\mathsf{Sig}(s)$ est paire ?*
a) $n = 16$ *(difficile)*, ***b)*** $n = 17$ *(difficile)*, ***c)*** $n = 18$ *(très difficile)*.

3.2 Problème du cavalier

Pour un entier $n > 0$ donné, le problème du cavalier revient à déplacer un cavalier sur toutes les cases d'un échiquier de taille $n \times n$ sans passer deux fois par la même case, tout en suivant les déplacements légaux de cette pièce au jeu d'échecs (c'est-à-dire, en « L »).

Pour ce problème, les éléments complets de S sont les m-uplets $(s_1, \ldots, s_m) \in \{0, \ldots, n-1\}^m$ avec $m = 2n^2$: un tel m-uplet représente les n^2 positions successives du cavalier sur

l'échiquier, les variables d'indice impair représentant le numéro de ligne (de 0 à $n-1$) et les variables d'indice pair le numéro de colonne.
Un état complet $s \in S$ est solution si et seulement si

— pour tout $i \in \{0, \ldots, n^2-2\}$, $|s_{2i+3} - s_{2i+1}| \times |s_{2i+4} - s_{2i+2}| = 2$, et
— pour tous i, $j \in \{0, \ldots, n^2-1\}$, $i \neq j \Rightarrow (s_{2i+1}, s_{2i+2}) \neq (s_{2j+1}, s_{2j+2})$.

Dans un premier temps, nous ne cherchons pas à énumérer toutes les solutions de ce problème, mais seulement à trouver la première solution suivant l'ordre lexicographique sur $\{0, \ldots, n-1\}^m$.

Question à développer pendant l'oral : Décrivez un algorithme permettant de trouver, pour un n donné, la première solution (suivant l'ordre lexicographique) du problème du cavalier sur un échiquier $n \times n$. Détaillez sa complexité en temps et en mémoire, ainsi que les heuristiques que vous avez éventuellement utilisées.

Question 5 *Pour chacune des valeurs de n suivantes, quelle est la signature de la première solution (suivant l'ordre lexicographique) du problème du cavalier dans un échiquier $n \times n$?*
a) $n = 5$, ***b)*** $n = 6$, ***c)*** $n = 7$, ***d)*** $n = 8$, ***e)*** $n = 9$.

Un problème très proche est le problème du tour du cavalier, qui consiste à faire en sorte que le cavalier revienne à sa position de départ et effectue donc une boucle passant par toutes les cases de l'échiquier. Comme pour le problème précédent, les états complets seront aussi des m-uplets de $\{0, \ldots, n-1\}^m$ avec $m = 2n^2$, et les solutions devront vérifier les mêmes contraintes que pour le problème du cavalier, avec la contrainte supplémentaire que $|s_1 - s_{m-1}| \times |s_2 - s_m| = 2$.

Question à développer pendant l'oral : Montrez que, lorsque l'on cherche à trouver la première solution (suivant l'ordre lexicographique) du problème du tour du cavalier dans un échiquier $n \times n$, si une telle solution s existe, alors les valeurs de ses quatre premières variables s_1, s_2, s_3 et s_4 sont déjà connues. Donnez ces valeurs.

Question 6 *Pour chacune des valeurs de n suivantes, quelle est la signature de la première solution (suivant l'ordre lexicographique) du problème du tour du cavalier dans un échiquier $n \times n$?*
a) $n = 6$, ***b)*** $n = 8$ *(difficile).*

Nous cherchons désormais à énumérer toutes les solutions du problème du cavalier.

Question à développer pendant l'oral : Décrivez un algorithme permettant de trouver, pour un n donné, toutes les solutions du problème du cavalier sur un échiquier $n \times n$. Détaillez sa complexité en temps et en mémoire, ainsi que les heuristiques que vous avez éventuellement utilisées.

Question 7 *Pour chacune des valeurs de n suivantes, quel est le nombre de solutions s du problème du cavalier dans un échiquier $n \times n$ dont la signature $\mathsf{Sig}(s)$ est paire ?*
a) $n = 5$, ***b)*** $n = 6$ *(très difficile).*

4 Cas avec critère à optimiser

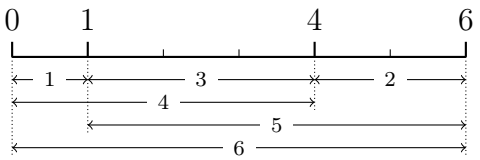
Dans certains cas, on dispose d'un critère c, application de S dans $\mathbb{Z}$, que l'on souhaite optimiser. Au lieu de chercher toutes les solutions d'un problème, on cherche alors les solutions $s \in S$ telles que $c(s)$ soit minimal. Ces solutions sont appelées solutions optimales.

Question à développer pendant l'oral : Comment modifier la fonction Backtrack afin de n'obtenir que les solutions optimales ?

Question à développer pendant l'oral : Supposons que le critère c vérifie que $c(s') > c(s)$ pour tout $s' \in f(s)$. Comment améliorer l'algorithme pour tirer parti de cette propriété ?

4.1 Règles de Golomb

Étant donné un entier $n > 0$, une règle de Golomb d'ordre n est un n-uplet d'entiers distincts $s_1 < s_2 < \cdots < s_n$ (aussi appelés marques) tels que les distances $s_i - s_j$ entre chaque paire de marques soient toutes distinctes. Comme une règle est invariante par translation, nous supposerons dans la suite, sans perte de généralité, que la plus petite marque est toujours $s_1 = 0$. La plus grande marque, s_n, est appelée longueur de la règle. De plus, comme le symétrique $(0, s_n - s_{n-1}, \ldots, s_n - s_2, s_n)$ d'une règle de Golomb $(0, s_2, \ldots, s_n)$ est une autre règle de même longueur, nous imposons la contrainte suivante : $s_2 - s_1 < s_n - s_{n-1}$. Enfin, pour un n donné, une règle de Golomb d'ordre n de longueur minimale est dite optimale.
Par exemple, la règle $(0, 1, 4, 6)$, illustrée ci-dessous, est une règle de Golomb optimale d'ordre 4 et de longueur 6 :



Question à développer pendant l'oral : Construisez une règle de Golomb optimale d'ordre $n = 5$.

Question à développer pendant l'oral : Donnez des bornes inférieure et supérieure sur la longueur d'une règle de Golomb optimale d'ordre n.

Un corollaire immédiat d'une conjecture d'Erdős affirme qu'il existe une règle de Golomb d'ordre n de longueur strictement inférieure à n^2 pour tout entier $n > 0$. Cette conjecture a été démontrée pour $n \leq 65000$, c'est-à-dire bien au delà des valeurs de n considérées dans cet énoncé.
Afin d'utiliser l'algorithme de backtracking pour chercher des règles de Golomb optimales d'ordre n donné, nous définissons les états complets de S comme les n-uplets $(s_1, \ldots, s_n) \in \{0, \ldots, n^2 - 1\}^n$ tels que

- $s_1 = 0$,
- $s_i < s_{i+1}$ pour tout $i \in \{1, \ldots, n-1\}$, et

— $s_2 - s_1 < s_n - s_{n-1}$.

Un état complet $s \in S$ est solution si et seulement si, pour tous i, j, k et $l \in \{1, \ldots, n\}$ avec $i \neq j$, on a $(i, j) \neq (k, l) \Rightarrow s_i - s_j \neq s_k - s_l$. Enfin, on cherche à optimiser le critère $c(s) = s_n$.

Question à développer pendant l'oral : Décrivez un algorithme permettant de trouver, pour un n donné, toutes les règles de Golomb optimales d'ordre n. Détaillez sa complexité en temps et en mémoire, ainsi que les heuristiques que vous avez éventuellement utilisées.

Question 8 *Pour chacune des valeurs de n suivantes, quelle est la signature minimale d'une règle de Golomb optimale d'ordre n ?*
a) $n = 5$, ***b)*** $n = 6$, ***c)*** $n = 7$, ***d)*** $n = 8$.

Question 9 *Pour chacune des valeurs de n suivantes, quelle est la signature minimale d'une règle de Golomb optimale d'ordre n ?*
a) $n = 9$, ***b)*** $n = 10$, ***c)*** $n = 11$, ***d)*** $n = 12$.

Question 10 *Pour chacune des valeurs de n suivantes, quelle est la signature minimale d'une règle de Golomb optimale d'ordre n ?*
a) $n = 13$ *(difficile)*, ***b)*** $n = 14$ *(très difficile)*.

4.2 Retour sur les n dames

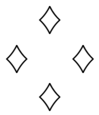
Pour de plus grandes instances du problème des n dames, nous ne souhaitons plus énumérer toutes les solutions, mais seulement trouver la solution s qui a la plus petite signature $\mathsf{Sig}(s)$. Nous reprenons donc la formalisation du problème vue à la Section 3.1, et y rajoutons juste le critère à optimiser $c(s) = \mathsf{Sig}(s)$.
Il est à noter que plusieurs solutions distinctes peuvent être optimales et ainsi avoir une signature de valeur minimale. Cela n'a pas importance ici, car il vous est demandé dans les questions suivantes de ne renvoyer que la signature d'une solution optimale (qui sera la même quelle que soit la solution optimale retenue), et non pas la solution proprement dite.

Question à développer pendant l'oral : Décrivez un algorithme permettant de trouver, pour un n donné, la signature minimale d'une solution du problème des n dames. Détaillez sa complexité en temps et en mémoire, ainsi que les heuristiques que vous avez éventuellement utilisées.

Question 11 *Pour chacune des valeurs de n suivantes, quelle est la signature minimale d'une solution du problème des n dames ?*
a) $n = 18$, ***b)*** $n = 19$, ***c)*** $n = 20$ *(difficile)*.

Question à développer pendant l'oral : Expliquez en quoi il peut être intéressant de choisir dans quel ordre les variables s_i sont instanciées. Déterminez un ordre adapté au problème considéré ici. Justifiez votre choix et modifiez votre algorithme en conséquence.

Question 12 *Pour chacune des valeurs de n suivantes, quelle est la signature minimale d'une solution du problème des n dames ?*
a) $n = 24$ *(difficile),* ***b)*** $n = 25$ *(difficile),* ***c)*** $n = 26$ *(difficile).*



Fiche réponse type: Retour sur trace

$\widetilde{\mathbf{u_0}}$ **:** 42

Question 1

a) 494521

b) 603454

c) 380468

Question 2

a) 2

b) 4

c) 2

d) 18

e) 46

f) 172

Question 3

a) 354

b) 1364

c) 7054

d) 36827

e) 183136

f) 1139718

Question 4

a) 7386785

b) 47895896

c) 333057792

Question 5

a) 3233482

b) 5165614

c) 7809977

d) 9718686

e) 11592563

Question 6

a) 5484206

b) 9791466

Question 7

a) 868

b) 3319662

Question 8

a) 505572

b) 496128

c) 660597

d) 862537

Question 9

a) 521065

b) 568983

c) 1042450

d) 962562

Question 10

a) 830097

b) 1083204

Question 11

a) 321923

b) 340450

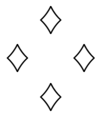
c) 355476

Question 12

a) 376533

b) 364227

c) 366480



Fiche réponse: Retour sur trace

Nom, prénom, u_0: .....

Question 1

a)

b)

c)

Question 2

a)

b)

c)

d)

e)

f)

Question 3

a)

b)

c)

d)

e)

f)

Question 4

a)

b)

c)

Question 5

a)

b)

c)

d)

e)

Question 6

a)

b)

Question 7

a)

b)

Question 8

a)

b)

c)

d)

Question 9

a)

b)

c)

d)

Question 10

a)

b)

Question 11

a)

b)

c)

Question 12

a)

b)

c)