Recherche de motifs

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

Durée de l'épreuve: 3 heures 30 minutes

Juin/Juillet 2010

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 Génération de phrases

Considérons la suite d'entiers $(u_k)_k$ définie par

$$u_{k+1} = 15091 \times u_k \mod 64007.$$

On s'assurera de précalculer et stocker suffisamment de valeurs de u_k de manière à pouvoir y accéder en temps constant par la suite.

Une phrase P_n est une séquence de n lettres $(\alpha_{k+1})_{0 \leq k < n}$ prises dans l'alphabet $\mathcal{A} = \{A, B, C, D, E\}$. La lettre α_k est choisie en fonction de la valeur $v_k = u_k \mod 16$:

$$\alpha_k = \begin{cases} A & \text{si } v_k = 0 \\ B & \text{si } 1 \leq v_k < 2 \\ C & \text{si } 2 \leq v_k < 4 \\ D & \text{si } 4 \leq v_k < 8 \\ E & \text{si } v_k \geq 8 \end{cases}$$

La phrase Q_n est la séquence de n lettres $(\alpha_{k+2})_{0 \leq k < n}$.

Question 1 *Quelles sont les phrases* ***a)*** P_5 *et* ***b)*** Q_5 *?*

Dans la suite de cette épreuve, la séquence $(m_i)_{0 \leq i < l}$ désignera le motif et la séquence $(t_i)_{0 \leq i < n}$ désignera le texte où ce motif est recherché. Une fenêtre à la position i est une portion du texte $(t_j)_{i \leq j < i+l}$ ayant la longueur du motif. Un algorithme de recherche va déplacer une fenêtre le long du texte jusqu'à ce que le contenu de la fenêtre corresponde exactement au motif : un motif m de longueur l se trouve à la position i $(0 \leq i < n - l)$ d'un texte t si on a $t_{i+j} = m_j$ pour tout j tel que $0 \leq j < l$.
Le nombre d'occurrences d'un motif dans un texte est le nombre de positions où se trouve ce motif. En particulier, il est possible que des occurrences se recouvrent partiellement. Par exemple, le motif $ABAB$ apparaît deux fois dans le texte $ABABAB$.

Question 2 *Combien y a-t-il d'occurrences du motif ECEEDEED dans les textes* ***a)*** P_{10000}, ***b)*** P_{20000} *et* ***c)*** P_{30000} *?*

L'objectif des algorithmes de recherche présentés ci-dessous sera de compter le nombre d'occurrences d'un motif dans un texte, tout en essayant de minimiser le nombre de lettres lues dans ce texte. Les motifs considérés seront $ECEEDEED$ et $EDDCEED$.

2 Coût de recherche

Un algorithme de recherche va généralement lire des lettres du texte et les comparer à celle du motif. Potentiellement, certaines lettres du texte ne seront pas lues du tout lors de la recherche d'un motif particulier. On s'intéresse au nombre de positions accédées dans le texte. Si une lettre est lue une ou plusieurs fois à la position i du texte, alors cette position est considérée comme accédée.

Question 3 *Considérons un algorithme qui accède aux k positions d'indices $(u_i \mod k)_{1 \leq i \leq k}$. Combien de positions différentes sont accédées pour les valeurs suivantes ?* ***a)*** $k = 10$, ***b)*** $k = 100$, ***c)*** $k = 1000$, ***d)*** $k = 10000$.

Le coût de recherche est défini comme le nombre de positions différentes accédées pour compter le nombre d'occurrences d'un motif dans un texte.
Considérons un algorithme naïf R_1 qui, pour chaque entier i entre 0 et $n - l$, teste si la fenêtre à la position i correspond au motif. Ce test s'effectue en comparant de droite à gauche les lettres de la fenêtre et du motif. Si une des lettres ne correspond pas, la comparaison s'arrête (les lettres plus à gauche sont ignorées), la fenêtre est décalée d'un cran vers la droite, et la comparaison reprend à la lettre la plus à droite de la fenêtre.

Question à développer pendant l'oral : Justifier que, dans le meilleur des cas, le nombre de lettres accédées par l'algorithme R_1 est $n - l + 1$. Quel est ce cas ?

Question 4 *Quel est le coût de la recherche des motifs suivants dans le texte Q_{10000} ?* ***a)*** P_{2000}, ***b)*** P_{5000}, ***c)*** P_{8000} *?*

3 Accélération par la dernière lettre

Si la dernière lettre ω de la fenêtre n'apparaît pas en avant-dernière position $(l - 2)$ du motif, la décaler d'un cran vers la droite est inefficace. En effet, la nouvelle fenêtre ne peut pas contenir le motif ; lancer une comparaison ne sert donc à rien. La position i de la fenêtre aurait pu directement être augmentée de 2 lors de la recherche des occurrences. Similairement, si ω n'apparaît ni en position $l - 2$ ni en position $l - 3$, la position de la fenêtre peut être incrémenté de 3 : aucune des fenêtres intermédiaires ne peut contenir le motif. Et ainsi de suite.
Considérons donc un algorithme R_2, variante de R_1 qui déplace la fenêtre en fonction de la dernière lettre contenue dans celle-ci.

Question 5 *Supposons que le motif recherché soit P_{100} et que la fenêtre en position i ait déjà été traitée. Quel est le plus grand incrément de i que l'on peut appliquer sans risquer de rater une occurrence si la lettre du texte en position $i + 99$ est* ***a)*** *un A ?* ***b)*** *un B ?* ***c)*** *un C ?* ***d)*** *un D ?* ***e)*** *un E ?*

Question à développer pendant l'oral : Quelle complexité temporelle a la création d'une table contenant ces valeurs en fonction de la taille de l'alphabet et de la taille du motif ? Peut-on l'améliorer ?

Question 6 *Quel est le coût de la recherche des motifs suivants dans la phrase P_{30000} pour l'algorithme R_2 ?* ***a)*** *ECEEDEED*, ***b)*** *EDDCEED*.

Question à développer pendant l'oral :
- Pour un texte donné et une longueur de motif l donnée, certains motifs favorisent particulièrement l'algorithme R_2, c'est-à-dire qu'ils ont un faible coût de recherche. Donner des exemples de tels couples motif/texte. Quel est alors le coût de la recherche en fonction de la taille du texte et de la taille du motif ?
- Pour ces couples motif/texte particuliers, qu'en est-il de la complexité temporelle totale de R_2 (création de la table puis nombre de comparaisons de lettres lors de la recherche des occurrences) ?

Les valeurs stockées dans la table sont correctes et optimales. Elles sont correctes parce qu'en effectuant les incréments correspondants, on ne risque jamais de dépasser un motif contenu dans un texte. Elles sont optimales parce que des incréments plus grands seraient incorrects. Par exemple, si la table ne contenait que des 1 (ce qui revient à l'algorithme R_1), toutes les valeurs seraient correctes mais une seule d'entre elles au plus serait optimale. On dit d'un incrément qu'il est approché si sa valeur est correcte mais pas forcément optimale.

Question à développer pendant l'oral : Quelles approximations peut-on introduire dans les valeurs de la table pour réduire la complexité temporelle de sa création tout en limitant le surcoût de la recherche ?

4 Accélération par le plus long suffixe

Repartons de l'algorithme R_1 et calculons les déplacements de la fenêtre d'une autre façon. Supposons que la comparaison du motif et de la fenêtre ait passé en revue k lettres avant de s'arrêter. Autrement dit, les $k-1$ dernières lettres de la fenêtre correspondent. Par contre, celle en position $l-k$ ne correspond pas (si k est strictement plus petit que la longueur l du motif). Dans ces conditions, déplacer la fenêtre de δ lettres n'est utile que si les $k-1$ dernières lettres déjà connues se trouvent aux positions $l-\delta-k+1 \ldots l-\delta-1$ du motif et si la lettre en position $l-k$ ne se trouve pas en position $l-\delta-k$. Dans ce qui précède, les positions négatives dans le motif sont ignorées.

Par exemple, si le motif est *CCBABABA* et si $k=3$, la fenêtre termine par xBA avec $x \neq A$. La position i de la fenêtre peut donc être incrémentée de $\delta = 4$ pour aligner xBA avec CBA. Les incréments plus petits ne respectent pas la présence des lettres BA ($\delta = 1, 3$) ou la contrainte $x \neq A$ ($\delta = 2$). Remarque : l'incrément $\delta = 4$ n'est optimal que si $x = C$. Prendre en compte la valeur précise de la dernière lettre lue sera le sujet de la Section 5. Pour l'instant, la seule chose qui importe est qu'elle est différente d'une certaine valeur.

Pour traiter le cas où toutes les lettres de la fenêtre ont été comparées, on prendra comme convention que, si $k = l$, la première lettre de la fenêtre ne correspond pas, et si $k = l+1$, la fenêtre contient le motif.

Question 7 *Supposons que le motif recherché soit P_{100} et que la fenêtre en position i ait déjà été traitée. Quel est le plus grand incrément de i que l'on peut appliquer sans risquer de rater une occurrence quand le nombre de lettres comparées a été* **a)** $k = 1$? **b)** $k = 2$? **c)** $k = 3$? **d)** $k = 4$? **e)** $k = 99$?

Question à développer pendant l'oral :
- Quelle complexité temporelle a la création de la table contenant tous les incréments pour $0 \leq k \leq l$ en fonction de la taille l du motif ? Peut-on l'améliorer ?
- Quelles approximations peut-on introduire dans les valeurs de la table pour réduire la complexité temporelle de sa création ?

Question 8 *Quel est le coût de la recherche des motifs suivants dans le texte P_{30000} pour un algorithme R_3 utilisant cette deuxième table ?* **a)** *ECEEDEED*, **b)** *EDDCEED*.

Les algorithmes R_2 et R_3 sautent tous deux les fenêtres pour lesquels ils sont sûrs qu'elles ne contiennent pas le motif. Il est possible de les combiner en un algorithme R_4 qui choisit toujours le plus grand des décalages proposés par R_2 et R_3 pour incrémenter la position de la fenêtre.

Question 9 *Quel est le coût de la recherche des motifs suivants dans le texte P_{30000} pour l'algorithme R_4 utilisant les deux tables à la fois ?* **a)** *ECEEDEED*, **b)** *EDDCEED*.

5 Automate de recherche

Il existe d'autres façons de combiner les approches R_2 et R_3. L'une d'elle est l'utilisation d'une machine à états. À chaque étape, cette machine lit une lettre à une certaine position de la fenêtre. En fonction de son état courant et de cette lettre, elle choisit alors quel sera son prochain état, de combien déplacer la fenêtre, et à quelle position dans la fenêtre lire la prochaine lettre.
En d'autres termes, cette machine est caractérisée par un ensemble fini S d'états et une fonction de transition $\delta : S \times \mathcal{A} \to S \times \mathbb{N} \times [0, l-1] \times \mathbb{B}$. Cette fonction prend en entrée un état et une lettre ; elle renvoie un état, un décalage de fenêtre, une position dans la fenêtre, et un booléen indiquant si la fenêtre courante (avant décalage) contient le motif entier.
Ainsi, si la fenêtre est en position i et si la fonction de transition renvoie (s, δ, j, V), alors le motif a été trouvé en position i du texte, la prochaine fenêtre sera en $i' = i + \delta$, la position absolue de la prochaine lettre lue sera $i' + j$.
Dans un premier temps, un état de la machine représentera le nombre de lettres à la fin du motif dont on sait qu'elles correspondent au contenu de la fenêtre courante. Il y aura donc l états avec l la longueur du motif. La machine partira de l'état 0 (pas d'information sur le contenu de la fenêtre courante, donc 0 lettres en correspondance) avec une fenêtre en position 0 et la première lettre sera lue en position $l - 1$.
Remarque : cette machine à états sert à compter des occurrences ; en particulier elle ne doit pas s'arrêter quand elle trouve un motif. C'est pourquoi un $l + 1$-ème état qui signifierait que les l lettres de la fenêtre correspondent à celle du motif ne sert à rien. Cette information est fournie par la fonction de transition et la fenêtre est nécessairement déplacée en sortant de l'état $l - 1$.

Question 10 *Supposons que le motif recherché soit P_{100} et que l'état courant de la machine soit 2 (les 2 dernières lettres de la fenêtre correspondent au motif). Quel est le résultat de la fonction de transition quand la lettre lue est* **a)** *un A ?* **b)** *un B ?* **c)** *un C ?* **d)** *un D ?* **e)** *un E ?*

Question 11 *Quel est le coût de la recherche des motifs suivants dans la phrase P_{30000} pour un algorithme R_5 utilisant cette machine à états ?* **a)** *ECEEDEED*, **b)** *EDDCEED*.

6 États à deux blocs

La machine à états R_5 se sert de l'information des k dernières lettres de la fenêtre uniquement pour incrémenter la position i de la fenêtre. Une fois celle-ci incrémentée, l'informa-

tion est perdue puisqu'elle n'est pas stockée dans les états. Par conséquent, la machine relira plusieurs fois les mêmes lettres.
Pour éviter ce problème, on peut concevoir une machine R_6 utilisant un plus grand nombre d'états. Un état sera maintenant un triplet d'entiers (p, q, r) avec $0 \leq p < q \leq r$. Ces entiers représentent deux blocs de lettres du motif dont on sait qu'elles sont dans les positions correspondantes de la fenêtre courante. Le premier bloc s'étend des positions 0 à $p-1$, le second de q à $r-1$. Par convention, on prendra $p = 0$ si le bloc de gauche n'est pas connu et $q = r = l$ si le bloc de droit ne l'est pas.
La machine part de l'état $(0, l, l)$ avec une fenêtre en position 0 et la première lettre sera lue en position $l-1$. La fonction de transition est construite de telle sorte que, à chaque étape, la position de la prochaine lettre lue soit r si $r < l$ et $q-1$ sinon, c'est-à-dire que l'on cherche à faire grossir le bloc de droite, de préférence du côté de la fin de la fenêtre.

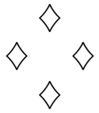
Question à développer pendant l'oral : Justifier les propriétés suivantes :
– Il est possible d'écrire la fonction de transition de telle sorte qu'aucune lettre connue du texte ne se trouve en dehors des blocs $0 \dots p-1$ et $q \dots r-1$ pour la fenêtre courante.
– Chaque lettre du texte n'est parcourue qu'une seule fois par R_6.

Question 12 *Quel est le coût de la recherche des motifs suivants dans la phrase P_{30000} pour l'algorithme R_6 utilisant des états à deux blocs ?* ***a)*** *ECEEDEED,* ***b)*** *EDDCEED.*

La création de la fonction de transition étant un processus coûteux, il est important de se limiter aux états qui sont effectivement atteignables étant donné un motif. Par exemple, les états de la forme (p, q, r) avec $p \geq 2$ sont impossibles à atteindre pour le motif $ABCDE$. Ils sont donc inutiles.

Question 13 *Combien y a-t-il d'états utiles pour les motifs suivants ?* ***a)*** P_{10}, ***b)*** P_{20}, ***c)*** P_{30} *?*

Question à développer pendant l'oral : Comment effectuer la génération de la machine à états pour éviter de construire les états inutiles ?



Fiche réponse type: Recherche de motifs

$\widetilde{u_0}$: 178

Question 1

a) *DECCE*

b) *ECCEE*

Question 2

a) 3

b) 6

c) 9

Question 3

a) 7

b) 61

c) 634

d) 6586

Question 4

a) 8004

b) 5008

c) 2001

Question 5

a) 2

b) 51

c) 1

d) 5

e) 3

Question 6

a) 13684

b) 14739

Question 7

a) 1

b) 3

c) 11

d) 100

e) 100

Question 8

a) 15193

b) 18701

Question 9

a) 11041

b) 13857

Question 10

a) (3, 0, 96, F)

b) (0, 100, 99, F)

c) (0, 11, 99, F)

d) (0, 52, 99, F)

e) (0, 45, 99, F)

Question 11

a) 9964

b) 13432

Question 12

a) 9789

b) 11282

Question 13

a) 55

b) 176

c) 442