ÉCOLE POLYTECHNIQUE

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2007

FILIÈRE MP - OPTION PHYSIQUE ET SCIENCES DE L'INGÉNIEUR **FILIÈRE PC**

COMPOSITION D'INFORMATIQUE

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.
On attachera une grande importance à la concision, à la clarté, et à la précision de la rédaction.

⋆ ⋆ ⋆

Compression bzip

Le temps d'exécution $T(f)$ d'une fonction f est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc.) nécessaire au calcul de f. Lorsque ce temps d'exécution dépend d'un paramètre n, il sera noté $T_n(f)$. On dit que la fonction f s'exécute :

en temps $O(n^\alpha)$, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leq Kn^\alpha$.

Dans ce sujet, il sera question de l'algorithme de Burrows-Wheeler qui compresse très efficacement des données textuelles. Le texte d'entrée à compresser sera représenté par un tableau t contenant des entiers compris entre 0 et 255 inclus.

1 Compression par redondance

La compression par redondance compresse un texte d'entrée qui possède des répétitions consécutives de lettres (ou d'entiers dans notre cas). Dans un premier temps, on calcule les fréquences d'apparition de chaque entier dans le texte d'entrée. Puis on compresse le texte.

Question 1 Écrire la fonction $\mathsf{occurrences}(t, n)$ qui prend en argument un tableau d'entrée t de longueur n ; et qui retourne un tableau r de taille 256 tel que $r[i]$ est le nombre d'occurrences de i dans t pour $0 \leq i < n$.

Question 2 Écrire la fontion $\mathsf{min}(t, n)$ qui prend en argument le tableau t de longueur n ; et qui retourne le plus petit entier de l'intervalle $[0, 255]$ qui apparaît le moins souvent dans le tableau t. (Le nombre d'occurrences de cet entier peut être nul)

L'entier $\mathsf{min}(t, n)$ servira de *marqueur*. On note # pour ce marqueur et, pour simplifier, on suppose que son nombre d'occurrences est nul. Donc $r[\#] = 0$ quand $r = \mathsf{occurrences}(t, n)$. La compression par redondance du texte t fonctionne comme suit : toute répétition maximale contigüe d'une lettre où $t[i] = t[i + 1] = \cdots = t[j] = k$ est codée par les trois entiers $\#, (j - i), k$; toute apparition unique d'une lettre k est codée par cette même lettre.

Par exemple, si le tableau t contient les valeurs $\langle 0,0,3,2,3,3,3,3,3,3,5\rangle$. Le marqueur est donc 1 car 1 n'apparaît pas dans ce tableau. Le texte t' compressé est alors

$$\underbrace{1}_{\#},\underbrace{1,1,0}_{0,0},\underbrace{3}_{3},\underbrace{2}_{2},\underbrace{1,5,3}_{3,3,3,3,3,3},\underbrace{5}_{5}$$

Question 3 Écrire la fonction tailleCodage(t,n) qui prend comme argument le tableau t et calcule la taille n' du texte compressé ($n'=10$ dans l'exemple ci-dessus).

Question 4 Écrire la fonction codage(t,n) qui prend comme paramètre le tableau t et retourne un tableau d'entiers t' représentant le texte compressé.

Pour pouvoir décoder un texte t' ainsi compressé, il suffit de connaître le marqueur utilisé. Or ce marqueur est le premier entier du texte compressé.

2 Transformation de Burrows-Wheeler

Le codage par redondance n'est efficace que si le texte présente de nombreuses répétitions consécutives de lettres. Ce n'est évidemment pas le cas pour un texte pris au hasard. La transformation de Burrows-Wheeler est une transformation qui, à partir d'un texte donné, produit un autre texte contenant exactement les mêmes lettres mais dans un autre ordre où les répétitions de lettres ont tendance à être contigües. Cette transformation est bijective.

Considérons par exemple le texte d'entrée `concours`. Pour simplifier la présentation, nous utilisons ici des caractères pour le tableau d'entrée. Cependant, dans les programmes, on considère toujours (comme dans la première partie) que le texte d'entrée est un tableau d'entiers compris entre 0 et 255 inclus. Le principe de la transformation suit les trois étapes suivantes :

1 – On regarde toutes les rotations du texte. Dans notre cas, il y en a 8 qui sont :

concours
oncoursc
ncoursco
courscon
oursconc
ursconco
rsconcou
sconcour

2 – On trie ces rotations par ordre lexicographique (l'ordre du dictionnaire).

concour**s**
courscon**n**
ncoursc**o**
oncours**c**
oursconc
rsconcou
sconcou**r**
ursconc**o**

3 – Le texte résultant est formé par toutes les dernières lettres des mots dans l'ordre précédent, soit `snoccuro` dans l'exemple, ainsi que de l'indice de la lettre dans ce texte résultant qui est la première lettre du texte original, soit 3 dans notre exemple. On appelle cet entier la *clé* de la transformation.

On remarque que les deux `c` du texte de départ se retrouvent côte à côte après la transformation. En effet, comme le tri des rotations regroupe les mêmes lettres sur la première colonne, cela conduit à rapprocher aussi les lettres de la dernière colonne qui les précèdent dans le texte d'entrée.

On le constate aussi sur la chaîne : `concours␣de␣l␣ecole␣polytechnique` dont la transformée par Burrows-Wheeler est `sleeeeen␣dlt␣ucn␣ooohcpcc␣iuryqo`.

En pratique, on ne va pas calculer et stocker l'ensemble des rotations du mot d'entrée. On se contente de noter par $rot[i]$ la i-ème rotation du mot. Ainsi, dans l'exemple, $rot[0]$ représente le texte d'entrée `concours`, $rot[1]$ représente `oncoursc`, $rot[2]$ représente `ncoursco`, etc.

Question 5 Écrire la fonction comparerRotations(t, n, i, j) qui prend comme arguments le texte t de longueur n et deux indices i, j ; et qui renvoie, en temps linéaire par rapport à n :

1 si $rot[i]$ est plus grand que $rot[j]$ dans l'ordre lexicographique,

-1 si $rot[i]$ est plus petit que $rot[j]$ dans l'ordre lexicographique,

0 sinon.

On suppose disposer d'une fonction triRotations(t, n) qui trie les rotations du texte donné dans le tableau t en utilisant la fonction comparerRotation. Elle retourne un tableau d'entiers r représentant les numéros des rotations $(rot[r[0]] \leq rot[r[1]] \leq \cdots \leq rot[r[n-1]])$. Cette fonction réalise dans le pire des cas $O(n \ln n)$ appels à la fonction de comparaison.

Question 6 Écrire une fonction codageBW(t, n) qui prend en paramètre le tableau t ; et qui renvoie un tableau contenant le texte après transformation. (La clé sera stockée dans la dernière case de ce tableau)

Question 7 Donner un ordre de grandeur du temps d'exécution de la fonction codageBW en fonction de n.

Pour réaliser l'ensemble du codage, il ne reste plus qu'à réaliser la compression par redondance sur la transformée t' du texte d'entrée t.

3 Transformation de Burrows-Wheeler inverse

Pour décoder le texte t' (`snoccuro3` dans l'exemple) de taille $n' = n + 1$ obtenu après transformation, on construit d'abord un tableau *triCars* de taille n qui contient les mêmes lettres que le texte t' mais dans l'ordre lexicographique croissant. Dans l'exemple, $triCars = \langle c, c, n, o, o, r, s, u \rangle$.

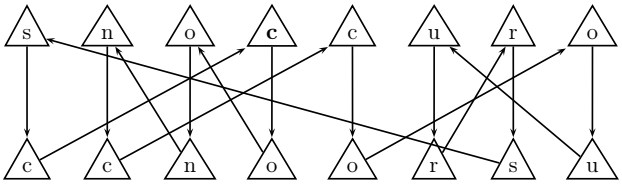
Question 8 Écrire une fonction frequences(t', n') qui prend comme argument un tableau t' de taille n' correspondant au texte codé (avec la clé dans la dernière case) ; et qui renvoie un tableau de taille 256 contenant le nombre d'occurrences de chaque lettre dans t'.

Question 9 Écrire la fonction triCarsDe(t', n') qui part du texte codé t' de taille n' ; et qui renvoie, en temps linéaire par rapport à n, le tableau *triCars* décrit précédemment.

Puis on considère le texte codé t' et le tableau *triCars* précédent (la clé est représentée en gras).

s	n	o	**c**	c	u	r	o	3
c	c	n	o	o	r	s	u	

À chaque lettre de la première ligne, on associe la lettre de la seconde à la même position. À chaque lettre de la deuxième ligne, on associe la même lettre de même rang dans la première ligne. La figure suivante montre ces deux correspondances.



On retrouve le texte de départ `concours` en partant de la clé (position de la lettre en caractère gras) et en suivant les flèches du dessin précédent.

Il faut donc construire le tableau *indices* tel que *indices*$[i]$ est l'indice de la lettre *triCars*$[i]$ dans le texte t'. Si plusieurs occurrences de cette lettre figurent dans t', on fait correspondre celle qui figure au même rang dans t'. Le tableau *indices* donne donc la correspondance représentée par les flèches de la seconde ligne vers la première. Sur l'exemple, le tableau *indices* contient les valeurs $\langle 3, 4, 1, 2, 7, 6, 0, 5\rangle$.

Question 10 Écrire la fonction trouverIndices(t', n') prenant en paramètre le texte t' codé de longueur n' ; et qui retourne le tableau *indices* précédemment décrit. Quel est son temps d'exécution en fonction de n' ?

Question 11 Écrire une fonction decodageBW(t', n') qui prend comme paramètre un texte t' de longueur n' ; et retourne le texte t d'origine. Quel est son temps d'exécution en fonction de n' ?

* *
*