Compression vers $[0, 1[$.

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

Durée de l'épreuve : 3 heures 30 minutes

Juin/Juillet 2022

ATTENTION !

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

Important.

Il vous a été donné 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.

Ce sujet est composé d'une introduction ainsi que de trois parties pouvant être traitées de façon indépendante. Ainsi, nous vous invitons à bien lire tout le sujet et d'éviter de rester trop longtemps bloqué. Pour résoudre la partie 3 (sauf sa dernière question), sur votre clé se trouve un jeu de données, ainsi qu'un bout de code qui le lit.

Dans ce sujet, le **logarithme** est toujours en base 2. En Python, cette fonction est calculée par `from math import log` puis `log(x, 2)`. En OCaml, la fonction `log` calcule le logarithme en base e donc pour gagner du temps, on définira une fonction `log2` en s'appuyant sur le fait que $\log_2 x = \log_b x / \log_b 2$ pour toute base entière $b \geq 2$.

Définitions

Un **mot** w de taille $n = |w|$ sur un alphabet Σ est constitué de n lettres $w[0] \cdots w[n-1]$ de Σ.

En théorie de l'information, on définit une **source** comme un objet qui produit de façon aléatoire une séquence de symboles, ce qui nous intéresse étant la probabilité d'apparition de chaque symbole. Dans ce sujet, pour simplifier on suppose qu'une source est un mot dont les symboles sont des lettres de Σ. Ainsi la probabilité d'apparition d'un symbole de S est son nombre d'occurrences dans S, divisé par la longueur de S.

L'**entropie** d'une source S dont chaque symbole x_i apparaît avec probabilité $p_i > 0$ est donnée par $H(S) = -\sum_i p_i \log_2 p_i$ et s'exprime en bits, cf. un exemple à la table 1. L'entropie est le nombre minimal de bits pour encoder un symbole en moyenne.

Compresser est une opération qui consiste à **coder**, c'est-à-dire transformer une source S en un **code** $\mathcal{C}$ qui peut être représenté par un certain nombre de bits. Dans ce sujet on s'intéresse à une compression sans perte, ce qui veut dire qu'il est toujours possible de **décoder**, c'est-à-dire récupérer la source originale à partir du code.

Écriture positionnelle en base b Soit b un entier supérieur ou égal à 2, K un entier positif. Un réel $r \in [0, 1]$ peut s'écrire comme un mot infini $a_1 a_2 \cdots$ de symboles de $\{0, \ldots, b-1\}$ tel que

$$r = \sum_{k=1}^{\infty} a_k b^{-k}.$$

Notez que cette écriture n'est pas nécessairement unique, comme par exemple $0{,}999\ldots = 1$ en base 10. À des fins de terminaison, on s'intéressera à une écriture tronquée à des mots de K caractères, et dans leur grande mansuétude, les examinateurs accepteront toutes les écritures possibles.

TABLE 1 – La source `ABBESSES.` émet les symboles `. A B E S` avec probabilité non nulle. Son entropie vaut $H(\texttt{ABBESSES.}) = 2 \times (-1/9 \log 1/9) - 2 \times (2/9 \log 2/9) - 3/9 \log 3/9 = 2{,}197160$ bits par symbole.

Symboles par ordre croissant	`.`	`A`	`B`	`E`	`S`
Nombre d'occurrences dans la source	1	1	2	2	3
Probabilité d'apparition	1/9	1/9	2/9	2/9	3/9

Générateur de nombres pseudo-aléatoires

Étant donné u_0 on définit la récurrence suivante :

$$u(0) = u_0$$

$$\forall t \in \mathbb{N},\ u(t+1) = (28 \times u(t)) \mod 336\,529$$

L'entier u_0 vous est donné, et doit être recopié sur votre fiche réponse avec vos résultats. Une fiche réponse type vous est donnée en exemple, et contient tous les résultats attendus pour une valeur de u_0 différente de la vôtre (notée $\widetilde{u_0}$). Il vous est conseillé de tester vos algorithmes avec cet $\widetilde{u_0}$. Pour chaque calcul demandé, avec le bon choix d'algorithme le calcul ne devrait demander qu'au plus quelques secondes, jamais plus d'une minute.

Question 1 *Calculer les valeurs suivantes :*

a) $u(1)\ \ mod\ \ 1\,000$ ***b)*** $u(42)\ \ mod\ \ 1\,000$ ***c)*** $u(10^5)\ \ mod\ \ 1\,000$.

Génération de source

L'alphabet comporte $M+1$ lettres de 0 à M. Dans ce sujet, $M = 29$. La source S_N est le mot de taille $N+1$ dont le t-ième symbole $S_N[t]$ pour $t = 0, \ldots, N$ vaut :

$$\begin{cases} S_N[t] = u(t) \mod M & t = 0, \ldots, N-1 \\ S_N[N] = M. \end{cases}$$

Ainsi, la source S_N termine toujours par le symbole de fin de séquence M.

Question 2 *Calculer les 5 derniers symboles de S_N pour les valeurs de N suivantes :*

a) $N = 10$ ***b)*** $N = 1\,000$ ***c)*** $N = 100\,000$.

Question 3 *Calculer l'entropie de la source S_N pour les valeurs de N suivantes, arrondie à 5 chiffres après la virgule.*

a) $N = 10$ ***b)*** $N = 1\,000$ ***c)*** $N = 100\,000$.

1 Codage de Huffman

On s'intéresse à une première méthode de compression qui s'appuie sur la construction d'un arbre binaire, dont l'arête vers le fils gauche (resp. droit) est étiquetée par 0 (resp. 1). Les feuilles représentent les symboles de la source, tandis que le chemin de la racine à une feuille est le mot binaire pour coder le symbole correspondant à cette feuille. Le code de la source est la concaténation des codes de ses symboles. Idéalement, on souhaite que les lettres fréquentes de la source soient codées avec des mots binaires courts. Un exemple est donné à la Figure 1.

L'algorithme de construction de l'arbre de Huffman s'appuie sur une structure de données $\mathcal{S}$ qui contient des arbres associés à des scores, et procède comme suit.

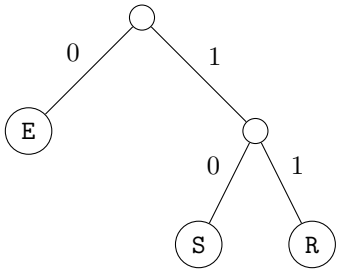


FIGURE 1 – Un exemple d'arbre de Huffman pour la source `SERREE`. `E`, lettre la plus fréquente, est codée par le bit `0` tandis que `R` et `S` sont codées sur deux bits, respectivement `10` et `11`. Ainsi le code de la source `SERREE` est `110101000` sur 9 bits.

Algorithme du codage de Huffman

1. Pour chaque symbole de la source (c'est-à-dire, apparaissant au moins une fois), insérer dans $\mathcal{S}$ une feuille étiquetée par le symbole, ayant pour score son nombre d'occurrences.
2. Tant qu'il y a plus d'un arbre dans $\mathcal{S}$
   (a) Extraire de $\mathcal{S}$ les deux arbres A_1, A_2 ayant les plus faibles scores p_1 et p_2. En cas d'égalité des scores, l'arbre ajouté le plus tôt dans $\mathcal{S}$ sort de $\mathcal{S}$ en premier.
   (b) Créer un nouvel arbre ayant pour sous-arbre gauche A_1 et sous-arbre droit A_2.
   (c) Ajouter ce nouvel arbre à $\mathcal{S}$ avec le score $p_1 + p_2$.
3. L'unique arbre restant dans $\mathcal{S}$ est l'arbre de Huffman.

Question 4 *Construire l'arbre de Huffman pour la source S_N. Si l'on code toute la source, quelle est la taille du code obtenu pour les valeurs de N suivantes ?*

a) $N = 10$ ***b)*** $N = 1\,000$ ***c)*** $N = 100\,000$.

Vous pouvez comparer vos réponses avec l'entropie à la question 3 qui correspond au nombre minimal de bits pour encoder un symbole en moyenne.

Question à développer pendant l'oral 1 *Quelle est votre complexité de construction de l'arbre de Huffman en fonction de M ? Peut-on faire mieux si l'on suppose que les symboles sont déjà triés dans $\mathcal{S}$ par nombre d'occurrences ? Comment décoder un message binaire $\mathcal{C}$ ainsi compressé et avec quelle complexité en fonction de la longueur du code $|\mathcal{C}|$?*

2 Codage arithmétique : compression vers $[0, 1[$

On s'intéresse à une nouvelle méthode de compression appelée codage arithmétique, inspirée par Shannon et aujourd'hui utilisée dans l'encodage vidéo H.264. L'idée consiste à diviser l'intervalle dans lequel on se trouve en zones proportionnellement à la fréquence de chaque symbole. Les zones sont ordonnées par l'ordre naturel sur les symboles. On part de l'intervalle $[0, 1[$ avec $L_0 = 0$ et $D_0 = 1$. Pour chaque symbole de la source, on emprunte la zone correspondant à ce symbole et on met à jour l'intervalle $[L_i, L_i + D_i[$. La source finit par le symbole de fin de séquence M qui permet d'identifier à quel moment interrompre le codage et surtout le décodage. Le code de la source peut enfin être n'importe quel nombre qui appartient à l'intervalle $[L_n, L_n + D_n[$ où n est la longueur de la source. Un exemple est donné à la Figure 2, où l'on a omis le symbole de fin de séquence pour simplifier l'illustration.

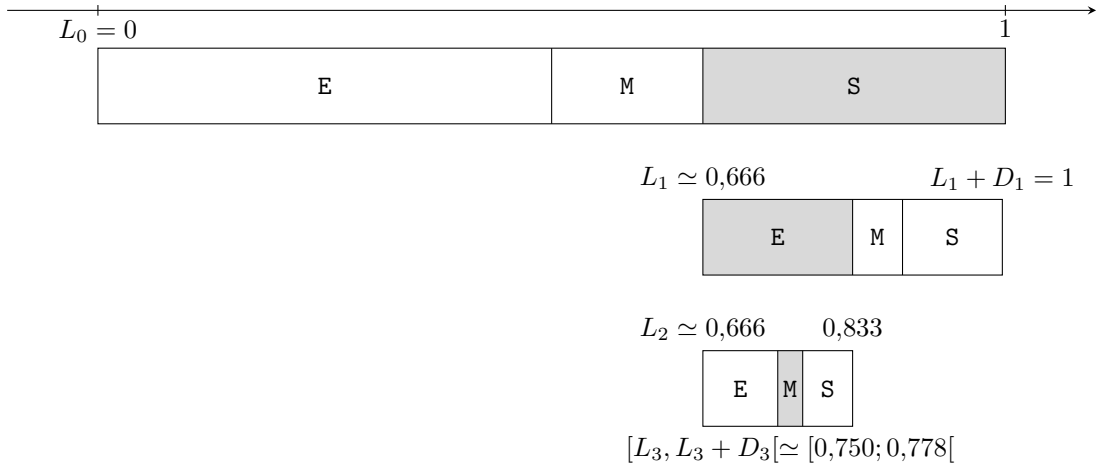


FIGURE 2 – Un exemple de codage arithmétique pour les trois premières lettres de la source `SEMEES`. Les fréquences des symboles `E M S` sont respectivement $3/6$, $1/6$, $2/6$. À cette itération, on sait que le code de la source sera dans l'intervalle $[0{,}750; 0{,}778[$.

Question 5 *Pour cette question on attend pour chaque valeur de N un nombre compris entre 0 et 1 arrondi à N chiffres en base 10 après la virgule. Implémenter le codage arithmétique et donner la valeur de L_{N+1} obtenue en compressant la source S_N pour :*

a) $N = 7$ ***b)*** $N = 10$ ***c)*** $N = 15$.

Question à développer pendant l'oral 2 *Présenter votre algorithme de compression et sa complexité en fonction de N et de la taille de l'alphabet M.*

Question 6 *Décoder les cinq premières lettres du code u_0/N sachant qu'on sait que la fréquence de la source correspond à S_N, pour les valeurs suivantes de N. Attention, ces valeurs ne sont pas les mêmes qu'aux autres questions.*

a) $N = 100$ ***b)*** $N = 500$ ***c)*** $N = 1\,000$.

Question à développer pendant l'oral 3 *Présenter votre algorithme de décodage et sa complexité en fonction de la longueur de la source à décoder N et de la taille de l'alphabet M. Que peut-il arriver lorsque la source est longue et que proposez-vous pour y remédier ?*

On cherche à représenter le mot de code avec le moins de bits possible, sachant qu'on peut tronquer l'écriture d'un mot s'il ne finit que par des zéros.

Question 7 *Écrire la décomposition en base 2 du nombre L_{N+1}. Pour cette question, si l'on reprend la notation $a_1 a_2 \cdots$ pour l'écriture en base 2, alors on attend les 6 bits $a_{15}a_{16}a_{17}a_{18}a_{19}a_{20}$.*

a) $N = 7$ ***b)*** $N = 10$ ***c)*** $N = 15$.

Question à développer pendant l'oral 4 *Présenter votre algorithme et sa complexité en fonction de la taille du code souhaitée $|\mathcal{C}|$, ici $|\mathcal{C}| = 20$ bits.*

Question à développer pendant l'oral 5 *Quelle est une borne supérieure convenable en bits pour écrire un nombre compris dans l'intervalle $[L_i, L_i + D_i[$, en fonction des caractéristiques de l'écriture de L_i et D_i ? Donner un algorithme pour identifier efficacement un code arithmétique de taille minimale pour une source donnée S_N, ainsi que sa complexité en fonction de l'écriture de L_i et D_i.*

Question à développer pendant l'oral 6 *Trouver un exemple simple de source pour lequel le codage arithmétique peut être arbitrairement meilleur que le codage de Huffman, c'est-à-dire qu'il nécessite moins de bits pour coder la source.*

3 Entropie et devinettes

Dans cette section on s'appuiera sur le jeu de données `data.txt` fourni avec le sujet, qui contient les mots uniques du dictionnaire dans un alphabet à 26 lettres, séparés par des retours de ligne. Si les mots sont numérotés de 0 à $D-1$, on ne considérera que les mots uniques dont les numéros sont dans l'ensemble $\{u(j), 0 \leq j < 50\,000\}$. Par exemple pour $\widetilde{u_0} = 42$ on considère seulement 48 081 mots du dictionnaire.
On cherche à deviner un mot au jeu du Pendu en un nombre minimal de coups. Pour simplifier les règles, au départ l'adversaire tire uniformément un mot du dictionnaire, et on ne connaît pas son nombre de lettres ; à chaque tour, on doit proposer une lettre et l'adversaire nous indique si elle se trouve dans le mot ou pas, sans donner sa position.
On cherche à construire un arbre binaire des états possibles : les nœuds contiennent la lettre qu'on joue, les arêtes indiquent les réponses de l'adversaire : 1, à gauche, si la lettre se trouve dans le mot à chercher ; 0, à droite, si la lettre ne s'y trouve pas. Dans un premier temps, on pose la question qui va le plus équilibrer l'espace des mots possibles, de façon à obtenir un arbre de hauteur moyenne faible, afin que le nombre de coups pour gagner soit petit, peu importe le mot choisi. En cas d'égalité entre deux lettres, on choisit la plus petite de l'alphabet. On ignore les lettres pouvant mener à un ensemble vide de mots possibles. Ainsi, si deux mots ont les mêmes ensembles de lettres, ils sont indistinguables, on considère que ça ne sert à rien de jouer davantage et qu'on a trouvé le mot.

Question 8 *Construire l'arbre binaire avec ce critère à partir du jeu de données fourni en entrée.*

a) *Quelle est la première lettre à jouer, c'est-à-dire l'étiquette de sa racine ?*

b) *Quelle est la longueur de sa plus petite et sa plus longue branche ?*

À présent on ne cherche plus à deviner le mot mais juste sa longueur, comprise entre 1 et 25. Au début de la partie, on a plus de chances de tomber juste si on devine une taille fréquente. La quantité qu'on cherche à réduire le plus est l'entropie des réponses possibles, soit $H(L_s)$ où L_s est le mot dont le i-ème caractère est la longueur du i-ème mot possible de l'ensemble s (l'ordre n'a pas d'importance). On préfère proposer la lettre ℓ telle que, si on note s_1 les mots contenant ℓ et s_0 les autres, la valeur $p(s_1)H(L_{s_1}) + p(s_0)H(L_{s_0})$ est minimale, où $p(s_1)$ désigne la probabilité que le mot à deviner contienne ℓ. Notons que si tous les mots d'un ensemble s ont la même longueur, alors $H(L_s)$ vaut 0 et le nœud correspondant est une feuille.

Question 9 *Construire l'arbre binaire avec ce critère à partir du jeu de données fourni en entrée.*

a) *Quelle est la première lettre à jouer, c'est-à-dire l'étiquette de sa racine ?*

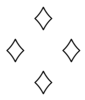
b) *Quelle est la longueur de sa plus petite et sa plus longue branche ?*

Enfin, on cherche à deviner une valeur entre 1 et N choisie par l'adversaire. On propose à chaque tour un nombre k tel que $1 \leq k \leq N$ et l'adversaire nous dit si la valeur à chercher est plus grande ou plus petite que k. Mais le coût n'est plus le nombre de coups joués, mais la somme des valeurs jouées : jouer le nombre k à un moment du jeu a un coût de k, et on cherche à minimiser le coût total de la partie dans le pire cas.

Question 10 *Pour les valeurs suivantes de N, quel est le coût d'une partie optimale dans le pire cas ?*

a) $N = u_0$ ***b)*** $N = 100 + u_0$ ***c)*** $N = 5\,000 + u_0$ ***d)*** $N = 100\,000 + u_0$.

Question à développer pendant l'oral 7 *Vous présenterez votre algorithme et sa complexité en fonction de N.*



Fiche réponse type : Compression vers $[0, 1[$.

$\widetilde{u_0}$: 42

Question 1

a) 176

b) 338

c) 645

Question 2

a) [20, 4, 16, 13, 29]

b) [6, 22, 22, 16, 29]

c) [10, 19, 27, 25, 29]

Question 3

a) 2.84535

b) 4.84049

c) 4.85763

Question 4

a) 32

b) 4896

c) 492471

Question 5

a) 0.3878305

b) 0.445078646

c) 0.492487883047874

Question 6

a) [11, 25, 24, 8, 9]

b) [2, 4, 4, 23, 12]

c) [1, 0, 0, 20, 20]

Question 7

a) [0, 0, 1, 1, 0, 1]

b) [0, 0, 1, 0, 1, 0]

c) [1, 1, 1, 0, 1, 0]

Question 8

a) o

b) 10 24

Question 9

a) n

b) 8 25

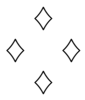
Question 10

a) 129

b) 645

c) 44365

d) 1275985



Fiche réponse : Compression vers $[0, 1[$.

Nom, prénom, u_0 : .....

Question 1

a)

b)

c)

Question 2

a)

b)

c)

Question 3

a)

b)

c)

Question 4

a)

b)

c)

Question 5

a)

b)

c)

Question 6

a)

b)

c)

Question 7

a)

b)

c)

Question 8

a)

b)

Question 9

a)

b)

Question 10

a)

b)

c)

d)