TD2 Théorie de l'information

Exo 1 On considère l'alphabet de l'ADN $\{A, C, T, G\}$ de probabilités d'apparition respectives $\{0.5, 0.3, 0.15, 0.05\}$. Calculer l'entropie de l'ADN et donner le code de Huffman de celui-ci.

Exo 2 Soit le message suivant : "UN MESSAGE TRES SIMPLE A CODER ".

1. Tracer l'arbre de Huffman correspondant à ce message;
2. Proposer un codage binaire optimal.

Exo 3 (Move-To-Front) L'algorithme *MTF* (pour *Move to Front*, déplacer vers l'avant) est un système de transformation de flot utilisé notamment dans le domaine de la compression de données en informatique. Elle consiste à remplacer chaque caractère par un indice, donné par un tableau évoluant de manière dynamique.
Fonctionnement : Le tableau est tout d'abord initialisé en rangeant les caractères utilisés pour l'encodage comme ceci :

Indice	0	1	2	3	4	5	6	...	26
Caractère	A	B	C	D	E	F	G	...	Z

Lorsqu'un caractère est lu, son indice est émis, puis ce caractère est placé en première position et tous les autres caractères décalés (d'où le nom de *Move to Front*). Par exemple si le premier caractère à encoder est un E, le tableau deviendrait :

Indice	0	1	2	3	4	5	6	...	26
Caractère	E	A	B	C	D	F	G	...	Z

Ainsi, lorsque des caractères semblables se suivent, le flux émis contiendra beaucoup de 0, ce qui dans une compression statistique (type codage de Huffman) augmentera considérablement le gain de compression. On note que dans ce cas, l'émission d'un 0 laisse le tableau identique, et que dans les autre cas, le réarrangement ne concerne que les premiers éléments du tableau. Par exemple, la séquence $EEEAABBB$ serait codée par la séquence *40010200*; le tableau évoluerait comme suit:

Indice	0	1	2	3	4	5	...
Etat initial	A	B	C	D	E	F	...
Tableau modifié par le 1*er* E	E	A	B	C	D	F	...
Tableau conservé par le 2*ème* E	E	A	B	C	D	F	...
Tableau conservé par le 3*ème* E	E	A	B	C	D	F	...
Tableau modifié par le 1*er* A	A	E	B	C	D	F	...
Tableau conservé par le 2*ème* A	A	E	B	C	D	F	...
Tableau modifié par le 1*er* B	B	A	E	C	D	F	...
Tableau conservé par le 2*ème* B	B	A	E	C	D	F	...
Tableau conservé par le 3*ème* B	B	A	E	C	D	F	...

Le décodage est tout aussi simple: à partir du même tableau initial, il suffit d'émettre le caractère correspondant à l'indice et de ranger le tableau en passant ce caractère en premier. Le tableau évolue exactement comme pendant la phase de codage.
Quel est le résultat de *MTF* pour le texte $AABBCC$ mais pour $ABCDEF$? Ecrire un algorithme qui implémente *MTF*. Quel est le décodage $EAAABAAEAAA$?
Exo 4 (Le codage δ de Elias)
Nous expliquons comment une succession de *zéros* peut être efficacement codée utilisant le codage de Elias.
Pour une mot hexadécimal w soit $\ell(w)$ sa longueur. Une succession $v = \underbrace{0\ 0\ \ldots\ 0}_{k}$ de k zeros sera codée par $v_1v_2v_3$, où:

- v_1 est une séquence de 0 de longueur égale à la longueur de v_2, et donc $\ell(v_1) = \lceil \log_{16} v_2 \rceil = \lceil \log_{16} \lceil \log_{16} k \rceil \rceil$.
- v_2 est la longueur v_3, i.e., $v_2 = \lceil \log_{16} k \rceil$ et $\ell(v_2) = \lceil \log_{16} \lceil \log_{16} k \rceil \rceil$.
- v_3 est la représentation hexadécimal de k, et donc $\ell(v_3) = \lceil \log_{16} k \rceil$.

Donc, la séquence $v = \underbrace{0\ 0\ \ldots\ 0}_{k}$ de longueur k sera codée par le mot $v_1 v_2 v_3$ de longueur

$$\lceil \log_{16} k \rceil + 2 \cdot \lceil \log_{16}(\lceil \log_{16} k \rceil) \rceil.$$

Exemple

v	longueur de v	v_1	v_2	v_3	longueur de $v_1 v_2 v_3$
0	1	$\underbrace{0}_{\ell(1)\text{ fois}}$	$\underbrace{1}_{\ell(1)}$	$\underbrace{1}_{1}$	3
0 0	2	$\underbrace{0}_{\ell(1)\text{ fois}}$	$\underbrace{1}_{\ell(2)}$	$\underbrace{2}_{2}$	3
0 0 0	3	$\underbrace{0}_{\ell(1)\text{ fois}}$	$\underbrace{1}_{\ell(3)}$	$\underbrace{3}_{3}$	3
$\cdots$	$\cdots$				
$\underbrace{0\ 0\ \cdots\ 0}_{45}$	45	$\underbrace{0}_{\ell(2)\text{ fois}}$	$\underbrace{2}_{\ell(2D)}$	$\underbrace{2D}_{45}$	4
$\cdots$	$\cdots$				
$\underbrace{0\ 0\ \cdots\ 0}_{768}$	768	$\underbrace{0}_{\ell(3)\text{ fois}}$	$\underbrace{3}_{\ell(3\ 0\ 0)}$	$\underbrace{3\ 0\ 0}_{768}$	5

Donner le codage de Elias pour $v = B\underbrace{0\ldots0}_{63\text{ fois}}AC\underbrace{0\ldots0}_{255\text{ fois}}AB$. Décoder la suite AB027FFD, qui a été codée par le codage de Elias.

Exo 5 Pour une fenêtre de taille 6 et un tampon de lecture de taille 3 donner :

- le résultat de l'alorithme de compression LZ77 appliqué au message :
  - ___*aabaacacb*, et
  - ___*ababcacbacc*.
- le résultat de l'algorithme de décompression LZ77 appliqué à la donnée
  - (0,0,b),(1,1,a),(2,2,b),(3,2,a), et
  - (0,0,a),(0,0,c),(1,1,a),(3,2,c).

Exo 6 Donner le résultat de l'algorithme de compression LZ78 appliqué au message :'aabbababababbbba bbbabb'.

Exo 7 Donner le résultat de l'algorithme de décompression LZ78 appliqué à la donnée : (0,a) (1,a) (0,b) (3,a) (4,a) (5,a) (4,b).

Exo 8 Pour que la compression soit rapide, il faut utiliser une structure de donnée arborescente appelée *trie*. Le problème consiste en effet à trouver rapidement le numéro de la plus longue *phrase* rencontrée précédemment sous forme d'arbre.
Chaque phrase correspond à un nœud de l'arbre. La phrase 0 qui représente la suite vide est la racine de l'arbre. Les fils de chaque nœud son numérotés par une lettre (on dit que cette lettre est l'étiquette de l'arc entre le père et le fils). Un nœud de l'arbre correspond à la phrase obtenue en lisant les étiquettes sur les arcs du chemin de la racine au nœud. Le numéro de la phrase est stocké dans le nœud corresondant de l'arbre. (Un nœud peut avoir autant de fils qu'il y a de lettres dans l'alphabet.)

Ecrire les tries obtenus apres la lecture des textes des exo 6 et 7.

Exo 9 : Codage arithmétique

Le codeur arithmétique traite le fichier dans son ensemble, en lui associant un unique nombre décimal rationnel. Ce nombre compris entre 0 et 1 dépend non seulement des symboles du fichier dans l'ordre où ils apparaissent mais aussi de leur distribution statistique.

Considérons l'alphabet $\{A, E, I, O, U, !\}$ avec les probabilités suivantes :

Caractère	Probabilité
A	$0,2$
E	$0,3$
I	$0,1$
O	$0,2$
U	$0,1$
!	$0,1$

Dans l'intervalle de probabilité $[0; 1]$, chaque symbole de l'alphabet se voit affecter un intervalle de probabilité :

Caractère	Probabilité	Intervalle
A	$0,2$	$[0; 0,2[$
E	$0,3$	$[0,2; 0,5[$
I	$0,1$	$[0,5; 0,6[$
O	$0,2$	$[0,6; 0,8[$
U	$0,1$	$[0,8; 0,9[$
!	$0,1$	$[0,9; 1[$

Coder le message $EAII!$

1. Initialisation de l'intervalle de travail $[0; 1[$
2. Le premier symbole est représenté par son intervalle de la première étape
3. Calcul des nouveaux intervalles des symboles suivants par la méthode suivante :
   (a) nouvelle borne inférieure = ancienne borne inférieure + largeur ancien intervalle $\times$ borne inférieure symbole lu
   (b) nouvelle borne supérieure = ancienne borne inférieure + largeur ancien intervalle $\times$ borne supérieure symbole lu

Sachant que ancienne *borne inférieure* et *largeur ancien intervalle* correspondent à la ligne précédente dans le tableau :

Caractère	Intervalle
Initialisation	$[0; 1[$
Lecture de E	$[0,2; 0,5[$
Lecture de A	$[0,2; 0,26[$
Lecture de I	$[0,23; 0,236[$
Lecture de I	$[0,233; 0,2236[$
Lecture de !	$[0,23354; 0,2336[$

Un nombre dans l'intervalle final, comme $0,23355$, code de façon unique le message $EAII!$

Décodage

- Comme
$$V_1 = 0,23355$$
est dans l'intervalle de E, la première lettre est E;
- $$V_2 = \frac{V_1 - \text{Inf}(E)}{\Delta(E)} = \frac{0,23355 - 0,2}{0,5 - 0,2} = 0,11133$$
Comme V_2 est dans l'intervalle de A, la deuxième lettre est A
- $$V_3 = \frac{V2 - \text{Inf}(A)}{\Delta(E)} = \frac{0,11133 - 0}{0,2 - 0} = 0,5556$$
Comme V_3 est dans l'intervalle de I, la troisième lettre est I.

... et ainsi de suite jusqu'à la dernière lettre du mot, sachant que l'on peut soit définir sa taille pour pouvoir le décoder, soit définir un caractère d'arrêt.

Question

Coder le massage *aacb* avec les probabilités données par :

Caractère	Probabilité
a	$0,5$
b	$0,2$
c	$0,3$