Représentation des caractères

Nous allons ici faire un point sur la représentation des données autres que numériques sur un ordinateur. Comment en effet peut-on représenter un texte, une image, un son, à partir uniquement de zéros et de uns ? Nous commençons par traiter le cas du texte, en retraçant de façon succinte l'évolution de la représentation des caractères.

Les caractères sont représentés sur ordinateur grâce à une norme, comme pour les réels. On s'appuie donc sur des tables, définies de façon officielle et internationale.

I Code Ascii

À l'origine, le format de représentation était le format Ascii – *American Standard Code for Information Interchange* – que vous trouverez dans la figure 1. Le format Ascii représente chaque caractère sur un octet. Le bit de poids fort sert à vérifier qu'une erreur n'est pas intervenue : il est égal à 0 si le nombre de 1 est pair, 1 sinon.

PDF : fr en v · d · m	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
002	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Figure 1 – Table de caractères Ascii - source Wikipedia

1 – Combien peut-on représenter de caractères avec ce système ?

2 – La lettre `a` est représentée par le code 97 : écrire l'octet permettant de coder cette lettre.

3 – Dans le tableau de la figure 1, le `a` a pour code 61 : pourquoi ?

4 – Quelle opération faut-il faire sur l'octet de codage pour passer d'une lettre en minuscule à la même lettre en majuscule ?

Dans la pratique actuelle, le bit de parité est oublié et remplacé par défaut par un zéro. Certaines positions de la table sont inutilisées : elles représentent des actions spécifiques. On a par exemple le code 10 qui représente le saut de ligne, et le code 13 qui représente le retour charriot.

5 – On donne le texte suivant en hexadécimal : 4d 50 53 69 20 21 21. Que signifie-t-il ?

6 – Quel inconvénient voyez vous à cette table Ascii ?

II Norme ISO-8859-1

Pour palier à ces inconvénients, la norme ISO-8859-1 a été mise en place. Le tableau de cette norme se trouve sur la figure 2. Pour ce codage, on oublie le bit de parité, ce qui permet de multiplier le nombre de caractères

par deux. Cette extension a été réalisée par un consortium international et date de 1986.

ISO/CEI 8859-1															
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*positions inutilisées*															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*positions inutilisées*															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

FIGURE 2 – Table de caractères ISO-8859-1 - source Wikipedia

7 – Y'a-t-il un caractère utile en français qui n'apparaît pas dans cette table ?

8 – En fonction de ce qui a été vu précédemment, que devrait-on avoir à la place du ß ?

9 – On a le texte suivant en hexadécimal sous la norme ISO-8859-1 : 6c 61 20 70 72 e9 70 61 20 63 27 65 73 74 20 65 78 74 72 61. Que signifie-t-il ?

10 – Voyez-vous un inconvénient à cette norme internationale de réprésentation des caractères ?

III UNICODE - UTF8

Pour réprésenter l'ensemble des caractères internationaux, la table de caractère UNICODE a été introduite en 1991. Elle affecte un code à plus de 110000 caractères. Elle est compatible avec la norme ISO-8859-1 mais permet de coder des caractères non latins : caractères arabes, chinois, japonais, cyrilliques... C'est le codage UTF8, le plus répendu, qui permet son implémentation sur ordinateur. Dans ce codage, un caractère est représenté sur 1, 2 3 ou 4 octets. Pour permettre le décodage, le codage respecte une forme bien particulière :

0xxxxxxx	1 octet pour 1 à 7 bits
110xxxxx 10xxxxxx	2 octets pour 8 à 11 bits
1110xxxx 10xxxxxx 10xxxxxx	3 octets pour 12 à 16 bits
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4 bits pour 17 à 21 bits

Pour connaître le codage d'un caractère, on cherche son code dans la table UNICODE, puis on le transcrit en binaire. En fonction de la longueur du résultat, on choisit le format à appliquer.

11 – Le code UNICODE du symbole euro est le `8364`. Donner son codage UTF8.

12 – On reprend la phrase précédente. Sachant que le numéro UNICODE du `é` est le `233`, donner sa représentation hexadécimale UTF8. Donner le texte complet sous forme hexadécimale.

13 – Décoder alors ce texte en supposant qu'il est écrit sous la norme ISO-8859-1. Que remarque-t-on ?

14 – Avez-vous déjà rencontré ce genre de problème ?

Le type de codage utilisé n'est pas une information du fichier : on est obligé de dire à l'application utilisée quelle type de codage employer pour le décodage. Vous pouvez faire le test sur un page comportant des accents avec votre navigateur préféré : on trouve souvent un menu du type `Affichage/Encodage`. Heureusement, le codage utf8 est aujourd'hui universellement reconnu, même s'il a pris du temps à s'imposer sous Windows.

15 – La lettre ε est codée en UTF8 par `ce b5`. Quel est son code UNICODE ? Comment se décode-t-il en ISO-8859-1 ?

Avant l'arrivée de cette harmonisation au format UNICODE ou ISO-8859-1, chaque pays avait en fait sa déclinaison d'une table de caractère. L'arrivée de l'Internet a soulevé tout un tas de problèmes de compatibilité.

IV Un principe de compression des données

Dans un texte, on retrouve souvent les mêmes caractères. Par exemple, le `e` est très présent dans la langue française. Son code ASCII est le 97 et sa représentation binaire (sans le bit de parité) est donc 1100001 : cela nécessite 7 bits, c'est à dire autant que pour le `z`, qui lui apparait bien moins souvent. Peut-on trouver une manière de coder l'information textuelle qui ferait correspondre aux caractères fréquents un code de faible longueur, et aux caractères rares un code de plus grande longueur ? Un tel codage permettrait d'utiliser moins de place dans la mémoire de l'ordinateur.

On considère le texte suivant : *« allez les bleus »*.

16 – Pour chaque caractère de la phrase, relever son nombre d'occurences et les ordonner du plus rare au plus fréquent.

17 – Tant que la liste des caractères n'est pas réduite à un élément, on effectue la démarche suivante :
- supprimer les deux premiers caractères de la liste ; on les note c_i et c_j ;
- créer un arbre à deux branches dont les feuilles sont c_i et c_j ;
- on définit le nombre d'occurences de cet arbre par la somme des occurences des deux feuilles ;
- introduire à sa place cet arbre dans la liste des caractères ;
- recommencer.

18 – On obtient ainsi un arbre qui permet de définir un code pour chaque caractère : pour obtenir le code d'un caractère on parcourt l'arbre : un embranchement à gauche donne 0, un embranchement à droite donne 1. Quel est le code de la lettre `z` ? Quel est le code de la lettre `l` ?

19 – Que remarque-t-on sur la longueur d'un code par rapport à la fréquence d'une lettre ?

20 – Donner le code en binaire compressé de ce texte ? Quel gain a-t-on fait par rapport au texte original ?

21 – Comment apparaît ce texte si on le décode en suivant le codage ASCII ?

Le codage de Huffman a été mis en place en 1952. Il a été montré qu'il était optimal : il est impossible, en moyenne, de mieux compresser l'information qu'avec ce codage. Il était utilisé sur les premiers Apple pour la compression des données et offrait un gain de 30% sur la mémoire. Il est encore aujourd'hui utilisé dans les formats d'image `jpeg` ou de video `mpeg`. C'est une méthode de compression sans perte, contrairement à la plupart des méthodes de compression d'images par exemple.