Cet article est rédigé par des élèves. Il peut comporter des oublis et imperfections, autant que possible signalés par nos relecteurs dans les notes d'édition.

Tour de Magie

Elèves chercheurs :
Alexandre Feghouli, Frédéric Fruit, Yorick Gromada, Brandon Pluchard, Barbara Soudant, Emeline Vanhove,
Collège Jean Jaurès,
rue du 8 mai 1945
59690 Vieux-Condé

Enseignant :
Nicolas Vanlancker

Chercheuse :
Sylvie Derviaux
LAMAV EA 4015, Université de Valenciennes et du Hainaut Cambrésis

Madame Derviaux nous a proposé de travailler sur un tour de magie.
On pense à un nombre entier compris entre entre 1 et 100. Elle nous propose des listes et on lui dit dans quelles listes ce nombre se trouve. Elle retrouve le nombre initial.

Voici les 7 listes proposées :

liste n°1 :

1	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79
81	83	85	87	89	91	93	95	97	99

liste n°2 :

2	3	6	7	10	11	14	15	18	19
22	23	26	27	30	31	34	35		
38	39	42	43	46	47	50	51		
54	55	58	59	62	63	66	67		
70	71	74	75	78	79	82	83		
86	87	90	91	94	95	98	99		

liste n°3 :

4	5	6	7	12	13	14	15	20
21	22	23	28	29	30	31	36	
37	38	39	44	45	46	47	52	
53	54	55	60	61	62	63	68	
69	70	71	76	77	78	79	84	
85	86	87	92	93	94	95	100	

liste n°4 :

8	9	10	11	12	13	14	15
24	25	26	27	28	29	30	31
40	41	42	43	44	45	46	47
56	57	58	59	60	61	62	63
72	73	74	75	76	77	78	79
88	89	90	91	92	93	94	95

liste n°5 :

16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95

liste n°6 :

32	33	34	35	36	37	38
39	40	41	42	43	44	
45	46	47	48	49	50	
51	52	53	54	55	56	
57	58	59	60	61	62	
63	96	97	98	99	100	

liste n°7 :

64	65	66	67	68	69	70
71	72	73	74	75	76	
77	78	79	80	81	82	
83	84	85	86	87	88	
89	90	91	92	93	94	
95	96	97	98	99	100	

On a commencé par faire des tests.
Par exemple, le nombre 65 est dans la liste 7 et dans la liste 1. Madame Derviaux a réussi à retrouver 65 quand on lui a donné les numéros de listes.
Elle nous a posé plusieurs questions :
- Comment fonctionne ce tour de magie ?
- Comment ont été fabriquées les listes ?
- Peut-on trouver un algorithme pour fabriquer ces listes ?
- Comment appliquer le fonctionnement de ces listes à l'envoi de messages sur internet ?

I Comment fonctionne ce tour de magie ?

Notre tour de magie fonctionne grâce a l'addition des premiers nombres de chaque liste où se trouve le nombre mystère.
Par exemple, le nombre 65 est dans la liste 7 (qui commence par 64) et dans la liste 1 (qui elle commence par le chiffre 1). Pour retrouver le nombre mystère, il suffit de calculer 64 + 1= 65.
Second exemple: Si on nous dit que le nombre choisi est dans la liste 7 (qui commence par 64), la liste 5 (qui commence par 16), la liste 3 (qui commence par 4), la liste 2 (qui commence par 2) et liste 1 (qui commence par 1), le magicien doit calculer 64+16+4+2+1. Le résultat est donc 87.

Nous avons aussi constaté que le premier nombre de chaque liste est le double du précédent. (1 – 2 – 4 – 8 – 16 – 32 et 64).
Madame Derviaux nous a dit que c'était important et qu'il s'agissait de **puissances de 2**.

$16 = 2 * 2 * 2 * 2 = 2^4$

$8 = 2 * 2 * 2 = 2^3$

$4 = 2 * 2 = 2^2$

$2 = 2^1$ *(on le définit comme cela)*

$1 = 2^0$

II Comment ont été créées ces listes ?

Madame Derviaux souhaitait faire le tour sur des nombres plus grands. Nous devions

compléter les listes. Nous avons donc dû comprendre comment elles avaient été créées.
Pour créer ces listes, il faut d'abord tenir compte de la remarque sur les puissances de 2 : ce sont les premiers nombres de chaque liste.
Ensuite, il faut ajouter les nombres les uns après les autres. Comment faire ?
Le 1, le 2, le 4 sont placés.
Le 3 se décompose en 3=2+1 donc on l'écrit dans la liste 1 et dans la liste 2.
Le 5 se décompose en 5=4+1 donc le 5 s'écrit dans la liste 3 et la liste 1...
Et ainsi de suite pour tous les nombres...

Pour placer, par exemple, 35, on remarque que 35 est compris entre 32 et 64.
Donc 35 = 32 + … ?...
Il manque 3, qui peut s'écrire sous la forme 2 + 1
Donc 35 = 32 + 2 + 1
35 doit s'écrire dans la liste 6 (qui commence par 32) , la liste 2 (qui commence par 2) et la liste 1 (qui commence par 1).
Autre exemple : 87= 64 + 16 + 4 + 2 + 1
donc 87 doit s'inscrire dans la liste 7, la liste 5, la liste 3, la liste 2 et la liste 1.

Pour le nombre 121, nous allons détailler tous les calculs :
121=64+ …
121-64=57
121=64+32+...
57-32=25
121=64+32+16+...
25-16=9
121=64+32+16+8+1
Finalement, 121 doit être écrit dans la liste 7, la liste 5, la liste 4, la liste 3 et la liste 1.

Pour les nombres plus grands, nous pouvons créer une huitième liste qui commencera par 128.

Nous sommes donc capables, petit à petit, de construire l'ensemble des listes, en rajoutant les nombres les uns après les autres.

Ensuite, nous nous sommes rendus compte qu'on pouvait écrire la décomposition des nombres dans un tableau :

	Liste 8	Liste 7	Liste 6	Liste 5	Liste 4	Liste 3	Liste 2	Liste 1
	Commence par $128=2^7$	Commence par $64=2^6$	Commence par $32=2^5$	Commence par $16=2^4$	Commence par $8=2^3$	Commence par $4=2^2$	Commence par $2=2^1$	Commence par $1=2^0$
65		OUI						OUI
87		OUI		OUI		OUI	OUI	OUI
121		OUI	OUI	OUI	OUI			OUI

Cela nous a permis de les écrire rapidement avec des puissances de 2 :
Par exemple

$$87=0\times2^7+1\times2^6+0\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0$$

ou encore :

$$121=0\times2^7+1\times2^6+1\times2^5+1\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0$$

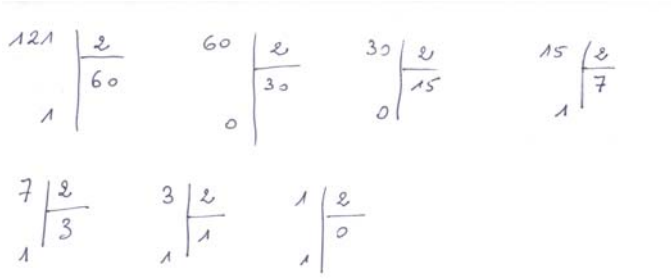
Ceci nous donne l'écriture en base 2 (en ne gardant que les coefficients 0 ou 1 devant les puissances) :
87= 01010111 et 121 = 01111001

Cette écriture en base 2 est composée de 8 chiffres des 0 ou des 1. On appelle donc ce qu'on a obtenu un **octet**.

III Un algorithme pour retrouver la décomposition en base 2

Nous avons trouvé une méthode automatique pour retrouver la décomposition en base 2.
En regardant la première méthode (celle que nous avons présentée dans la partie II) nous avons vu qu'on cherchait toujours ce qu'il restait quand on retirait les premiers nombres des listes...
C'est ce RESTE qui nous a donné l'idée de diviser. (1)

Pour retrouver la décomposition de 121 en base 2, on fait comme ceci :
- on calcule 121 divisé par 2 : ce qui fait 60, il reste 1.
- On prend le quotient qui est 60. on divise 60 par 2, ce qui fait 30, il reste 0.
- On calcule 30 divisé par 2. Cela fait 15, il reste 0.
- On divise 15 par 2 : 7 et il reste 1.
- Ensuite, on calcule 7 divisé par 2, cela donne 3, il reste 1.
- On calcule 3 divisé par 2, cela fait 1, il reste 1.
- Et pour finir, on divise 1 divise par 2, 0 il reste 1.
- si on recommence, ce sera la même étape que précédemment. On a donc terminé.



Pour finir et écrire 121 en base 2, on réécrit les restes en commençant par le dernier trouvé.
121 en base 2 donne 1111001 et pour obtenir un octet, il suffit d'ajouter un zéro devant : 01111001

Les listes que nous avait données Madame Derviaux lors de notre première rencontre sont donc des décompositions en base 2 des 100 premiers nombres.
Lors du congrès, un spectateur nous a demandé si on avait essayé de faire les listes avec des puissances de 3. Nous avons essayé et nous avons un souci puisque, par exemple, 4 s'écrit $4=1\times3^1+1\times3^0$ et 5 s'écrit $5=1\times3^1+2\times3^0$. Ce coefficient 2 nous pose problème pour fabriquer un tour de magie. Nous n'avions que des zéros et des uns avec les puissances de 2.

IV Une utilisation en informatique.

En informatique, il faut coder les informations sous forme de uns et de zéros, correspondant au passage du courant ou non.

Les informations sont traduites sous forme d'octets.
Il existe un tableau avec tous les caractères d'écriture possibles qui se nomme une table ASCII.

Voici un extrait d'une table ASCII :

Table ASCII (0 - 127)

Décimal	Octal	Hex	Binaire	Caractère
000	000	00	00000000	NUL (Null char.)
001	001	01	00000001	SOH (Start of Header)
002	002	02	00000010	STX (Start of Text)
003	003	03	00000011	ETX (End of Text)
004	004	04	00000100	EOT (End of Transmission)
005	005	05	00000101	ENQ (Enquiry)
006	006	06	00000110	ACK (Acknowledgment)
007	007	07	00000111	BEL (Bell)
008	010	08	00001000	BS (Backspace)
009	011	09	00001001	HT (Horizontal Tab)
010	012	0A	00001010	LF (Line Feed)
011	013	0B	00001011	VT (Vertical Tab)
012	014	0C	00001100	FF (Form Feed)
013	015	0D	00001101	CR (Carriage Return)
014	016	0E	00001110	SO (Shift Out)
015	017	0F	00001111	SI (Shift In)
016	020	10	00010000	DLE (Data Link Escape)
017	021	11	00010001	DC1 (XON)(Device Control 1)
018	022	12	00010010	DC2 (Device Control 2)
019	023	13	00010011	DC3 (XOFF)(Device Control 3)
020	024	14	00010100	DC4 (Device Control 4)
021	025	15	00010101	NAK (Negative Acknowledgement)
022	026	16	00010110	SYN (Synchronous Idle)
023	027	17	00010111	ETB (End of Trans. Block)
024	030	18	00011000	CAN (Cancel)
025	031	19	00011001	EM (End of Medium)
026	032	1A	00011010	SUB (Substitute)
027	033	1B	00011011	ESC (Escape)
028	034	1C	00011100	FS (File Separator)
029	035	1D	00011101	GS (Group Separator)
030	036	1E	00011110	RS (Request to Send)(Record Separator)
031	037	1F	00011111	US (Unit Separator)
032	040	20	00100000	SP (Space)
033	041	21	00100001	! (exclamation mark)
034	042	22	00100010	" (double quote)
035	043	23	00100011	# (number sign)
036	044	24	00100100	$ (dollar sign)
037	045	25	00100101	% (percent)
038	046	26	00100110	& (ampersand)
039	047	27	00100111	' (single quote)
040	050	28	00101000	((left opening parenthesis)
041	051	29	00101001	) (right closing parenthesis)
042	052	2A	00101010	* (asterisk)
043	053	2B	00101011	+ (plus)
044	054	2C	00101100	, (comma)
045	055	2D	00101101	- (minus or dash)
046	056	2E	00101110	. (dot)
047	057	2F	00101111	/ (forward slash)
048	060	30	00110000	0
049	061	31	00110001	1
050	062	32	00110010	2

105	151	69	01101001	i
106	152	6A	01101010	j
107	153	6B	01101011	k
108	154	6C	01101100	l
109	155	6D	01101101	m
110	156	6E	01101110	n
111	157	6F	01101111	o
112	160	70	01110000	p
113	161	71	01110001	q
114	162	72	01110010	r
115	163	73	01110011	s
116	164	74	01110100	t
117	165	75	01110101	u
118	166	76	01110110	v
119	167	77	01110111	w
120	170	78	01111000	x
121	171	79	01111001	y
122	172	7A	01111010	z
123	173	7B	01111011	{ (left opening brace)
124	174	7C	01111100	| (vertical bar)
125	175	7D	01111101	} (right closing brace)
126	176	7E	01111110	~ (tilde)
127	177	7F	01111111	DEL (delete

Par exemple, le numéro 121 est le « y minuscule »

Si un ordinateur envoie le signal 01111001 (via internet par exemple), cela voudra donc dire « y minuscule ».

V Une nouvelle opération

Madame Derviaux nous a appris une nouvelle opération. Il s'agit de l'opération ⊕ qui fonctionne sur les nombres 0 et 1.
Elle nous l'a définie ainsi :
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$0 \oplus 0 = 0$
$1 \oplus 1 = 0$

On peut aussi le lire dans le tableau ci-dessous.

⊕	0	1
0	0	1
1	1	0

Ensuite, Madame Derviaux nous a expliqué comment cette opération fonctionnait avec les octets. Avec les octets, il suffit de calculer les chiffres les uns avec les autres, place par place, en utilisant les règles de ⊕ .

$$\begin{array}{r} 01101011 \\ \oplus\ 10010001 \\ \hline = \ 11111010 \end{array}$$

Nous avons cherché les propriétés de cette opération :

- Cette opération est commutative. En effet, si on prend deux octets a et b, pour calculer a ⊕ b, on doit utiliser l'opération ⊕ sur chacun des nombres des octets, « place par place ». Et si on regarde comment est définie l'opération ⊕ sur 0 et 1, on voit qu'elle est commutative. Finalement ⊕ est commutative pour les octets (a ⊕ b = b ⊕ a)
- Cette opération a un neutre (« un octet qui, lorsqu'il est *ajouté* à n'importe quel autre octet, ne change pas ce dernier ») c'est l'octet 00000000 (cela vient aussi du principe de fonctionnement de ⊕ sur les nombres 0 et 1 : le 0 n'a pas d'effet sur l'autre nombre !)
- a ⊕ a = 00000000 (on dit que l'octet a est l'inverse de a). Cela vient toujours du principe que le calcul se fait « place par place » et que 0 ⊕ 0 = 0 et 1 ⊕ 1 = 0.
- Si on prend deux octets a et b,

$$a \oplus b \oplus b = a$$

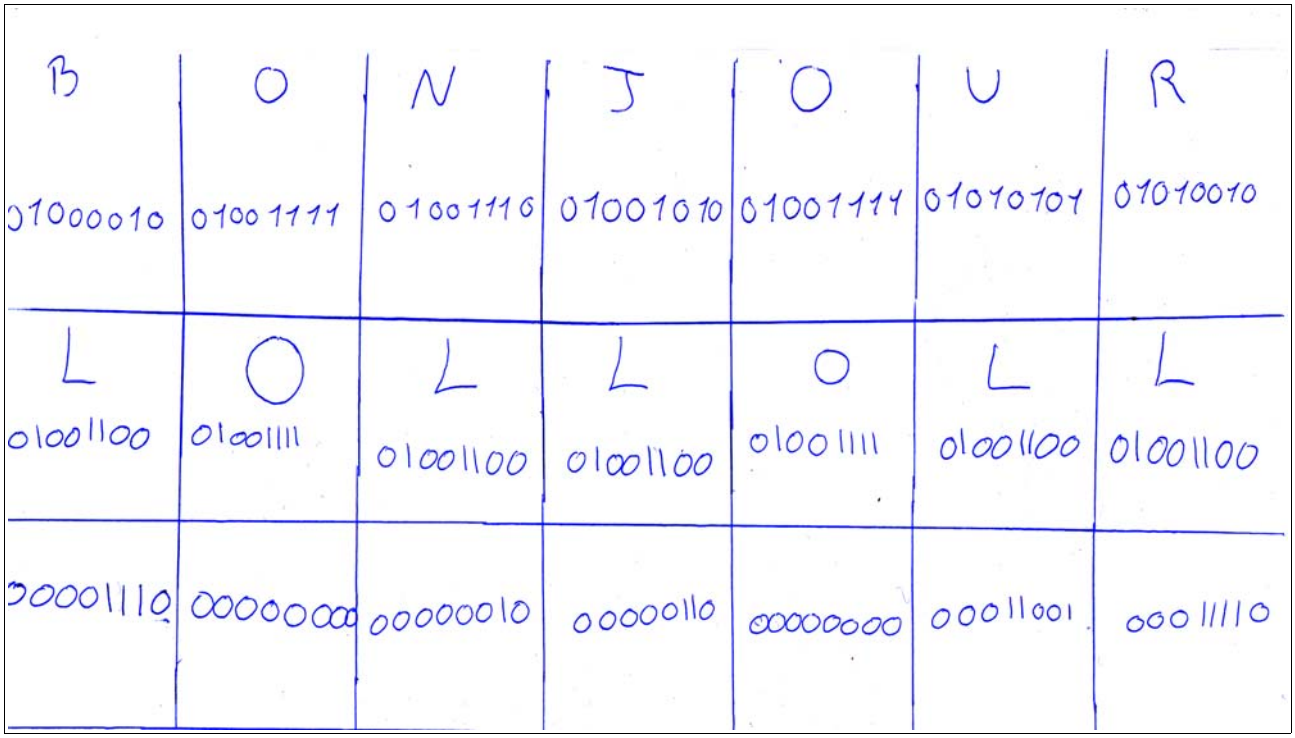
Cela vient juste de la propriété précédente.

VI Une application en informatique :

« *Comment envoyer un mail à Pierre pour lui dire « Bonjour » sans que personne ne puisse comprendre le message?* »

Nous cherchons à comprendre comment on peut envoyer un message à un ami sur internet sans que tout le monde puisse le lire.
La méthode est la suivante :
1- On prend chaque lettre du mot que l'on veut envoyer (ici « BONJOUR ») et on repère le numéro correspondant à chaque lettre ou chaque symbole, inscrit dans la table ASCII
2- On écrit chaque numéro en octet. Ainsi le B de BONJOUR devient 01000010 etc..
3- On choisit une « clé » que seul l'expéditeur et le receveur connaissent. Cela peut être une phrase longue ou simplement quelques lettres. Dans l'exemple, la clé est « LOL » On écrit donc « LOL » sous le mot « BONJOUR ». Comme la clé n'est pas assez longue, on répète le mot « LOL » plusieurs fois.
Ensuite on traduit « LOL » avec les nombres de la table ASCII, écrits en octets.
4- On additionne les octets lettre par lettre selon la règle précédente. (on « ajoute » donc le B de « BONJOUR » et le L de « LOL »).
5- On trouve un ensemble de nombres sous formes d'octets. C'est le message codé qui est envoyé.
6- Le destinataire du message l'additionne avec la clé, lettre par lettre et grâce à la règle a ⊕ b ⊕ b = a, il obtient le message que l'on voulait faire passer.

Illustration : comment coder le mot « BONJOUR » avec la clé « LOL »



Note d'édition

(1) Il manque une justification de l'algorithme : pourquoi celui-ci produit bien le résultat attendu ?