Tout a un reflet numérique

> PAR GILLES DOWEK, THIERRY VIÉVILLE ET EMMANUEL BACCELLI, CHERCHEURS A L'INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE (INRIA), JEAN-PIERRE ARCHAMBAULT ET BENJAMIN WACK, ENSEIGNANTS

A L'information et son reflet

À chaque instant, notre cerveau capte, enregistre et traite des informations. Ainsi pouvons-nous regarder, mémoriser, analyser, puis retranscrire une image, un texte ou un son. Depuis longtemps, l'homme a cherché à capter et à reproduire ce reflet de la réalité par un dispositif technique. Il a donc inventé le téléphone, la photographie, le cinéma, etc. Mais qu'en est-il aujourd'hui avec les machines que nous utilisons, qui sont des dispositifs de calcul programmé ?

Prenons l'exemple d'une calculette électronique : quand nous disons qu'elle « calcule », ses circuits électriques s'ouvrent et se ferment de telle sorte que l'affichage corresponde au résultat du calcul effectué. Mais le fait que cela prenne le sens d'un calcul tient à l'humain qui utilise cette calculette ; la machine ne « comprend » évidemment pas les différentes opérations, qui s'exécutent uniquement grâce aux propriétés électriques de ses circuits : elle ne possède que le reflet électronique du calcul.

INFORMATION, HASARD ET COMPLEXITÉ

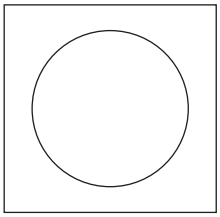
Regardons cette suite de chiffres : 14159 26535 89793 23846 26433 83279 50288 41971. Il y a sûrement plus d'informations dans cette suite-là que dans celle-ci : 00000 00000 00000 00000 00000 00000 00000 00000. Mais en quoi la première est-elle plus complexe ? Est-elle aléatoire, chaotique, sans ordre ni régularité ? Ou bien fortement structurée, riche en informations « utiles » ?

En fait, si nous regardons bien le début de cette suite, nous reconnaissons les premières décimales de π, peut-être le nombre le plus célèbre de toute l'histoire des mathématiques : autant dire que son information est très utile… du moins pour les géomètres et les mathématiciens ! En tout cas, il est bien difficile de retenir les chiffres de la première suite. Mais, pour la deuxième, il suffit de retenir : « huit fois cinq zéros ». Cet exemple, cité par Jean-Paul Delahaye dans son ouvrage *Information, complexité et hasard* (Hermes Science, 1999, coll. « Langue, raisonnement, calcul »), est très instructif, car il nous montre que nous devons bien distinguer le contenu *brut* d'une information, qui ne dépend pas du sens, et la *valeur* d'une information qui, elle, dépend du but fixé.

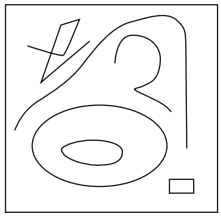
B Fabriquer un reflet numérique à partir d'un objet réel

DÉCRIRE UN DESSIN PAR UNE DÉFINITION STRUCTURÉE

Comment décrire le dessin ci-dessous ? Une solution est de dire : « Cette image est formée d'un cercle. » Nous pouvons même être plus précis et indiquer les coordonnées du centre du cercle, son rayon, sa couleur, l'épaisseur du trait, etc. À partir de cette description, il est possible de reconstituer le dessin.

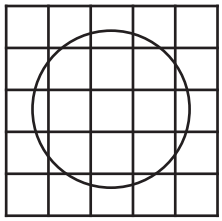


Mais n'utiliser que des mots serait bien moins pratique pour le dessin suivant. En effet, comment décrire facilement et en détail ces traits qui semblent tracés au hasard ? Qu'en serait-il avec un dessin représentant un objet réel ou un visage ?

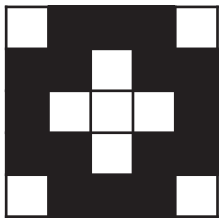


DÉCRIRE AVEC DES NOMBRES BINAIRES

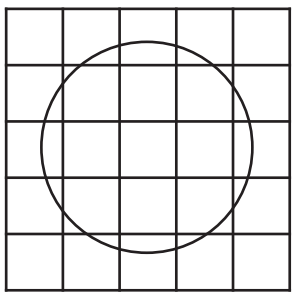
Une autre méthode, qui a l'avantage de pouvoir être utilisée pour n'importe quel dessin, consiste à superposer un quadrillage au tracé. Chaque case de ce quadrillage est appelée « pixel » (contraction de l'anglais *picture element*).

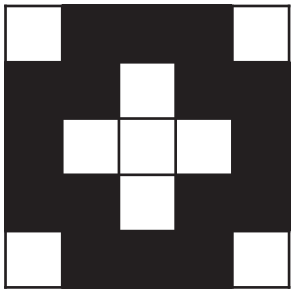


On noircit ensuite les pixels qui contiennent une portion de trait :

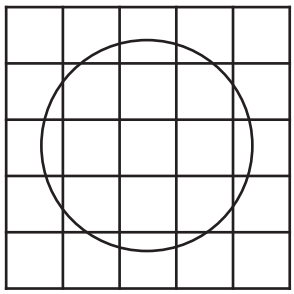


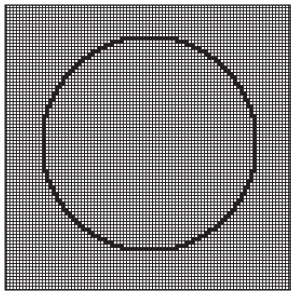
Puis, il suffit d'indiquer la couleur de chacun des pixels, en les lisant de gauche à droite et de haut en bas, comme un texte. Cela donne : blanc, noir, noir, noir, blanc, noir, noir, blanc, noir, noir, noir, blanc, blanc, blanc, noir, noir, noir, blanc, noir, noir, blanc, noir, noir, noir, blanc. Cependant, cette description est assez approximative, et l'on constate une grande différence entre ces deux images :





Il est possible de rendre la description plus précise en utilisant un quadrillage, non plus de 5 x 5, mais de 100 x 100 pixels :





À partir de quelques millions de pixels, nous ne serions plus capables de faire la différence entre les deux images. Cette méthode est approximative, mais universelle : n'importe quel dessin, même très compliqué, se décrit exactement comme un dessin simple. Celui que nous venons de voir se décrit donc par une suite de mots : « blanc » ou « noir ». Comme seuls ces mots sont utilisés, nous pouvons être plus économes et remplacer chacun d'eux par un seul symbole : le mot « noir » par la lettre *n* ou le chiffre 0, et le mot « blanc » par la lettre *b* ou le chiffre 1. Le dessin ci-dessus, avec une grille de 5 x 5, se décrit alors par une suite de 25 chiffres : 10001001000111000010010001.

Ces différentes descriptions sont équivalentes : le dessin se décrit toujours par une suite de symboles choisis parmi deux. Leur nom importe peu, et nous pourrions les appeler « noir » et « blanc », *a* et *b*, 0 et 1, pile et face, - et +, Dupond et Dupont, etc. Cela ne changerait pas grand-chose. En général, on choisit 0 et 1. Ces symboles sont appelés « chiffres binaires » ou « bits » (abréviation de l'anglais *binary digits*, « chiffres binaires ») ; une suite de 0 et de 1, « suite binaire » ou, parfois, « mot binaire ». Notre dessin se décrit ainsi avec 25 chiffres binaires, ou encore 25 bits.

LE POINT-CLÉ : CONVENIR D'UN STANDARD

Bien entendu, il faut que nous soyons tous d'accord pour décrire l'ensemble des dessins de la même façon et décider, comme nous l'avons fait dans l'exemple précédent, que le noir est représenté par 0, le blanc par 1, et que les pixels se lisent de gauche à droite et de haut en bas. Il faut donc mettre au point un standard. Le codage de l'information est avant tout une affaire de convention. Regardons alors comment nous pourrions convenir de représenter d'autres objets.

C Décrire numériquement toutes sortes d'objets

DU DESSIN AUX IMAGES PHOTOGRAPHIQUES

Si nous voulons décrire une image en couleur – par exemple, une image dans laquelle chaque pixel peut être noir, rouge, vert ou bleu –, une première méthode consiste à décrire cette image par une suite dont chaque élément représente l'un des quatre mots. Prenons l'image ci-dessous :

Elle se décrit de cette manière : noir, rouge, noir, rouge, noir, noir, rouge, noir, rouge, noir, bleu, noir, bleu, noir, vert, bleu, bleu, noir, bleu, vert, bleu, bleu, vert, vert, vert. Comme pour les images en noir et blanc, on peut être plus économe et décrire chaque couleur par une seule lettre ou un seul chiffre. Par exemple, en utilisant la correspondance :

n	noir
r	rouge
v	vert
b	bleu

L'image se décrit donc par la suite : nrnrnnrnrnbnbnvbbnbvbbvvv. Cela fonctionne à condition, bien entendu, que celui qui code et celui qui décode cette image suivent parfaitement le standard proposé ici. Cette méthode utilise quatre symboles différents. Il est possible aussi de décrire l'image par une suite binaire, c'est-à-dire dans laquelle on utilise seulement deux symboles et non quatre. Mais comme il y a plus de deux couleurs, il est impossible de décrire chaque couleur par un seul chiffre binaire : 0 ou 1. En revanche, on peut décrire chaque couleur par une suite de deux bits. Par exemple :

00	noir
01	rouge
10	vert
11	bleu

Cette image se décrit alors par la suite : 00 10 00 10 00 00 10 00 10 00 01 00 01 00 11 01 01 00 01 11 01 01 11 11, donc 25 x 2 = 50 bits. Les espaces entre chaque couple de valeurs binaires sont inutiles, sauf pour en faciliter la lecture.

Une fois choisi le nombre de pixels (ici, 5 x 5 ; pour une photographie numérique, il en faudrait quelques millions) et le nombre de bits sur lequel on représente la couleur de chaque pixel (ici, 2 bits ; en réalité de 8 à 32 bits, selon la palette des couleurs), toutes les images se représentent par une suite binaire de la même longueur : cette longueur s'obtient simplement en multipliant le nombre de pixels de l'image par le nombre de bits utilisés pour décrire la couleur d'un pixel. C'est ainsi que se décrivent les photographies dans les ordinateurs et les autres machines numériques.

DÉCRIRE DES TEXTES PAR DES SUITES BINAIRES

Décrire un texte semble *a priori* plus simple que décrire une image, puisqu'un texte est déjà une suite de symboles, les lettres de l'alphabet. L'alphabet latin, que l'on utilise en français, contient 26 lettres. Mais, avec les majuscules et les minuscules, les lettres accentuées, les chiffres et les symboles de ponctuation, écrire en français nécessite plutôt une centaine de symboles.

Il faut aussi, cependant, pouvoir décrire toutes les autres langues. Par exemple, les 28 symboles de l'alphabet arabe, avec ses symboles complémentaires, comme on les voit sur une ancienne machine à écrire. Ainsi la phrase « Tout se code en binaire » s'écrira-t-elle « دوكلا يئانث يف يش لك » (qui se lit de droite à gauche) et se codera-t-elle comme suit : 1011 1100 1001 1111 0000 1010 1111 0111 1101 0110 0101 1111 0110 1100 1011 1110 1000, en utilisant les codes binaires à 4 bits des 16 lettres suivantes :

ء	آ	أ	ؤ	إ	ئ	ا	ث	د	ش	ف	ك	ل	ن	و	ي
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Ce code de 16 lettres, nous nous sommes amusés à l'inventer uniquement pour cette présentation. Cependant, là encore, pour partager les informations, il ne faut pas utiliser un code particulier, comme celui-ci, mais un code standard, adopté par tous, et qui représente tous les caractères de la langue utilisée. Les caractères les plus courants en Occident, ceux que nous tapons au clavier, sont codés sur 8 bits, avec un code universel, dit ASCII (*american standard code for information interchange*) étendu. Pour coder tous les textes du monde, c'est l'Unicode, contenant plus de 120 000 caractères, qui a été retenu. Ce standard utilise le code ASCII et des codes complémentaires à 16 ou 24 bits.

D Décrire des nombres

Le codage des valeurs numériques est de nature différente, car ce n'est pas un code arbitraire « standard », mais un choix issu de l'arithmétique. En effet, il serait possible de représenter un nombre tel que 2741 comme la succession des chiffres 2, 7, 4 et 1, puis de coder chacun de ces chiffres par son équivalent ASCII pour obtenir : 00110001 00110111 00110100 00110001. Mais, si un tel codage représente bien la chaîne « 2741 », il ne représente pas le nombre 2741 et ne permet pas d'effectuer des calculs arithmétiques tels que l'addition et la multiplication. Pour obtenir une représentation utilisable dans des calculs, il suffit cependant de passer de la numération en base dix classique à la numération en base deux.

Le tableau suivant donne les dix premiers nombres en base deux dans la colonne du milieu et leur représentation binaire dans la colonne de droite :

0x*8*+**0**x*4*+**0**x*2*+**0**x*1*=	0	**0000**
0x*8*+**0**x*4*+**0**x*2*+**1**x*1*=	1	**0001**
0x*8*+**0**x*4*+**1**x*2*+**0**x*1*=	2	**0010**
0x*8*+**0**x*4*+**1**x*2*+**1**x*1*=	3	**0011**
0x*8*+**1**x*4*+**0**x*2*+**0**x*1*=	4	**0100**
0x*8*+**1**x*4*+**0**x*2*+**1**x*1*=	5	**0101**
0x*8*+**1**x*4*+**1**x*2*+**0**x*1*=	6	**0110**
0x*8*+**1**x*4*+**1**x*2*+**1**x*1*=	7	**0111**
1x*8*+**0**x*4*+**0**x*2*+**0**x*1*=	8	**1000**
1x*8*+**0**x*4*+**0**x*2*+**1**x*1*=	9	**1001**

Si l'on observe bien ce codage, on constate quelque chose de plus : il est le résultat d'un calcul, montré dans la colonne de gauche. Le codage binaire d'un nombre (que nous écrivons quasiment toujours en base dix, en utilisant 10 chiffres) est ce même nombre, écrit en base deux.

DU CODAGE BINAIRE AU NOMBRE ENTIER POSITIF

Voici le calcul qui permet de retrouver un nombre à partir de son code binaire : additionner les bits, chacun multiplié par la puissance de deux qui lui correspond, comme ci-dessous, donne le nombre en base dix.

$$\begin{aligned} &\mathbf{1}\times 8 + \mathbf{0}\times 4 + \mathbf{0}\times 2 + \mathbf{1}\times 1 = \\ &\mathbf{1}\times 2^3 + \mathbf{0}\times 2^2 + \mathbf{0}\times 2^1 + \mathbf{1}\times 2^0 = \\ &((\mathbf{1}\times 2 + \mathbf{0})\times 2 + \mathbf{0})\times 2 + \mathbf{1} = \end{aligned} \quad \Big| \quad \mathbf{9}$$

À la première ligne, nous avons multiplié respectivement par 1, 2, 4, 8 de droite à gauche les bits, qui représentent 9. À la seconde ligne, nous nous sommes aperçus que 1, 2, 4, 8 sont, en fait, les puissances de 2 : 2^0, 2^1, 2^2, etc. À la troisième ligne, nous factorisons 2 autant que possible ; cette écriture, avec ses parenthèses bien placées, nous montre qu'il suffit de multiplier par deux, puis d'additionner les bits de gauche à droite pour trouver le nombre en base dix. Voilà donc un calcul, une méthode mécanique appelée « algorithme », qui permet de passer du codage binaire au nombre dans sa représentation usuelle.

DU NOMBRE ENTIER POSITIF À SON CODAGE BINAIRE

Réciproquement, il existe un calcul pour obtenir le code binaire de n'importe quel nombre entier. Si nous notons 9 % 2 = 1, le reste de la division par 2 de 9, ou 4 % 2 = 0, le reste de la division de 4 par 2, etc., alors le calcul suivant, où nous calculons successivement le reste de la division par deux, puis divisons par deux, donne… le codage binaire du nombre 9 :

$$\begin{aligned} &9 \,\%\, 2 = \mathbf{1} \\ &(9 / 2 = 4) \,\%\, 2 = \mathbf{0} \\ &\quad (4 / 2 = 2) \,\%\, 2 = \mathbf{0} \\ &\quad\quad (2 / 2 = \mathbf{1}) \,\%\, 2 = \mathbf{1} \end{aligned}$$

REPRÉSENTER DES ENTIERS POSITIFS DE TOUTES TAILLES

Bien sûr, cela marche avec tous les nombres entiers positifs, et nous pouvons facilement calculer combien de bits sont nécessaires pour les coder :

Entiers	0 à 1	0 à 3	0 à 7	0 à 15	0 à 255	0 à 1023	0 à 4 milliards	0 à plus de $1{,}8 \times 10^{20}$
Codés sur	1 bit	2 bits	3 bits	4 bits	8 bits	10 bits	32 bits	64 bits

La plupart de nos ordinateurs utilisent aujourd'hui 32 ou 64 bits pour coder les nombres entiers ; ils peuvent donc coder directement des nombres très importants, et, si nécessaire, en utilisant davantage de bits… des nombres vertigineusement grands !

CALCULER DIRECTEMENT SUR LES NOMBRES BINAIRES

Nous pouvons donc représenter des nombres entiers positifs par la suite de bits de leur représentation binaire. Ce codage est très précieux, car il permet de faire directement des opérations sur ces nombres. Par exemple, additionner deux entiers positifs revient à additionner leur représentation binaire : on obtient directement le codage binaire du résultat. De même pour toutes les autres opérations numériques, pour comparer ces entiers entre eux, etc. Souvenons-nous que toutes les valeurs numériques sont non seulement stockées en binaire, mais aussi manipulées en binaire dans l'ordinateur.

REPRÉSENTER DES NOMBRES ENTIERS NÉGATIFS ET POSITIFS

Jusqu'ici, nous n'avons considéré que des nombres entiers positifs. Comment représenter des nombres entiers soit positifs, soit négatifs ? Choisissons de représenter le signe par un bit supplémentaire :

0	+
1	-

Avec cet ajout, nous représentons la valeur absolue du nombre comme un entier positif, et son signe avec ce bit en plus. Nous pouvons alors définir toutes les opérations numériques que nous connaissons en tenant compte de ce signe.

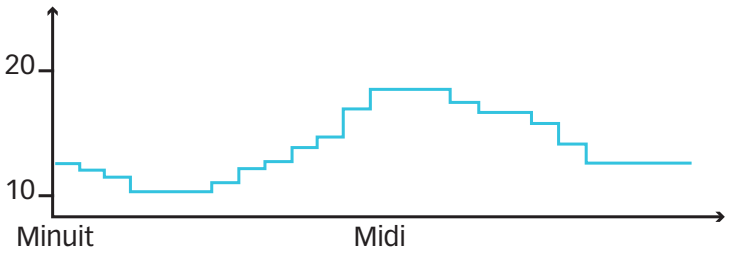
REPRÉSENTER LES NOMBRES RÉELS

Pour représenter un nombre réel – s'il s'agit d'un nombre décimal, par exemple 3,1416 –, on utilise la « notation scientifique » 31416 10^{-4}, qui code d'une part l'exposant (ici, $^{-4}$) et d'autre part les décimales du nombre, plus exactement sa mantisse (ici, 31416), sous forme de deux entiers, codés en binaire et mis bout à bout. C'est de cette façon que sont codés les nombres dans les calculatrices et les ordinateurs. Les calculs numériques se font alors directement sur les nombres binaires et sont ensuite transcrits pour afficher le résultat attendu. Mais attention : les fractions comme 10/3 = 3,333 333 333… ou encore les nombres réels comme π = 3,141 592 653 589 793 238 462 643 383 279 502 884 197 1… ont une suite infinie de décimales ! Cela veut dire que nous ne pouvons pas représenter leur valeur numérique, mais uniquement une *approximation* de cette valeur. De ce fait, *tous* les calculs que nous ferons avec un ordinateur ne seront jamais justes ! Ils ne seront pas vraiment faux non plus, ils seront « approximatifs ».

Décrire toutes les informations humaines

CODER UNE MESURE PHYSIQUE

Comment coder la courbe de température de votre chambre au fil du jour ? On peut coder le nombre qui correspond à la mesure de température à chaque heure et les mettre bout à bout.



Ces 24 nombres vont donner une bonne approximation du chaud et du froid… et du fait que votre chauffage est visiblement coupé la nuit.

CODER UN SON

Le son est représenté par une courbe qui correspond aux vibrations de la voix, des bruits ou des instruments. Coder un son revient à coder une suite de valeurs numériques qui correspondent à cette courbe. Bien sûr, il faut beaucoup de valeurs (environ 10 000 par seconde) pour obtenir une bonne approximation de toutes les vibrations sonores ; il faut 16 bits pour coder correctement la valeur de l'intensité et de la fréquence du son à chaque instant. Et pour coder une minute de son : 16 bits x 10 000 valeurs/seconde x 60 secondes = presque 10 millions de bits.

C'est ce qui se passe dans les lecteurs MP3, par exemple, à une différence importante près. En effet, le codage du son à l'aide du standard MP3 est « astucieux » : il met en œuvre des méthodes de compression avec perte qui tiennent compte de la perception humaine. Cette approximation supplémentaire permet de gagner de la place dans la mémoire et du temps lors de la communication.

CODER DE LA MUSIQUE

Il suffit de choisir un code binaire pour chaque note – sa hauteur (aiguë ou grave) et sa durée –, chaque instrument. Ensuite, en mettant bout à bout ces codes, on obtient la séquences des notes. On code ainsi la partition musicale et son interprétation. Un tel codage existe ; il est standard et se nomme MIDI (*musical instrument digital interface*).

CODER UN FILM

Un film n'est jamais qu'une séquence d'images. Schématiquement, il suffit donc de coder celles-ci et de placer des suites de bits bout à bout. Sans oublier de coder aussi le son. Là encore, il existe des astuces pour regrouper les parties qui se ressemblent, pour gagner de la place et du temps. Mais le principe reste le même.

Qu'avons-nous découvert ici ? La polyvalence des ordinateurs et des autres machines numériques : ils traitent des données très diverses, mais toujours décrites de la même manière, par des suites binaires utilisant deux symboles.

F Pourquoi utiliser un codage binaire ?

IL FAUT AU MOINS DEUX SYMBOLES

Pouvons-nous utiliser moins de deux symboles pour décrire les images, les textes ou les nombres ? Non, car si nous avons un seul symbole, par exemple 0, une fois le nombre de bits fixé à 10, 100 ou 1 million, il n'y a plus qu'une seule suite de 10, 100 ou 1 million de symboles : 0000000000000000000000000… qui ne peut donc décrire qu'un seul objet ! L'information commence avec la dualité – opposition et complémentarité – du 0 et du 1, du chaud et du froid, du bas et du haut, etc. Quand on a un symbole unique, tout est uniforme ; il n'y a pas d'information.

EN UTILISER DEUX, C'EST BIEN PRATIQUE…

En revanche, nous aurions pu choisir d'utiliser plus de deux symboles : les 10 chiffres arabes, les 22 lettres de l'alphabet phénicien, les 24 lettres de l'alphabet grec ou les 26 lettres de l'alphabet latin. Mais n'utiliser que deux symboles a plusieurs avantages. Le premier est que ce choix est indépendant de la nature des informations décrites. Il aurait été dommage de concevoir des ordinateurs grecs qui utilisent 24 symboles et des ordinateurs latins qui en utilisent 26.

Par ailleurs, la description de l'information avec deux symboles est plus simple à mettre en œuvre dans un ordinateur. Cela se réalise électriquement avec un interrupteur ouvert ou fermé, un courant électrique présent ou absent, etc. Ce choix rend aussi les ordinateurs plus robustes : s'il y avait plus de deux états intermédiaires, le système physique pourrait les mélanger, et donc provoquer plus facilement des erreurs. En se limitant à 0 et 1, ce risque est minimisé.

… ET CELA PERMET AUSSI DE MESURER LA QUANTITÉ D'INFORMATION

Le nombre de bits est exactement la *quantité brute d'information qui a été codée.* Pour comprendre cette idée, jouons au jeu du portrait : devinez le nom d'une personne en ne posant que des questions où l'on répond par « oui » ou par « non », en suivant l'exemple ci-dessous :

Qui ?	Genre ?	*Look* emo ?	Moins de 15 ans ?	
Pierre	gars	oh non !	oui	'001'
Nadia	fille	good-good	oui	'111'
David	gars	énormément	non	'010'
Hamdi	gars	certes	oui	'011'
Marie	fille	surtout pas	non	'100'
Adèle	fille	évidemment	non	'110'
Ming Yu	fille	beurk	oui	'101'
Igor	gars	horreur !	non	'000'

En trois questions, « Est-ce une fille ?, Aime-t-elle le *look* emo ?, A-t-elle moins de 15 ans ? », vous allez forcément deviner, parmi les huit personnes, celle qui correspond au portrait, car vous aurez toute l'information utile ici. D'ailleurs, nous avons symbolisé ce fait en utilisant un bit – reporté dans la colonne de droite – pour chacune des questions binaires.

Deux questions binaires, c'est-à-dire deux bits d'information, ne suffisent pas ici. Si vous ne savez pas que la personne recherchée a moins de 15 ans, alors Adèle et Nadia ou Igor et Pierre, par exemple, ne peuvent pas être distingués, puisqu'ils pensent la même chose du *look* emo. On constate bien ici que le nombre de bits correspond au nombre de questions binaires à poser pour deviner toute l'information, et donc à la taille en information. Prenons un autre exemple : devinons l'âge d'un élève du secondaire, entre 11 et 18 ans. Nous pourrions poser huit questions, « A-t-il 11 ans ? A-t-il 12 ans ? A-t-il 13 ans ? A-t-il 14 ans ? A-t-il 15 ans ? A-t-il 16 ans ? A-t-il 17 ans ? A-t-il 18 ans ? », avec le risque, si nous manquons de chance, de devoir poser les huit questions avant de connaître la solution.

Voici une meilleure méthode : demandons d'abord si l'élève a moins de 15 ans. Si la réponse est « oui », alors nous savons que l'élève a entre 11 et 14 ans ; si la réponse est « non », alors nous savons que l'élève a entre 15 et 18 ans. Dans les deux cas, nous sommes passés d'une fourchette de 8 ans à celle de 4 ans. En divisant l'intervalle de recherche par deux, nous avons gagné un « atome » d'information, un bit d'information. Si nous continuons à poser une question qui divise l'intervalle de recherche par deux, nous arrivons à une fourchette de deux ans, puis à la troisième question, à une fourchette d'un an, etc., et nous avons trouvé la solution : l'âge de l'élève. Nous voyons bien ici que ce choix parmi 8 valeurs correspond à 3 questions, donc 3 bits d'information, chaque bit correspondant, de gauche à droite, aux réponses « oui » (= 1) ou « non » (= 0) aux trois questions.

11	12	13	14	15	16	17	18
‘111’	‘110’	‘101’	‘100’	‘011’	‘010’	‘001’	‘000’

Un « atome » d'information est donc un bit, la réponse « oui » ou « non » à une question binaire, le choix entre deux options, etc. La taille en information du choix entre plusieurs options correspond au nombre de bits pour le coder. Par exemple, choisir entre quatre couleurs : noir, rouge, vert, bleu nécessite deux bits. Choisir entre les sept couleurs que nous voyons aujourd'hui dans l'arc-en-ciel nécessite trois bits, tandis que le code 000 n'est pas utilisé ici, ce qui importe peu.

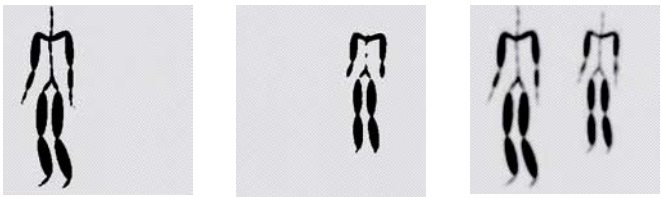
violet	indigo	bleu	vert	jaune	orange	rouge
‘111’	‘110’	‘101’	‘100’	‘011’	‘010’	‘001’

La taille en information de deux informations indépendantes s'additionne. Par exemple, si nous codons l'âge d'un élève du secondaire sur 3 bits et son genre (fille/garçon), alors il faudra en tout 4 bits. Mais la taille en information de deux informations redondantes ne s'additionne pas. Par exemple, si nous codons l'âge d'un élève du secondaire sur 3 bits et le fait qu'il ait le droit ou non de conduire un scooter sur 1 bit, puisque nous savons que toute personne de 14 ans et plus peut conduire un scooter, il est inutile d'ajouter cette information, car elle est « contenue » dans l'âge : il y a donc toujours 3 bits d'information.

Nous avons ainsi découvert une mesure physique très intéressante : de même que la chaleur se mesure avec la température en degrés, la tension électrique en volts, la quantité brute d'information se mesure en bits. Cette mesure correspond tout simplement au nombre de bits minimal nécessaire pour stocker l'information sans la tronquer. Évidemment, elle ne dit rien de la valeur de l'information, ni de sa complexité. Par exemple, la séquence binaire 1100100100001111110110101010001000100001011010001100001000110100 est constituée de 64 bits. A-t-elle une grande valeur ? Aucune, si les 0 et les 1 ont été choisis sans raison. A-t-elle une grande complexité, c'est-à-dire est-elle difficile à calculer ? Non, si nous avons juste appuyé au petit bonheur la chance sur le 0 et le 1 du clavier. Mais cette séquence correspond au codage binaire, au milliardième près, du nombre le plus célèbre en mathématiques : le nombre π. Voilà qui le rend plus précieux et qui indique qu'il est plutôt complexe à calculer. Et pourtant, il est toujours constitué de 64 bits. C'est le contenu brut en information qui est mesuré ici, indépendamment de sa valeur ou de sa complexité.

Commencer à manipuler les objets numériques que sont les images

AJOUTER DEUX IMAGES : L'EFFET DE CALQUE



En ajoutant la valeur de chaque pixel de la première image à celle de chaque pixel de la seconde image, on obtient une image qui amalgame les valeurs des deux, un peu comme un effet de calque. Si nous avions considéré des pixels de couleurs, nous aurions mélangé les couleurs et obtenu des combinaisons diverses.

SOUSTRAIRE DEUX IMAGES : UN DÉTECTEUR DE MOUVEMENT

À quoi peut correspondre maintenant la « soustraction » de deux images, c'est-à-dire le fait de soustraire à chaque pixel de la première image la valeur du pixel correspondant à la seconde ? Il est facile de voir que si les pixels ont une même valeur, alors la différence sera nulle : si rien n'a changé d'une image à l'autre, il n'y aura pas de différence. Si, au contraire, quelque chose a changé, alors la différence sera ou positive (disons plutôt noire) ou négative (disons plutôt blanche), et nous détecterons ainsi les changements, comme nous le voyons ci-dessous pour deux images entre lesquelles la figurine a quelque peu bougé :