L'idée de Leonid Levin est que plus un fichier est complexe, plus la probabilité qu'il soit produit par un programme de calcul choisi aléatoirement est faible. Le « théorème de codage » de Leonid Levin stipule : « La complexité de Kolmogorov $K(s)$ d'un fichier numérique s est déterminée par la probabilité $p(s)$ qu'un programme choisi aléatoirement produise s. Complexité et probabilité sont reliées par : $K(s) \approx -\log_2 p(s)$. »

Si l'on admet que, dans l'Univers, toute interaction est assimilable au calcul d'un programme, il s'ensuit, selon le théorème de codage, que l'on observera avec une plus grande fréquence des structures simples (droites, cercles, sphères, cubes…) que des structures complexes (ce rocher ici, ce nuage aujourd'hui dans le ciel, etc.).

Le théorème suggère une généralisation de la complexité de Kolmogorov lui donnant un sens pour des fichiers numériques courts. Pour calculer la probabilité $p(s)$ d'une suite s, on utilise un très grand ensemble de machines élémentaires (par exemple des machines de Turing, *voir l'encadré 1*). On lance chaque machine sur un ruban dont, initialement, toutes les cases portent un 0 et l'on examine (après qu'elle s'est arrêtée) la suite composée de 0 et de 1 qu'elle a écrite sur les cases visitées du ruban : certaines machines donnent la séquence 000, d'autres la séquence 0100111, etc. La distribution des fréquences des séquences produites donne une approximation de la probabilité $p(s)$ pour les séquences les plus courtes. Par application de la formule reliant $p(s)$ et $K(s)$, on en tire une valeur de la complexité.

26 000 MILLIARDS DE MACHINES

En 2014, en utilisant les 26 559 922 791 424 machines de Turing à cinq états pouvant produire des séquences de 0 et de 1, Fernando Soler-Toscano, de l'université de Séville, Hector Zenil, de l'université d'Oxford, Nicolas Gauvrit, de l'École pratique des hautes études, et moi-même avons mené un immense calcul. Ces 26 000 milliards de machines sont assimilables aux programmes les plus simples et leur fonctionnement fournit l'approximation attendue de la complexité pour les suites courtes de 0 et de 1. Le calcul indique par exemple un classement, avec *ex æquo*, pour les 24 séquences les plus simples (la complexité de *xxx* étant ici notée '*xxx*') :

'0' = '1' < '00' = '01' = '10' = '11' < '000' = '111' < '001' = '011' = '100' = '110' < '010' = '101' < '1111' = '0000' < '0001' = '0111' = '1000' = '1110' < '0010' = '0100' = '1011' = '1101' < …

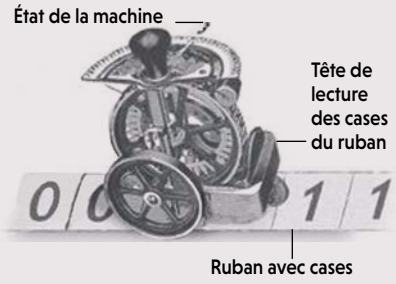
En ne considérant que les séquences de longueur 7 (il y en a $2^7 = 128$), le classement par complexité croissante mesurée par les 26 000 milliards de machines est donné dans l'encadré ci-contre. Un classement non limité >

1 DES MILLIARDS DE MACHINES DE TURING

Une machine de Turing comporte une tête de lecture-écriture se déplaçant sur un ruban découpé en cases, cases où sont écrits des symboles (par exemple des 0 et des 1). En fonction de son état interne, pris dans un ensemble fini d'états possibles, et de ce qu'elle lit sous sa tête, un 0 ou un 1, la machine se déplace vers la droite, vers la gauche ou s'arrête, après avoir réécrit le symbole lu sur le ruban et changé d'état. Partant d'un ruban couvert de 0 *(voir le schéma ci-contre)*, une machine donnée calcule indéfiniment ou produit une séquence de symboles et s'arrête (la production de la machine ne prend en compte que les cases où elle est passée). Elle produit par exemple la séquence 0101010 avant de s'arrêter. Si elle ne s'arrête pas, on ne prend pas en compte son calcul.

Le nombre de machines différentes à n états est $(4n + 2)^{2n}$, ce qui pour $n = 5$ donne 26 559 922 791 424 machines différentes. Fernando Soler-Toscano a fait fonctionner toutes ces machines sur le ruban initial composé de 0 et examiné ce qu'elles produisaient ; cela a pris 18 jours aux supercalculateurs du Centre d'informatique scientifique d'Andalousie. Pour $n = 6$, le même calcul serait environ 10 000 fois plus long, ce qui est inenvisageable aujourd'hui.

Considérons par exemple les 128 séquences de longueur 7 produites par ces machines, et classons-les en fonction du nombre de fois qu'elles ont été obtenues.



Cela donne le tableau ci-dessous. Les séquences d'une même ligne ont été obtenues le même nombre de fois, et les séquences les plus fréquentes sont en tête. Les séquences se groupent en 36 paquets comportant chacun 2 ou 4 séquences. Comme le théorème du codage l'annonçait, on observe que la complexité des séquences s'accroît d'un paquet au suivant. Dans chaque paquet, on note que les séquences ont la même structure.

D'autres classements de séquences ont été réalisés par la même méthode en considérant des machines de Turing utilisant plus de deux symboles ou opérant sur un plan quadrillé au lieu d'un ruban *(voir l'encadré 2)*.

01	0000000	1111111			19	0100010	1011101		
02	0000001	0111111	1000000	1111110	20	0010100	1101011		
03	0101010	1010101			21	0110110	1001001		
04	0000010	0100000	1011111	1111101	22	0001100	0011000	1100111	1110011
05	0000100	0010000	1101111	1111011	23	0011010	0101100	1010011	1100101
06	0001000	1110111			24	0100110	0110010	1001101	1011001
07	0000011	0011111	1100000	1111100	25	0111110	1000001		
08	0100101	0101101	1010010	1011010	26	0000111	0001111	1110000	1111000
09	0010010	0100100	1011011	1101101	27	0010110	0110100	1001011	1101001
10	0000110	0110000	1001111	1111001	28	0001101	0100111	1011000	1110010
11	0001010	0101000	1010111	1110101	29	0010011	0011011	1100100	1101100
12	0010001	0111011	1000100	1101110	30	0011101	0100011	1011100	1100010
13	0010101	0101011	1010100	1101010	31	0011001	0110011	1001100	1100110
14	0101001	0110101	1001010	1010110	32	0110001	0111001	1000110	1001110
15	0000101	0101111	1010000	1111010	33	0011110	0111100	1000011	1100001
16	0001001	0110111	1001000	1110110	34	0001110	0111000	1000111	1110001
17	0101110	0111010	1000101	1010001	35	0011100	1100011		
18	0100001	0111101	1000010	1011110	36	0001011	0010111	1101000	1110100