El número más grande que puede escribirse en una pizarra

Cómo se pueden escribir números finitos grandísimos con números infinitos que ocupan muy poco en la pizarra

Agustín Rayo

Imagínese el duelo siguiente: dos contendientes; una pizarra; quien escriba el número más grande gana. A eso, ni más ni menos, me desafió mi amigo Adam Elga, ahora profesor de la Universidad de Princeton.

Fijamos algunas reglas. Decidimos, en primer lugar, que estableceríamos turnos escribiendo números sobre la pizarra, y que el último en presentar una entrada válida ganaría. En segundo lugar, acordamos que sólo los números *finitos* contarían como entradas válidas. El número 17 contaría como entrada válida, por ejemplo, o el número $10^{10^{10^{10^{10}}}}$, pero no el número ω, o el ω + ω.

(El número omega, ω, es el primero de los ordinales *transfinitos*. Viene justo después de los ordinales finitos: 0, 1, 2, 3, ... A ω le siguen ω + 1, ω + 2,... Después vienen ω + ω, ω + ω + 1, ω + ω + 2, ... luego ω + ω + ω, ω + ω + ω + 1, ω + ω + ω + 2, ..., y así sucesivamente hasta llegar a ω × ω, (ω × ω) + 1, (ω × ω) + 2, ... ¡Y ese es sólo el principio! Los teóricos de conjuntos suelen identificar un ordinal con el conjunto de sus predecesores. En particular: 0 = { }, 1 = {0}, 2 = {0, 1}, 3 = = {0, 1, 2},... ω = {0, 1, 2, 3,...}, ω + 1 = = {0, 1, 2, 3,..., ω} y ω + ω = {0, 1, 2, 3,..., ω, ω + 1, ω + 2, ω + 3,...}.)

En tercer lugar, decidimos que todo vocabulario *semántico* —es decir, palabras como 'nombra', 'remite a', o 'significa'— quedaría vedado. La restricción es crucial. Sin ella, Adam hubiera estado en posición de escribir sobre la pizarra 'el número más grande que Agustín nombrará en su vida, más uno', y cantar victoria en la primera ronda. Peor todavía: yo hubiera estado en posición de responder escribiendo 'el número más grande que Adam nombrará en su vida, más uno', dejándonos con un problema. (Si está bien definido lo que dice Adam, el número que nombra tiene que ser más grande que cualquiera que nombraré yo; y si está bien definido lo que digo yo, el número que nombro tiene que ser más grande que cualquiera que nombrará él. Por tanto, alguno de los dos tiene que haber dicho algo que no está bien definido.)

Por último, hicimos un acuerdo de caballeros. Cada vez que uno de nosotros escribiera un número sobre la pizarra, el otro debería reconocer la derrota o responder con un número *mucho* más grande. ¿Cuánto más grande? Tan grande como para que sea imposible, en términos prácticos, rebasarlo utilizando sólo métodos introducidos antes en la lid. Eso significa, por ejemplo, que si la última entrada de Adam fue '$10^{10^{10}}$', no sería de caballeros que yo respondiera con '$10^{10^{10}} + 1$', o con '$10^{10^{10^{10}}}$'. (Estos dos números pueden ser fácilmente rebasados, en términos prácticos, utilizando un método ya introducido; en este caso, la iteración de exponenciales.) Queríamos que el duelo fuera una guerra de originalidad, no una guerra de paciencia.

Establecidas las reglas, nos citamos en el auditorio de mi centro de trabajo, el Instituto de Tecnología de Massachusetts (o MIT, por sus siglas en inglés). En casi cualquier otro lugar, habría sido difícil encontrar público para un evento de este tipo. Pero en el MIT el auditorio estaba lleno a reventar. En cuanto entré y vi tanta gente, comencé a ponerme nervioso. Conozco a Adam desde que éramos alumnos de doctorado —también en el MIT— y años de experiencia me han enseñado que cuando de acertijos lógicos se trata, Adam es una fiera. Pero el árbitro del duelo no me dio tiempo de arrepentirme. "En esta esquina" —comenzó a vocear— "Adam... el Loco Bayesiano... ELGAAA. En esta otra, Agustín... el Multiplicador Mexicano... RAYOOO".

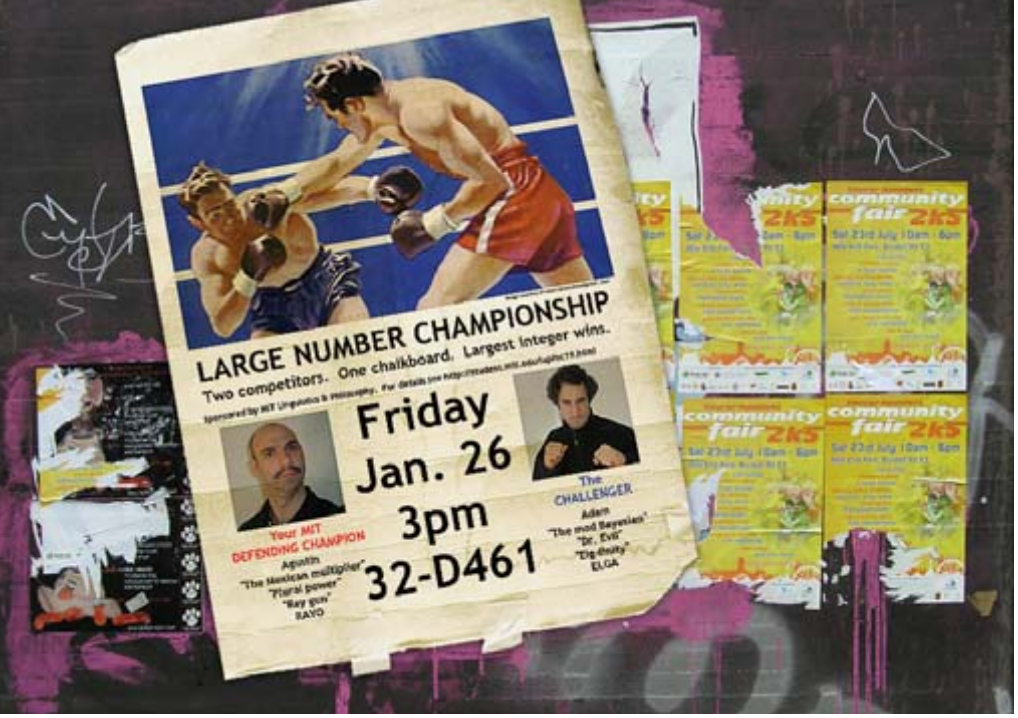
El concurso había comenzado. Dado que estaba compitiendo en campo propio, fui el primero en enfrentarme a la pizarra. Sin pensarlo mucho, escribí una secuencia de treinta o cuarenta unos: '1111 111111111111111111111111111'. "Estamos apenas calentado motores", pensé. Pero mi primer esfuerzo resultó desastroso. Adam se acercó a la pizarra y trazó con su borrador una línea a través de todos mis unos excepto los primeros dos, dejando lo siguiente: '11!!!!!!!!!!!!!!!!!!!! !!!!!!!!!'. "Once factorial, factorial, factorial, ..." declaró triunfante. El público se volvió loco de entusiasmo. $n!$ (o n factorial) es $n \times (n-1) \times ... \times 2 \times 1$. Eso significa que 11! es 39.916.800, un número cercano a la población de España; 11!! es aproximadamente $6 \times 10^{286.078.170}$ (muchísimo más que el número de partículas en el universo: 10^{87}, de acuerdo con una estimación generosa); y 11!!! es tan grande, que resulta imposible, en términos prácticos, aproximarlo usando una expresión de la forma '10^n', donde 'n' es un numeral en base 10 (y, por tanto, imposible, en términos prácticos, aproximarlo escribiendo una secuencia de unos sobre la pizarra). ¡El número de Adam —11 seguido de 30 o 40 factoriales— era *mucho* más grande que el mío!

Por fortuna, recordé la función del castor laborioso, CL(n). Aunque todo programa de cómputo consiste en una secuencia finita de símbolos, hay algunos programas que no se detendrían nunca, una vez puestos en marcha. Un programa que consista, por ejemplo, en la instrucción "Imprime un uno en pantalla; repite." comenzaría a escribir unos en pantalla y no se detendría nunca. Digamos que un programa de cómputo tiene *productividad k* si acaba por detenerse y antes de detenerse imprime en pantalla una secuencia de k unos. La función del castor laborioso es una medida de la productividad. Fijemos un lenguaje de programación: BASIC, por ejemplo. Entonces CL(n) es la productividad del más productivo de los programas BASIC escritos

¿Quieres saber más?

La idea del concurso nos vino de leer este artículo de Scott Aaronson: <http://www.scottaaronson.com/writings/bignumbers.html>

Aquí hay más detalles sobre el número ganador, y referencias adicionales: <http://web.mit.edu/arayo/www/bignums.html>



CORTESIA DE WWW.INTERNATIONALPOSTER.COM (*ilustración superior en el poster*); DISEÑO DEL POSTER ADAM ELGA

con *n* símbolos o menos. Mi siguiente entrada en el concurso fue CL(10^{100}), la productividad del más productivo de los programas BASIC escritos con un *googol* de símbolos o menos.

¿Cómo se compara este número con la entrada de Adam? Es posible escribir en unas cuantas líneas un programa BASIC que compute la función factorial y que imprima en notación unaria (es decir, como una secuencia de unos) el resultado de aplicarle esta función al número 11 treinta o cuarenta veces. No sé cuál será el programa más corto que lo haga, pero sin duda tiene menos de 10^3 símbolos. De esto se sigue que CL(10^3) —un número mucho menor que CL(10^{100})— es por lo menos del tamaño del número de Adam. (De hecho, es muchísimo mayor.) CL(10^{100}) es un número gigantesco. Por grande que sea un número *g*, mientras sea posible escribir en la práctica un programa BASIC que se detenga habiendo escrito en pantalla *g* unos o más, podemos estar seguros de que CL(10^{100}) será más grande que *g*.

Durante las siguientes rondas, el duelo se convirtió en una búsqueda de lenguajes de programación cada vez más potentes. Una característica del BASIC es que no existe ningún programa BASIC que permita determinar, para un programa BASIC arbitrario, si acabará o no por detenerse. (Lo mismo es cierto de cualquier otro lenguaje de programación que seamos capaces, en términos prácticos, de poner en marcha.) De tal limitación se sigue que no es posible escribir en BASIC un programa que compute CL(*n*). Pero pensemos en el lenguaje BASIC^1. BASIC^1 sería exactamente como BASIC, con una salvedad: tiene un *oráculo*, una operación primitiva que le permite determinar instantáneamente si un programa en BASIC (sin oráculos) se detendría o no. (Aquí no es necesario que sea posible *construir* un oráculo. Basta con que el concepto abstracto de un oráculo esté bien definido.) En BASIC^1 sí es posible escribir un programa que compute CL(*n*). Por tanto, hay programas en BASIC^1 que son mucho más productivos que cualquier programa en BASIC de igual o menor longitud; en consecuencia, la función CL^1(*n*) —la función del castor laborioso aplicada a BASIC^1— puede usarse para expresar números mucho mayores que CL(10^{100}). Un ejemplo es CL^1(10^{100}).

Resulta ser que ningún programa en BASIC^1 puede computar CL^1(*n*). Para computar esta función habría que reforzar BASIC^1 con un oráculo para BASIC^1. Se tendría así un nuevo lenguaje, BASIC^2, con su correspondiente función del castor laborioso, que permitiría escribir en la práctica números mucho mayores que CL^1(10^{100}). Y así sucesivamente. Conforme pasaron las rondas, Adam y yo generamos una jerarquía de funciones del castor laborioso. Miembros subsecuentes de la jerarquía nos permitieron escribir números más y más grandes. Primero usamos niveles finitos de la jerarquía: CL(*n*), CL^1(*n*), CL^2(*n*),... Luego, niveles transfinitos: CL$^\omega$(*n*), CL$^{\omega+1}$(*n*), CL$^{\omega+2}$(*n*),..., CL$^{\omega+\omega}$(*n*),..., CL$^{\omega+\omega+\omega}$(*n*),..., CL$^{\omega^\omega}$ (*n*),... CL$^{\omega^{\omega^\omega}}$ (*n*),...

Importa advertir que, aunque utilizamos números infinitos para construir la jerarquía, y aunque la jerarquía misma es infinita, el resultado de aplicarle un miembro de la jerarquía a un número finito (10^{100}, por ejemplo) es siempre un número finito y, por tanto, una entrada válida en la lid.

Algo interesante había sucedido. Nuestro esfuerzo por descubrir números *finitos* cada vez mayores nos había llevado a buscar números *infinitos* cada vez mayores. La jerarquía que logramos construir en la competencia llegó hasta CL$^\theta$(*n*), donde θ es el primer ordinal no recursivo —un número transfinito relativamente pequeño. (Los ordinales recursivos son aquellos cuya construcción puede especificarse utilizando un programa de ordenador sin oráculos, como BASIC.)

Mis amigos teóricos de conjuntos me aseguran que la conexión entre números transfinitos y números finitos va mucho más allá. Si existiera, por ejemplo, un cardinal *medible* —un número transfinito de tamaño verdaderamente monstruoso, cuya existencia no está garantizada por los axiomas usuales de la teoría de conjuntos— existirían también oráculos muchísimo más poderosos que cualquiera de los que construimos Adam y yo, oráculos que permitirían escribir en términos prácticos números mucho mayores que CL$^\theta$(10^{100}).

La última entrada en el concurso, la entrada ganadora, fue un número más grande todavía. A saber: el número más pequeño con la propiedad de ser mayor que cualquier número que pueda ser nombrado en el lenguaje de la teoría de conjuntos —un lenguaje en el que puede expresarse el 99,9 % de la matemática contemporánea— usando 10^{100} símbolos o menos. Esta descripción del número ganador no hubiera sido válida en el concurso porque incluye la palabra 'nombrado', que cuenta como vocabulario semántico. Pero en el concurso utilizamos una descripción de ese mismo número que no requiere vocabulario prohibido. El secreto consiste en usar un lenguaje de *segundo orden*: un lenguaje que permite expresar no sólo la cuantificación singular ("existe un número tal que ese número es tal cosa"), sino también la cuantificación *plural* ("existen unos números tales que esos números son, colectivamente, tal cosa").

Los lenguajes de segundo orden son mucho más expresivos que los de primer orden. ¿Y si tuviéramos un lenguaje más expresivo todavía? Entonces podríamos referirnos, en términos prácticos, a números más grandes todavía... ¿Qué se le ocurrirá a Adam la próxima vez?