En el "Ángulo de Devlin"

La Columna de Keith Devlin en la MAA.[1]

Traducida y anotada por Diego Pareja Heredia. ***Universidad del Quindío.***

Mayo de 2009[2]

El Computador más Pequeño del Mundo

Mucho antes que aparecieran el iPod Nano y los portátiles, aun mucho antes que naciera Bill Gates, un matemático británico probó que era posible construir computadores capaces de ser programados para llevar a cabo cualquier tipo de computación. Este matemático fue Alan Turing y el artefacto teórico que inventó en la década de los años de 1930 se conoce hoy como la máquina de Turing. Para ser exactos, hay una familia completa de máquinas hipotéticas de computación. Para una introducción sobre este tema se puede visitar: **http://en.wikipedia.org/wiki/Turing_machine**

La segunda guerra mundial le dio a Turing la oportunidad de poner en práctica su teoría, pues pasó los años de la guerra en Bletchley Park (la institución británica dedicada a quebrar códigos secretos) construyendo un computador real para decodificar los mensajes de la codificadora Enigma de los alemanes, hecho que se constituiría en un punto decisivo en el curso de la guerra.

---

[1] La columna original del mes de mayo se puede leer en: **http://www.maa.org/devlin/devlin_05_09.html**

[2] Esta columna apareció en Noviembre de 2007, pero por su interés la reproduzco aquí. La columna original en inglés puede leerse en: **http://www.maa.org/devlin/devlin_11_07.html**

El computador construido en Bletchley Park[3], como todos los computadores de allí en adelante, es mucho más complicado que una máquina teórica de Turing. En principio usted o yo podríamos construir físicamente una máquina de Turing y en efecto varias personas lo han hecho, pero no sería útil en un sentido práctico, pues le tomaría decenas o cientos de años programarla para que efectúe cálculos útiles; peor aun, resolver ciertos problemas le tomaría mucho más. Ellas pueden ser simples, pero no son prácticas. Su importancia radica en la posibilidad que nos dan de entender qué es una computación y cómo opera. (Por ejemplo, la seguridad del número de las tarjetas de crédito cuando se ordena una compra en línea depende crucialmente de las matemáticas involucradas en las máquinas de Turing).

La idea de construir máquinas para hacer aritmética (en particular) es tan antigua como las matemáticas mismas. Los primeros intentos, tales como arrumes de piedrecillas o el ábaco, no hacen en realidad ninguna operación matemática, ellos simplemente permiten una forma conveniente de almacenar los números que son manipulados por el operario, quien es en realidad el que hace las matemáticas. Blas Pascal (1623-1662) fue una de las primeras personas en diseñar y construir una máquina, la Pascalina, que si realizaba cálculos. Charles Babbage (1791 - 1871) fue otro pionero famoso en la historia del diseño de máquinas calculadoras. Pero todos los intentos iniciales estaban enfocados al diseño de máquinas con un propósito específico. Nadie había pensado en construir máquinas que pudieran hacer cualquier tipo de computación.

Parte de la razón está en que no fue sino hasta los años de 1930 que alguien trató de especificar el sentido exacto de lo que se quiere significar por “computación”. Turing inventó el concepto de “Máquina de Turing” con el propósito de formular una definición precisa: una función de números a números (digamos) es “computable” si ésta puede calcularse con el recurso de una máquina de Turing.

De primerazo, esta definición parece excesivamente estrecha. Las máquinas de Turing son artefactos de cálculo demasiado simples. ¿Pero qué hay sobre funciones que pueden calcularse, pero sólo con el recurso de una máquina más complicada que una máquina de Turing? Bien, hay algo divertido en torno a este punto. Ni en el tiempo de Turing, ni posteriormente hasta ahora, alguien ha producido un simple ejemplo de una función que sea obviamente (probable) computable (por uno u otro artefacto) que no pueda ser calculado por una máquina de Turing. Más aun, alrededor del mismo tiempo en que Turing desarrollaba sus trabajos, otros matemáticos (entre ellos Kurt

[3] Para otros detalles de los aportes de Turing a los computadores ver mi artículo *Breve Historia del Computador en*: http://www.matematicasyfilosofiaenelaula.info/historiam.htm

Gödel, Stephen Kleene y Alonzo Church) formularon definiciones formales alternativas de “funciones computables” y al final todas resultaron ser equivalentes a la noción introducida por Turing. De este modo se ha establecido un consenso entre los matemáticos de que la definición de Turing captura nuestras más intuitivas nociones de lo que significamos por computable. Este consenso, que aun permanece hoy, es conocido como la Tesis de Church- Turing. Se reemplaza así, una noción intuitiva con otra noción precisa, definida matemáticamente.

Un resultado importante que probó Turing acerca de su nuevo concepto fue que no era necesario especificar una máquina de Turing en particular para cada computación dada. Es posible construir lo que el llamó una “máquina universal de Turing”, que pudiera interpretar un “programa” introducido en ella como datos a efecto de calcular cualquier función computable que se le presente. Hoy estamos familiarizados por supuesto con la idea de que los computadores son generalmente programables, procesando palabra un minuto, al siguiente haciendo matemáticas y luego bajando canciones del Internet. Pero en el pasado, en el tiempo de Turing, esto fue una idea novedosa, aunque solamente en el sentido teórico.

Una pregunta fascinante acerca de la máquina universal de Turing es: ¿qué tan simple una máquina universal de Turing puede ser (y que sea capaz de realizar toda computación posible)? En los años de 1950 y 1960, mucho esfuerzo se hizo a fin de probar que una máquina de Turing con dos estados internos, y computación sobre sólo dos símbolos, no puede ser universal. (Los computadores reales de hoy en día trabajan binariamente (con dos símbolos), pero tienen un gran número posible de estados internos.) Marvin Minsky, el pionero de la ciencia de la computación en el MIT construyó una máquina universal de Turing con siete estados y con cuatro símbolos de computación.

Entonces, en la década de 1990, el matemático Stephen Wolfram, mejor conocido por su programa *Mathematica,* logró construir una máquina universal de Turing con dos estados y con cinco símbolos de computación. El también propuso una candidata a ser aún más simple, una que podría ser la más simple posible. La máquina de Wolfram tiene dos diferentes estados internos y opera con sólo tres símbolos. Se la conoce como la Wolfram 2,3.

Wolfram tenía razón al creer que su computador era en efecto capaz de realizar cualquier computación, pero no fue capaz de probarlo. Él describió el problema en su libro de 2002 *Una Clase Nueva de Ciencia* y eso condujo a varios intentos por encontrar una prueba, pero nadie lo logró. Comenzando el año de 2007, Wolfram ofreció un premio de US$25000 para la primera persona que resolviera el problema.

Justamente unos pocos meses después, un estudiante universitario de veinte años en Birmingham, Inglaterra, Alex Smith, encontró la solución. Su prueba, la que se encuentra en la Web, cubre unas cincuenta páginas de razonamientos matemáticos.

Alex es el mayor de tres hermanos. Sus padres son ambos profesores de la Universidad de Birmingham, donde Alex es estudiante de pregrado en electrónica e ingeniería eléctrica y de sistemas. Él empezó a usar computadores desde cuando tenía seis años de edad, y conoce alrededor de veinte lenguajes de programación.

Se puede hallar los archivos PDF de la prueba de Smith, más un montón de detalles sobre el problema de Wolfram, en el sitio Web de la Wolfram: **www.wolframscience.com/prizes/tm23/**