Análisis y Diseño de Algoritmos

Héctor Jiménez Salazar
Jesús Lavalle Martínez
hjimenez@fcfm.buap.mx jlavalle@aleteya.cs.buap.mx

agosto de 2002

El 10 de Agosto de 2002 murió E.W. Dijkstra, un físico dedicado a la computación hace más de cuarenta años. Él propuso el algoritmo de camino mínimo para recorrer un mapa, el sistema operativo de multiprogramación "THE", las bases de la programación estructurada, y participó en la definición del lenguaje de programación ALGOL, entre otras contribuciones. El hecho es significativo no sólo por la pérdida humana sino por el coincidente fin del liderazgo de la Ciencia de la Computación en el manejo de la información, tal como J. Denning dio a entender en febrero de 2001: es una ironía que quien dio vida a la computación pase a ser una parte más de lo que hoy se llama la Tecnología de la Información (TI). Así, ha terminado el ciclo del establecimiento de los principios fundamentales de una ciencia. La tecnología que ésta ha generado no sólo avanza con más rapidez, sino que impone la línea de desarrollo. Los profesionistas de la TI se apoyan indiscutiblemente en las bases desarrolladas por los científicos de la computación, aquéllos usan los sistemas operativos, protocolos de comunicación, lenguajes de programación, sistemas de bases de datos, sistemas de ayuda al diseño, etc, para muy diversos fines que la sociedad demanda, muy probablemente sin conocer detalles fundamentales que garantizan el funcionamiento de los sistemas construidos. Es un hecho, al igual que la Ingeniería Hidráulica se apoya en la Física, la TI se apoya en la Ciencia de la Computación, aun la última con su núcleo en la Teoría de la Computación.

En el inicio de la Computación la TI era parte de la primera, sin diferenciar todavía la explosión de áreas que fueron surgiendo. Bien que la Matemática ha dado la pauta para la solución de problemas de muy diversas disciplinas, en muchas aplicaciones había que desarrollar el algoritmo pues éste no existía. Con el tiempo se reunió una cantidad considerable de algoritmos que abarcan las más variadas aplicaciones, o que pueden ser adaptados con facilidad. Al mismo tiempo se fueron planteando preguntas fundamentales de la Ciencia de la Computación como el asegurar que un programa termine (decidibilidad). O bien, si sabemos que en ciertas condiciones el algoritmo encontrará un resultado desearíamos saber si éste se obtiene en un tiempo razonable (tratabilidad). Y, aunque pueda ser razonable el tiempo en que un algoritmo termine, nos preguntamos si ese tiempo puede ser reducido (ubicarlo en una clase de complejidad menor). Éstas son cuestiones eminentemente de la Ciencia de la Computación pero, como dijimos, al proporcionar más piezas de diferentes características (algoritmos) podemos armar muy sofisticados e insospechados rompecabezas (sistemas), lo cual, con base en que una computadora es un sistema que habilita el funcionamiento de otros sistemas (máquina universal), este proceso demandó herramientas que ayudaran a la tarea de no únicamente ensamblar rompecabezas, además, acercándose a las preguntas fundamentales, se concibió asegurar que el rompecabezas construido funcionara como era esperado. Lo anterior refiere a procesos de ingeniería, que, como sabemos, en computación se llama Ingeniería de Software. Particularmente, de la garantía de que un programa funcione se ocupa, con un fuerte sesgo teórico, la Especificación Formal. Así como la Lógica Matemática creó PROLOG, fueron propuestos, para la Ingeniería de Software, lenguajes que permitieran especificar sistemas formalmente. Con este ejemplo tenemos una muestra del tejido entre Ciencias de la Computación y la TI.

Al parecer el Análisis y Diseño de Algoritmos estaría condenado a la extinción pues ya hay muchos algoritmos, e incluso código en clases para los lenguajes orientados a objetos, sin embargo cada día se reafirma la idea de C. Babbage que se puede expresar como: siempre habrá necesidad de crear un algoritmo más veloz para una nueva máquina. Esto es, un cientŕfico de la Computación no puede prescindir de saber cómo se analiza un algoritmo, o cómo se adapta o se diseña uno para nuevas condiciones, ya que una de sus tareas es crear algoritmos. La sentencia de Babbage no pierde vigencia pues no debe olvidarse que aún cuando en el término de un año pueda triplicarse, con el Hardware, la velocidad de cómputo el descenso de la complejidad de un algoritmo puede representar una mejora no en múltiplos de velocidad sino en órdenes de magnitud. Precisemos esto un poco más. El aumento de velocidad de una computadora afecta a todos los programas que ésta ejecute, mientras que el mejoramiento "real" de un algoritmo solamente afecta a los programas que lo usen. Aclarado lo anterior, nos permitimos hacer la siguiente comparación: puede decirse que en 4 años la velocidad de cómputo aumenta por un factor de cuatro, en tanto puede afirmarse que como fruto del estudio de algunos algoritmos "sencillos" en cinco años la velocidad aumenta por un factor de cien. Pero, a diferencia del aumento de la velocidad ganada por una nueva computadora (que es un múltiplo de la velocidad previa), la velocidad mejorada por un algoritmo no se debe a un factor fijo sino a uno variable o mejor dicho a una función. Esto es, si aumentamos nuestros datos de entrada al algoritmo diez veces más entonces, tendremos un factor de velocidad de mil y no de cien. Supuesto que cada vez se demanda el procesamiento de más datos, en conclusión, el Análisis y Diseño de Algoritmos puede verse como una forma de mejorar parte del de la máquina con una "tecnología" que se desarrolla potencialmente en una pequeña fracción del tiempo que requiere el desarrollo de las nuevas computadoras (para nuestro ejemplo, el Hardware necesitaría aproximadamente 250 años para lograr la velocidad de un algoritmo mejorado).

Empecemos entonces a estudiar un tema que está en la frontera de la Ciencia Computacional y muchas áreas de la TI.

1. Bases matemáticas

Siempre una necesidad ha sido asegurar, bajo ciertas condiciones, un suceso. Este principio ha guiado el desarrollo de la ciencia, y también de otros sistemas. En términos prácticos debe asugurarse, por ejemplo, que un puente no se caiga, que no haya corrupción, que un programa obtenga resultados en cierto periodo de tiempo, etc.

Aunque estamos acostumbrados a vivir en un mundo çausa-efecto", es decir que para conseguir algo sabemos qué debe hacerse, no ha sido así todo el tiempo. Estas reglas han sido decubiertas o convenidas a lo largo de muchas experiencias. En los mundos "desconocidos"debemos, antes de resolver problemas, experimentar y aprender. Para este propósito suele ser útil la experiencia ganada en problemas semejantes.

La matemática ha sido lo común en la solución de muchos problemas. A través de los modelos, la matemática ha proporcionado una forma de ver el problema y predecir su comportamiento en ciertas condiciones.

La modelación nos permite aplicar un método a un problema que satisface ciertas condiciones. Suficiente para muchos problemas resulta aplicar algún método matemático. Sin embargo, tendrá que conocerse, al menos, el fundamento de dicho método para aplicarlo a nuevas situaciones. Esto conlleva a ejercer el razonamiento matemático.

El razonamiento matemático es una vivencia de convencimiento sobre abstracciones. El ejercitador matemático no sólo asiente sus expresiones, tiene argumentos, que se basan en las convenciones establecidas, para sostenerlas. Esta explicación del "por qué"se afirma algo se llama demostración. El lector de matemáticas habrá de consentir igualmente la explicación del escritor o, en su caso, refutarla en el mismo sistema de convenciones.

Estableceremos un modelo que represente el *comportamiento* de un algoritmo para llevar a cabo su análisis. Esto lo haremos en la sección dos, por lo pronto presentaremos las herramientas necesarias