Problemas computacionales - Introducción

Decir que un problema se puede resolver algorítmicamente implica, informalmente, que es posible escribir un programa de computadora que producirá la respuesta correcta para cualquier entrada si permitimos que se ejecute durante el tiempo suficiente y le proporcionamos todo el espacio de almacenamiento que necesite.

En la década de 1930, antes de la llegada de las computadoras, los matemáticos trabajaron con gran celo para formalizar y estudiar el concepto de algoritmo, que entonces se definía de manera informal como un conjunto claramente especificado de instrucciones sencillas a seguir para resolver un problema o calcular una función.

Se idearon e investigaron varios modelos de cómputo formales. Muchos de los primeros trabajos en este campo, llamado *teoría de la computabilidad*, hicieron hincapié en describir o caracterizar los problemas que se podían resolver algorítmicamente, y en presentar algunos problemas que no se podían resolver de esa manera.

Uno de los resultados negativos importantes, establecido por Alan Turing, fue la demostración de la insolubilidad del "problema del paro". Este problema consiste en determinar si cualquier algoritmo (o programa de computadora) determinado llegará en algún momento a un punto en el que se detendrá (en vez de, digamos, entrar en un ciclo infinito) al trabajar con una entrada dada. No puede existir un programa de computadora que resuelva este problema.

Aunque la teoría de la computabilidad tiene implicaciones obvias y fundamentales para las ciencias de la computación, el saber que en teoría un problema se puede resolver con una computadora no basta para decirnos si resulta práctico hacerlo o no. Por ejemplo, se podría escribir un programa perfecto para jugar ajedrez.

La tarea no sería demasiado difícil; las formas de acomodar las piezas de ajedrez en el tablero son finitas, y bajo ciertas reglas una partida debe terminar después de un número finito de movimientos. El programa podría considerar todos los movimientos posibles que podría efectuar la computadora, cada una de las posibles respuestas del oponente, cada una de sus posibles respuestas a esos movimientos, y así hasta que cada una de las sucesiones de posibles movimientos llegara a su fin. Entonces, dado que la computadora conoce el resultado final de cada movimiento, podrá escoger la mejor.

Según algunas estimaciones, el número de acomodos distintos de las piezas en el tablero que es razonable considerar (mucho menor que el número de sucesiones de movimientos) es de aproximadamente 10^{50}. Un programa que las examinara todas tardaría varios miles de años en ejecutarse. Es por ello que no se ha ejecutado un programa semejante.

Es posible resolver una gran cantidad de problemas con aplicaciones prácticas —es decir, se pueden escribir programas para hacerlo— pero las necesidades de tiempo y almacenamiento son demasiado grandes para que tales programas tengan utilidad práctica.

Es evidente que las necesidades de tiempo y espacio de los programas son importantes en la práctica; por ello, se han convertido en el tema de estudios teóricos en el área de las ciencias de la computación llamada *complejidad computacional*.

Una rama de tales estudios, se ocupa de establecer una teoría formal y un tanto abstracta de la complejidad de las funciones computables. (Resolver un problema equivale a calcular una función que a partir del conjunto de entradas proporcione el conjunto de salidas.) Se han formulado axiomas para medir la complejidad, éstos son básicos y lo bastante generales como para poder usar el número de instrucciones ejecutadas o bien el número de bits de almacenamiento que ocupa un programa como medida de su complejidad.

Utilizando esos axiomas, podemos demostrar la existencia de problemas arbitrariamente complejos y de problemas para los que no existe un programa óptimo.