Problemas P y NP-Completo

Alumno: López Mendoza Ricardo

¿Qué es P?

Hasta principios del sXX se creía que todo problema matemático se podía resolver siguiendo una serie de pasos determinados. No se cuestionaba la dificultad de encontrar dichos pasos, pero su existencia se daba por supuesta fuese cual fuese el problema, no fue hasta bien entrado el sXX que David Hilbert formalizó ese problema y planteó tres cuestiones fundamentales:

1-. ¿Son las matemáticas completas? ¿Es decir, se puede probar que cualquier sentencia (matemática) es cierta o falsa?

2-. ¿Son las matemáticas consistentes? ¿Es decir, no se puede probar como cierto algo falso?

3.-Son las matemáticas decididles (o computables)? ¿Es decir, se puede probar que cualquier sentencia (matemática) es cierta o falsa siguiendo un número finito de pasos?

Cuando Hilbert formalizó las tres preguntas su intuición era que la respuesta a las tres era Sí. Es decir, las matemáticas, el lenguaje universal, eran completas, consistentes y decididles. Poco duró esa esperanza, porque las dos primeras suposiciones las desmontó Gödel, cuando formuló su teorema de incompletitud donde venía a decir que las matemáticas no podían ser a la vez consistentes y completas.

Una máquina de Turing es una construcción teórica, concretamente un modelo matemático en forma de autómata capaz de ejecutar cualquier algoritmo. tiene una entrada (un símbolo) y un estado (que va cambiando en función del estado actual y el símbolo de entrada), A esta máquina de Turing la llamamos máquina de Turing determinista y ejecuta algoritmos deterministas. Los algoritmos deterministas son con los que normalmente trabajamos: les das una entrada, realizan ciertas cosas y devuelven un resultado.

En computación, cuando el tiempo de ejecución de un algoritmo (mediante el cual se obtiene una solución al problema) es menor que un cierto valor calculado a partir del número de variables implicadas (generalmente variables de entrada) usando una fórmula polinómica, se dice que dicho problema se puede resolver en un tiempo polinómico.

P es la clase de complejidad que contiene problemas de decisión que se pueden resolver en un tiempo polinómico contiene a la mayoría de los problemas naturales, algoritmos de programación lineal, funciones simples etc.

EJEMPLOS

Calcular el cuadrado de un número es un problema de tipo P (hay un algoritmo en tiempo polinomial para hacerlo: multiplicar el número por si mismo). La verificación también sencilla. ¿Es y el cuadrado de x? Nos basta con hacer la raíz cuadrada de y verificando que su valor es x.

Problemas NP-completos

Por otra parte, tenemos el grupo de problemas NP. Este grupo está formado por aquellos problemas que podemos verificar en tiempo polinomial pero que para solucionarlos puede ser que requiramos un algoritmo de orden superior.

Los problemas NP-completos, en la cual se agrupan los problemas más costosos de la clase NP, se demostró que cualquier problema NP se puede reducir a otro tipo de problemas NP. Esos otros problemas NP son los que llamamos NP-Completos. Decimos que un problema X reduce a X2 si X2 es más complejo de resolver que X, de forma que el algoritmo (o parte de él) usado para resolver X2 también resuelve X.Por lo tanto, y esa es la clave, un problema C es NP-Completo si cualquier problema NP se reduce a él, lo que significa que el algoritmo usado para resolver C puede resolver cualquier NP. Por lo tanto si encontramos un algoritmo para resolver C en un tiempo polinomial este mismo algoritmo (o parte de él) se podrá usar para resolver en tiempo polinomial cualquier NP.

se demostró que todos los problemas NP-Completos eran equivalentes entre sí. Eso significa que si un solo problema NP-Completo tiene una solución polinómica entonces por definición todos la tienen. Y viceversa: si se demuestra que tan solo un problema NP-Completo no tiene solución en tiempo polinómico, entonces ninguno la tiene.

La criptografía actual depende de un problema de la clase NP, el de la descomposición en factores de un número, para el que no tenemos algoritmos eficientes. El más eficiente de todos tardó 18 meses en descomponer en factores un número de 200 cifras decimales, que son los que se usan habitualmente en criptografía.

EJEMPLOS

- El problema del viajante. Dicho problema nos dice que debemos obtener el camino más corto que pase una vez por cada una de las N ciudades y que vuelva al punto de origen. Sabes la distancia entre todas las ciudades y puedes ir de cualquier ciudad a cualquier otra en cualquier orden.
- El problema de la suma de subconjuntos. Consiste en, dado un conjunto de enteros, determinar si existe algún subconjunto suyo tal que la suma de sus elementos sea cero. Este problema es claramente NP (fácil de verificar si una solución x es cierta o no (basta con comprobar que x es un subconjunto del conjunto original y que sus elementos suman 0)), pero difícil de solucionar. Pues se demostró que ese problema es también NP-Completo.