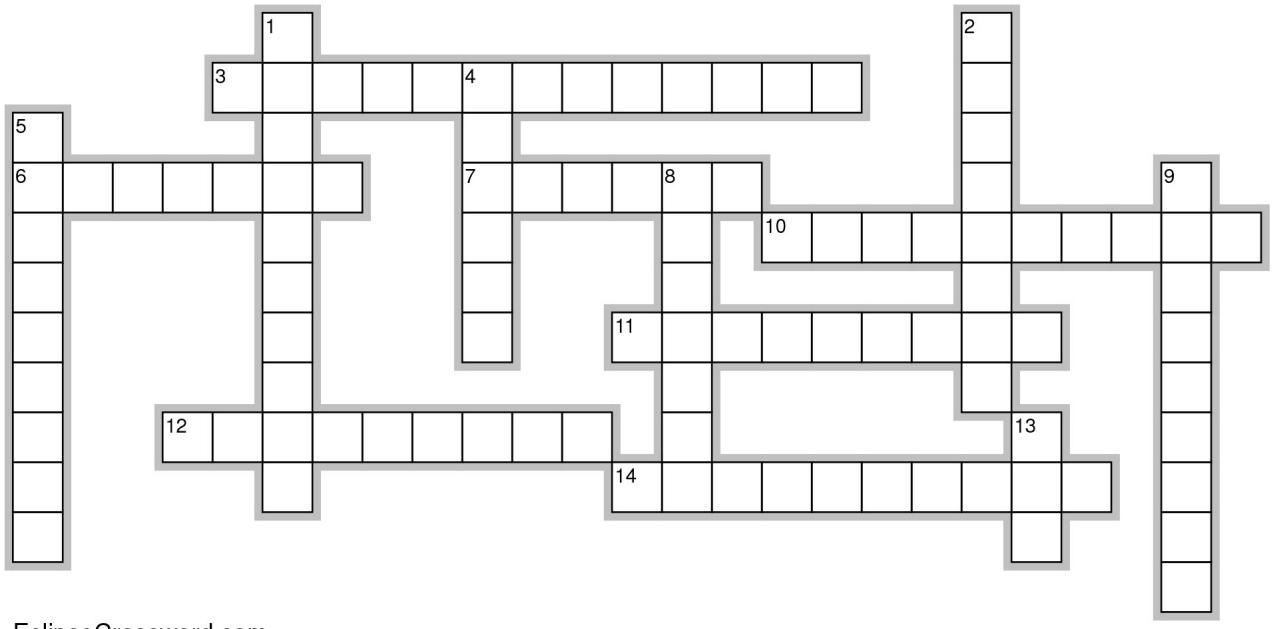
El arte de medir lo imposible

30/05/2024

EL ARTE DE MEDIR LO IMPOSIBLE

A. REQUENA & VALLE DE ELDA © 2024



EclipseCrossword.com

HORIZONTALES

3. La teoría de esta complejidad es una rama de la informática teórica y de las matemáticas que se ocupa de clasificar y estudiar los problemas en función de los recursos computacionales que se necesitan para resolverlos.
6. La clase PSPACE incluye los problemas que pueden ser resueltos usando una cantidad polinómica de éste (memoria), sin importar el tiempo que puedan requerir.
7. Una propiedad fundamental del juego de Hex es que no puede haberlo; siempre hay un ganador.
10. El tiempo polinómico es importante porque es una medida práctica de ésta.
11. Uno de los interrogantes más emblemáticos es el de la dificultad de éstos.
12. El problema del viajanet de comercio, consiste en encontrar el más corto que permita visitar cada ciudad exactamente una vez y regresar a la ciudad de origen.
14. Los algoritmos que se ejecutan en tiempo polinómico se consideran eficientes y escalables para tamaños de entrada de este tipo.

VERTICALES

1. La clase P (tiempo polinómico), abarca los problemas que pueden ser resueltos en un tiempo así, por una máquina determinista.
2. Una máquina de Turing no determinista puede hacerlo soluciones en cada paso de computación.
4. Un problema se considera que puede ser resuelto en tiempo polinómico no determinista si existe una máquina no determinista que puede resolver el problema siendo éste polinómico.
5. Cada clase representa un conjunto de problemas que pueden serlo con una cantidad específica de recursos.
8. El tiempo polinómico no determinista se refiere a la capacidad de este tipo de una máquina no determinista para resolver problemas en tiempo polinómico.
9. La clase de problemas NP-Completo, son problemas que lo son, al menos, tanto como cualquier otro problema en NP.
13. El juego así denominado, es un juego de tablero estratégico para dos jugadores que tiene mucho interés tanto en matemáticas como en teoría de la complejidad computacional.

Uno de los interrogantes más emblemáticos es el de la dificultad de los problemas. A cada uno nos parece que el nuestro es el de mayor envergadura o el más nimio, en función del estado de ánimo y del momento. Ciertamente, la cuestión es de mayor alcance, por cuanto se da en todos los ámbitos, sin excluir ninguna esfera de la existencia humana. El interrogante, en todo caso es: ¿por qué unos problemas son más difíciles que otros? Tiene especial significación en el área de conocimiento de ciencias de la computación en la que se ha generado un campo que se denomina teoría de la complejidad computacional.

La teoría de la complejidad computacional es una rama de la informática teórica y de las matemáticas que se ocupa de clasificar y estudiar los problemas en función de los recursos computacionales que se necesitan para resolverlos. Estos recursos incluyen el tiempo (número de pasos computacionales) y el espacio (cantidad de memoria) necesarios para ejecutar un algoritmo.

Los problemas se agrupan en clases de complejidad, como P, NP, NP-completo y PSPACE. Cada clase representa un conjunto de problemas que pueden ser resueltos con una cantidad específica de recursos. La clase P (tiempo polinómico) abarca los problemas que pueden ser resueltos en tiempo polinómico por una máquina determinista. El término "tiempo polinómico" se refiere a la cantidad de tiempo que emplea un algoritmo para resolver un problema, expresada como una función polinómica del tamaño de la entrada. En otras palabras, un algoritmo se dice que tiene una complejidad temporal polinómica si el tiempo de ejecución del algoritmo puede ser acotado por una función de la forma O(nk), donde n es el tamaño de la entrada y k es una constante positiva. Por aclarar los conceptos, un algoritmo se ejecuta en tiempo polinómico si existe un polinomio p(n) tal que para cualquier entrada de tamaño n, el tiempo de ejecución del algoritmo es O(p(n)). Esto significa que, para entradas de tamaño n, el tiempo de ejecución no crece más rápido que una función polinómica de n. Unos ejemplos aclaran el enunciado: a) búsqueda lineal, en una lista de n elementos tiene una complejidad temporal de O(n). Aquí, la función polinómica es p(n)= n. b) ordenamiento por inserción, que tiene una complejidad temporal de O(n2) en el peor caso. Aquí, la función polinómica es p(n)= n2. c) algoritmo de Dijkstra, para encontrar el camino más corto en un gráfico, tiene una complejidad temporal de O(n2) si se implementa con una matriz adyacente, que se utiliza para describir las conexiones entre los nodos (o vértices) de un grafo. Las filas y las columnas representan los nodos, y las entradas de la matriz indican si hay una arista (o enlace) entre los nodos correspondientes. d) Algoritmo de Euclides para encontrar el máximo común divisor (MCD) de dos números, que tiene una complejidad temporal de O(log n), donde n es el valor del número mayor de los dos. e) la multiplicación de matrices n×n tiene una complejidad temporal de O(n3) en su implementación más sencilla.

El tiempo polinómico es importante porque es una medida práctica de eficiencia. Los algoritmos que se ejecutan en tiempo polinómico se consideran eficientes y escalables para tamaños de entrada razonables. En contraste, los algoritmos que tienen tiempos de ejecución exponenciales (por ejemplo, O(2n), se vuelven impracticables para tamaños de entrada moderadamente grandes. La Clase NP (tiempo polinómico no determinista) incluye problemas cuyas soluciones se pueden verificar (no se pueden resolver, sino verificar) en tiempo polinómico por una máquina determinista. Un problema se considera que puede ser resuelto en tiempo polinómico no determinista si existe una máquina no determinista que puede resolver el problema en tiempo polinómico. El tiempo polinómico no determinista (Nondeterministic Polynomial time, NP) es un concepto de la teoría de la complejidad computacional que describe una clase de problemas para los cuales una solución puede ser verificada en tiempo polinómico por una máquina determinista, aunque encontrar dicha solución puede no ser tan simple. Es importante destacar que una máquina no determinista es un modelo teórico que no tiene equivalente directo en hardware real, pero sirve para conceptualizar la potencia computacional necesaria para ciertos tipos de problemas.

Una máquina de Turing no determinista puede "adivinar" soluciones en cada paso de computación. En lugar de seguir una única secuencia de pasos como hace una máquina determinista, puede seguir múltiples caminos simultáneamente. Como ejemplo consideremos el problema de la satisfactilidad booleana (SAT), que consiste en el problema de determinar si existe una asignación de valores a las variables que haga que una fórmula booleana sea verdadera. Un algoritmo no determinista considera dos etapas: a) adivinación (o prospección), en la que la máquina no determinista "adivina" una asignación de valores (verdadero o falso) para cada variable de la fórmula y b) verificación, en la que la máquina determinista verifica en tiempo polinómico si la asignación adivinada satisface la fórmula booleana. La cuestión radica en que si existe al menos una asignación que satisface la fórmula, la máquina no determinista puede encontrarla y verificarla en tiempo

polinómico. La noción de tiempo polinómico no determinista es crucial para entender el problema todavía abierto y más famoso en la teoría de la complejidad computacional, el problema P versus NP. Este problema pregunta si cada problema cuya solución puede ser verificada en tiempo polinómico por una máquina determinista (es decir, está en NP) también puede ser resuelto en tiempo polinómico por una máquina determinista (es decir, está en P).

Un ejemplo que aclara el concepto de tiempo polinómico no determinista es el problema del viajante de comercio, cuya descripción es: dado un conjunto de ciudades y las distancias entre cada par de ellas, encontrar el recorrido más corto que permita visitar cada ciudad exactamente una vez y regresar a la ciudad de origen. En la fase de adivinación, la máquina no determinista "adivina" un posible recorrido y en la fase de verificación, la máquina determinista verifica si el "recorrido adivinado" tiene una longitud menor o igual a un valor dado efectuarlo en tiempo polinómico.

En suma, el tiempo polinómico no determinista se refiere a la capacidad teórica de una máquina no determinista para resolver problemas en tiempo polinómico. Esto implica que la máquina puede explorar simultáneamente muchas soluciones posibles y verificar en tiempo polinómico si una de ellas es correcta. La clase NP incluye muchos problemas importantes en la computación, y la relación entre P y NP sigue siendo una cuestión fundamental en la teoría de la complejidad computacional.

La clase de problemas NP-Completo, son problemas que son, al menos, tan difíciles como cualquier otro problema en NP. Si se encontrara un algoritmo polinómico para uno de estos problemas, todos los problemas en NP podrían resolverse en tiempo polinómico. Estos problemas son extremadamente importantes porque tienen dos propiedades principales a) están en NP, lo que significa que sus soluciones pueden ser verificadas en tiempo polinómico por una máquina determinista y b) NP-duros, ya que cualquier problema en NP puede ser reducido en tiempo polinómico a un problema NP-completo. Esto implica que resolver eficientemente un problema NP-completo (en tiempo polinómico) permitiría resolver eficientemente todos los problemas en NP. Un ejemplo aclara el concepto: problema de la mochila, en el que dado un conjunto de elementos, cada uno con un peso y un valor, determinar si se puede seleccionar un subconjunto de elementos cuyo peso total no exceda una capacidad dada y cuyo valor total sea al menos un valor dado. Otro ejemplo es el problema del coloreado de Grafos, en el que dado un grafo y un número k, determinar si es posible colorear los nodos del grafo con k colores de tal manera que no haya dos nodos adyacentes con el mismo color.

Los problemas NP-completos son fundamentales en la teoría de la complejidad computacional, porque representan los problemas más difíciles dentro de la clase NP. Si se encuentra un algoritmo eficiente (polinómico) para resolver un problema NP-completo, significaría que todos los problemas en NP pueden ser resueltos eficientemente, resolviendo así la famosa pregunta de si P=NP. Los problemas NP-completos son útiles para entender la dificultad y la relación entre distintos problemas computacionales.

La clase PSPACE incluye los problemas que pueden ser resueltos usando una cantidad polinómica de espacio (memoria), sin importar el tiempo que puedan requerir. Es una clase importante para entender los límites de la computación en términos de memoria, en contraste con la clase NP, que se centra en el tiempo de cálculo. Este tipo de problemas son muy frecuentes, en especial en el contexto de la IA. Muchos problemas de juegos, que conllevan cómo determinar si existe una estrategia ganadora en juegos combinatorios de dos jugadores, están en PSPACE. Un ejemplo clásico es el problema de decidir si el jugador que inicia en un juego de Hex tiene una estrategia ganadora.

El juego de Hex es un juego de tablero estratégico para dos jugadores que tiene mucho interés tanto en matemáticas como en teoría de la complejidad computacional. Fue inventado por el matemático danés Piet Hein en 1942 y redescubierto de forma independiente por John Nash en 1948. Consiste en un tablero de hexágonos en forma de romboide con $n \times n$ celdas (generalmente 11x11, aunque puede ser de otros tamaños), dos jugadores, uno con fichas de un color (normalmente rojo) y otro con fichas de otro color (normalmente azul) y un objetivo, ya que cada jugador intenta conectar los lados opuestos de su color del tablero con una cadena ininterrumpida de sus fichas. El jugador rojo intenta conectar el lado izquierdo con el lado derecho, mientras que el jugador azul intenta conectar el lado superior con el lado inferior. Los jugadores se turnan para colocar una ficha de su color en cualquier celda vacía del tablero; una vez colocada, la ficha no se puede mover ni quitar; el primer jugador que conecta sus lados opuestos gana el juego.

Una propiedad fundamental del juego de Hex es que no puede haber un empate; siempre hay un ganador. Esto se puede demostrar utilizando la teoría de la topología y el teorema del punto fijo de Brouwer. El juego es PSPACE-completo, lo que significa que determinar si un jugador tiene una estrategia ganadora desde una posición

arbitraria es un problema que puede ser resuelto en espacio polinómico. Esto implica que es un problema difícil desde el punto de vista computacional. John Nash, inmortalizado en el film una mente prodigiosa, formuló el teorema de Nash, que demostró que el primer jugador (jugador que empieza) tiene una estrategia ganadora en un tablero de Hex de cualquier tamaño, aunque la estrategia explícita no es conocida y es extremadamente compleja. El análisis de juegos como Hex ayuda en la comprensión de problemas de verificación y optimización en sistemas computacionales y en el desarrollo de algoritmos para inteligencia artificial en juegos.

El juego de Hex no solo es un juego de estrategia interesante, sino que también tiene una profunda conexión con la teoría de la complejidad computacional. La demostración de que es PSPACE-completo destaca la dificultad de desarrollar estrategias ganadoras y la importancia del juego en el estudio de problemas computacionales complejos. La combinación de estrategia, matemática y teoría computacional hace que Hex sea un tema fascinante tanto para jugadores como para teóricos de la computación.



Imagen creada con ayuda de ChatGPT con DALL-E.

Uno de los aspectos relevantes en el tratamiento de la complejidad computacional son las técnicas para transformar un problema en otro, preservando la dificultad. Esto es fundamental para mostrar que un problema es NP-completo, al reducir un problema conocido a otro nuevo. Adquiere especial relevancia el concepto de complejidad Intrínseca, que conlleva el estudio de la dificultad inherente de los problemas, independientemente del algoritmo específico. El objetivo subyacente es el análisis y diseño de algoritmos que resuelvan problemas en tiempo polinómico y la estimación y medición del tiempo que conlleva ejecutar un algoritmo en función del tamaño de la entrada. Desde el punto de vista de la complejidad espacial conlleva el análisis de la cantidad de memoria que un algoritmo necesita y el estudio de cómo el tiempo y el espacio se relacionan y pueden intercambiarse en algunos casos.

Los problemas inherentemente difíciles son: a) problemas insolubles para los cuales no existe un algoritmo que siempre produzca una solución correcta, p.e., el problema de la parada, problema fundamental en la teoría de la computación, formulado por Alan Turing en 1936. Es uno de los primeros ejemplos de un problema que se ha demostrado que es indecidible, lo que significa que no existe un algoritmo general que pueda resolverlo para todas las posibles entradas. Se puede enunciar de la siguiente manera: dado un programa P y una entrada E, determinar si P se detendrá (es decir, terminará su ejecución) en algún momento cuando se ejecute con la entrada I. Alan Turing demostró que el problema de la parada es indecidible. Es decir, no existe una máquina de Turing que pueda resolver el problema de la parada para todas las posibles combinaciones de programas P y entradas E. La prueba de la indecidibilidad del problema de la parada es un ejemplo clásico de una prueba por contradicción y utiliza una técnica conocida como diagonalización. La indecidibilidad del problema de la parada tiene varias implicaciones profundas en la teoría de la computación, porque establece límites fundamentales sobre lo que es computacionalmente posible. No todos los problemas pueden ser resueltos por una máquina de Turing. Tiene impacto en la verificación de programas, ya que muestra que no es posible crear un programa general que pueda analizar otros programas y determinar de manera fiable si se detendrán y es una base para demostrar la indecidibilidad de muchos otros problemas en la computación mediante reducciones. La demostración de su indecidibilidad por Alan Turing es una piedra angular en la comprensión de lo que es y no es computacionalmente posible, y sigue siendo un tema central en la informática teórica. b) problemas Intratables, que no pueden ser resueltos en tiempo polinómico y para los cuales se cree que no existe un algoritmo eficiente (p.e., problemas NP-completos, si $P \neq NP$).

La teoría de la complejidad computacional tiene un impacto significativo en diversas áreas, incluyendo: a) criptografía, basada en la dificultad de ciertos problemas de complejidad como la factorización de números grandes; b) optimización, con la identificación de problemas que pueden ser resueltos de manera eficiente y aquellos que requieren heurísticas o aproximaciones; c) teoría de juegos y economía, con los análisis de problemas de decisión y estrategias óptimas y d) en inteligencia artificial, para evaluación de la viabilidad de

algoritmos para tareas complejas.

La teoría de la complejidad computacional proporciona un marco teórico para entender las limitaciones fundamentales de los algoritmos y la computación, orientando el diseño de algoritmos eficientes y el entendimiento de la dificultad inherente de los problemas.

Durante las últimas cinco décadas, la investigación de la teoría de la complejidad ha aportado conocimientos fundamentales en la teoría de la computación, así como aplicaciones prácticas en muchas áreas, desde la criptografía a la computación paralela e IA y en muchos campos. La idea es resolver problemas con algoritmos paso a paso. Son problemas computacionales, pero no tienen por qué limitarse al trabajo con los ordenadores. Un ejemplo, lo aclara: ¿quién no ha tenido alguna vez el problema de ordenar alfabéticamente los libros de su estantería, con objeto de encontrar fácilmente o que pretende? Es un ejemplo computacional y existen muchas formas de resolverlo, ya que podemos formular muchos algoritmos para ello. Uno, sencillo, inmediato, en el que casi todo el mundo cae es colocar los libros en los estantes, de forma aleatoria. Una vez colocados, los sacamos y volvemos a efectuar la operación. Somos capaces de esperar que en alguna de las veces en que repetimos la operación, lograremos el fin buscado. Es evidente que cualquier otra estrategia será más afortunada que ésta. Cualquier otro algoritmo será más rápido.

Es frecuente que en el ámbito de la computación se hable de problemas fáciles y difíciles, cuando en realidad se está refiriendo algoritmos rápidos y lentos. De las distintas formas que hay para resolver un problema, si alguna de ella, algún algoritmo es rápido, el problema se considera fácil. Si no se ha dado con un algoritmo fácil, el problema es difícil. Algunos problemas han sido tratados durante largo tiempo, sin lograr un algoritmo rápido para resolverlo, pero no se puede descartar que lo haya. ¿cómo serían estos problemas? Aquí radica la esencia del problema P frente al NP. Mientras que los problemas P son calificados como fáciles, los NP pueden no serlo, aunque tengan algoritmos rápidos para verificar si una solución propuesta es correcta o no. Los teóricos de la complejidad se debaten con la cuestión P frente a NP y no es fácil lograr la respuesta hasta el punto de que parece que este problema sobre la dificultad de los problemas computacionales es en sí mismo difícil de resolver. Para muchos problemas, no se dispone de algoritmos que ofrezcan mejor resultado que el de simplemente verificar todas las soluciones posibles, Afortunadamente, la capacidad de los ordenadores actuales lo permite, casi siempre. Cabe que ni siquiera se disponga de capacidad para enumerar todas las posibilidades.

Siempre queda mucho camino por recorrer. Si tenemos la impresión de que estamos o hemos llegado al final del camino, una vez más y son muchas las registradas en la historia, estamos lejos de ello. Cada vez que descorremos la cortina de la ignorancia, surgen otros interrogantes a resolver. Así es la vida y así esconde la Naturaleza sus secretos. Como la raza humana es osada, ahí estamos intentando medir lo imposible. Es la única forma de algún día hacerlo posible.

EL ARTE DE MEDIR LO IMPOSIBLE

A. REQUENA & VALLE DE ELDA © 2024

