TEORIA DE LA COMPLEJIDAD COMPUTACIONAL

La teoría de la complejidad computacional o teoría de la complejidad informática es una rama de la teoría de la computación que se centra en la clasificación de los problemas computacionales de acuerdo con su dificultad inherente, y en la relación entre dichas clases de complejidad.

Una de las metas de la teoría de la complejidad computacional es determinar los límites prácticos de qué es lo que se puede hacer en una computadora y qué no. Otros campos relacionados con la teoría de la complejidad computacional son el análisis de algoritmos y la teoría de la computabilidad. Una diferencia significativa entre el análisis de algoritmos y la teoría de la complejidad computacional, es que el primero se dedica a determinar la cantidad de recursos requeridos por un algoritmo en particular para resolver un problema, mientras que la segunda, analiza todos los posibles algoritmos que pudieran ser usados para resolver el mismo problema.

En resumen, la teoría de la complejidad estudia cómo crece el coste computacional principalmente en memoria y tiempo de resolver un determinado problema en relación a lo que crece el tamaño de dicho problema, tiene su importancia pues de esta manera se puede decidir si resolver un problema merece la pena o no hacerlo.

Problema computacional

Un problema computacional constituye una pregunta a ser respondida, teniendo generalmente varios parámetros, o variables libres, cuyos valores no se han especificado.

Un problema se describe mediante:

1. Una descripción general de todos sus parámetros (pueden ser de entrada o de salida).
2. Una sentencia que describa las propiedades que la respuesta, o la solución, debe cumplir.

Problemas de decisión

Un problema de decisión es un tipo especial de problema computacional cuya respuesta es solamente "sí" o "no" (o, de manera más formal, "1" o "0").

Un problema de decisión pudiera verse como un lenguaje formal, donde los elementos que pertenecen al lenguaje son las instancias del problema cuya respuesta es "sí", los que no pertenecen al lenguaje son aquellas instancias cuya respuesta es "no". El objetivo es decidir, con la ayuda de un algoritmo, si una determinada entrada es un elemento del lenguaje formal considerado. Si el algoritmo devuelve como respuesta "sí", se dice que el algoritmo *acepta* la entrada, de lo contrario se dice que la *rechaza*.

Algoritmos

Podemos decir informalmente, que los algoritmos son procedimientos paso-a-paso para resolver problemas. Se puede pensar en ellos como simples programas de computadora, escritos en un lenguaje artificial específico

Se dice que un algoritmo resuelve un problema A, si dicho algoritmo se puede aplicar a cualquier instancia I de A, y se garantiza que siempre produce una solución para dicha instancia. De manera general, nos interesa encontrar el algoritmo más "eficiente" para resolver cierto problema.

La clase NP

La clase de complejidad NP consta de los problemas "verificables" en tiempo polinómico. Por verificable se entiende a un problema tal que dado un certificado de solución (candidato a solución), se puede verificar que dicho certificado es correcto en un tiempo polinómico en el tamaño de la entrada. A los problemas en la clase NP usualmente se les llama *problemas NP*. De forma más especifica El término NP proviene de *no determinista en tiempo polinómico.*

La clase de complejidad NP contiene problemas que no pueden resolverse en un tiempo polinómico. Cuando se dice que un algoritmo no puede obtener una solución a un problema en tiempo polinómico siempre se intenta buscar otro procedimiento que lo consiga mejorar.

La importancia de esta clase de problemas de decisión es que contiene muchos problemas de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas.

La clase P

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. P contiene a la mayoría de problemas naturales, algoritmos de programación lineal, funciones simples, Por ejemplo, las sumas de dos números naturales se resuelven en tiempo polinómico (para ser más exactos es de orden 2n). Entre los problemas que se pueden resolver en tiempo polinómico nos encontramos con diversas variedades como los logarítmicos (log(n)), los lineales (n), los cuadráticos (n2), los cúbicos (n3), Volviendo al ejemplo principal llegamos a la conclusión que la función de elevar al cuadrado está contenida en la clase P.

La clase P juega un papel importante en la teoría de la complejidad computacional debido a que:

1. P es invariante para todos los modelos de cómputo que son polinómicamente equivalentes a la Máquina de Turing determinista.
2. A grandes rasgos, P corresponde a la clase de problemas que, de manera realista, son solubles en una computadora.

Ejemplos NP

Problema de partición

En ciencias de la computación, el Problema de la partición es un problema NP-completo, que visto como un problema de decisión, consiste en decidir si, dado un multiconjunto de números enteros, puede éste ser particionado en dos "mitades" tal que sumando los elementos de cada una, ambas den como resultado la misma suma.

Más precisamente, dado un multiconjunto S de enteros: ¿existe alguna forma de particionar S en dos subconjuntos S_1 y S_2, tal que la suma de los elementos en S_1 sea igual que la suma de los elementos en S_2?

Problema de Clique

Dado un grafo G=(N,A), decimos que G tiene un clique de tamaño k si existe un subgrafo G' = (N',A') de G tal que N' es subconjunto de N, |N'|=k y A'=N'×N', vale decir, todos sus vértices están conectados entre ellos. En el grafo de la derecha, los vértices 1, 2 y 5 forman un clique porque cada uno tiene un arco que le une a los otros. En cambio, los vértices 2, 3 y 4 no, dado que 2 y 4 no son adyacentes.

El problema de clique es un problema de decisión para determinar cuándo un grafo contiene un clique de al menos un tamaño k. Una vez que tenemos k o más vértices que forman un clique, es trivial verificar que lo son, por eso es un problema NP. El correspondiente problema de optimización, consiste en encontrar un clique de tamaño máximo en un grafo (un subgrafo completo de tamaño máximo). Este problema se puede enunciar como un problema de decisión si la pregunta que se hace es saber si existe un clique de tamaño k en el grafo.

Problema de la Mochila

En algoritmia, el problema de la mochila, comúnmente abreviado por KP (del inglés *Knapsack problem*) es un problema de optimización combinatoria, es decir, que busca la mejor solución entre un conjunto finito de posibles soluciones a un problema. Modela una situación análoga al llenar una mochila, incapaz de soportar más de un peso determinado, con todo o parte de un conjunto de objetos, cada uno con un peso y valor específicos. Los objetos colocados en la mochila deben maximizar el valor total sin exceder el peso máximo.

Este problema se ha resuelto tradicionalmente mediante programación lineal entera.

El hecho de que se trate de programación lineal hace referencia a que la función a optimizar y las inecuaciones que constituyen las restricciones han de ser lineales, es decir, han de ser funciones cuyas incógnitas estén elevadas exclusivamente a la unidad.

Existe otra forma de resolver este tipo de problema, a través de los denominados algoritmos voraces. Una aproximación voraz consiste en que cada elemento a considerar se evalúa una única vez, siendo descartado o seleccionado, de tal forma que si es seleccionado forma parte de la solución, y si es

descartado, no forma parte de la solución ni volverá a ser considerado para la misma. Con este método no siempre es posible dar una solución a un problema.

BIBLIOGRAFIA:

colaboradores de Wikipedia. (2019, 18 septiembre). ***Problema del ciclo hamiltoniano*****. Wikipedia, la enciclopedia libre.** **https://es.wikipedia.org/wiki/Problema_del_ciclo_hamiltoniano**

Marí, R. P. (2017, 22 mayo). ***El problema que los informáticos no han podido resolver en 45 años*****. EL PAÍS.** **https://elpais.com/tecnologia/2017/05/19/actualidad/1495202801_698394.html**

colaboradores de Wikipedia. (2019b, septiembre 20). ***Lista de 21 problemas NP-completos de Karp*****. Wikipedia, la enciclopedia libre.** **https://es.wikipedia.org/wiki/Lista_de_21_problemas_NP-completos_de_Karp**

colaboradores de Wikipedia. (2019c, octubre 11). ***Coloración de grafos*****. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Coloraci%C3%B3n_de_grafos**

colaboradores de Wikipedia. (2019b, septiembre 19). ***Problema de la mochila*****. Wikipedia, la enciclopedia libre.** **https://es.wikipedia.org/wiki/Problema_de_la_mochila**

colaboradores de Wikipedia. (2020, 30 mayo). ***Problema de la clique*****. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Problema_de_la_clique**

MFMA. (2014, 8 abril). ***PROBLEMAS CLASES P, NP Y NP-COMPLETOS*****. prezi.com.** **https://prezi.com/vjxp4_afeppd/problemas-clases-p-np-y-np-completos/**

colaboradores de Wikipedia. (2019a, septiembre 18). ***P (clase de complejidad)*****. Wikipedia, la enciclopedia libre.** **https://es.wikipedia.org/wiki/P_(clase_de_complejidad)**

colaboradores de Wikipedia. (2020a, abril 4). ***NP (clase de complejidad)*****. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/NP_(clase_de_complejidad)**

colaboradores de Wikipedia. (2020b, mayo 16). ***Teoría de la complejidad computacional*****. Wikipedia, la enciclopedia libre.** **https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_complejidad_computacional**