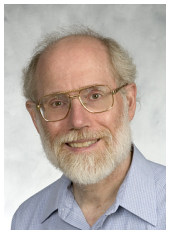
Complejidad - Problemas NP-Completos

Algoritmos y Estructuras de Datos III

Teoría de Complejidad

Michael Garey



David Johnson

- M. Garey y D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

Teoría de Complejidad

- Un algoritmo eficiente es un algoritmo de complejidad polinomial.
- Un problema está bien resuelto si se conocen algoritmos eficientes para resolverlo.
- El objetivo es clasificar los problemas según su complejidad.
- Un problema de decisión es un problema cuya respuesta es "sí" o "no".
- La clasificación y el estudio de teoría de complejidad se hace para problemas de decisión.

Distintas versiones de un problema de optimización Π

Dada una instancia I del problema Π:

- Versión de evaluación: Determinar el **valor** de una solución óptima de Π para I.
- Versión de optimización: Encontrar una **solución óptima** del problema Π para I (de valor mínimo o máximo).
- Versión de decisión: Dado un número k, ¿existe una solución factible de Π para I tal que $c(S) \leq k$ si el problema es de minimización (o $c(S) \geq k$ si el problema es de maximización)?
- Versión de localización: Dado un número k, determinar una **solución factible** de Π para I tal que $c(S) \leq k$.

Ejemplo: Problema del viajante de comercio

Dado un grafo G con longitudes asignadas a sus aristas:

- Versión de evaluación: Determinar el valor de una solución óptima, o sea la longitud de un circuito hamiltoniano de G de longitud mínima.
- Versión de optimización: Determinar un circuito hamiltoniano de G de longitud mínima.
- Versión de decisión: Dado un número k, ¿existe un circuito hamiltoniano de G de longitud menor o igual a k?
- Versión de localización: Dado un número k, determinar un circuito hamiltoniano de G de longitud menor o igual a k.

Problemas intratables

Definición: Un problema es intratable si no puede ser resuelto por algún algoritmo eficiente.

Un problema puede ser intratable por distintos motivos:

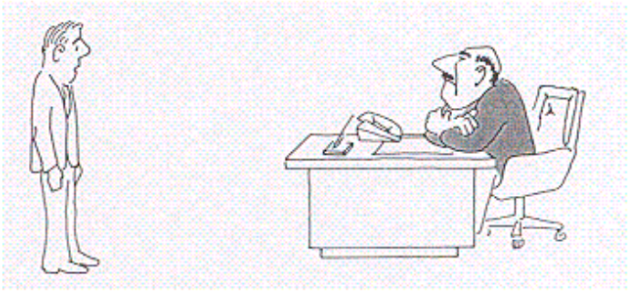
- El problema requiere una repuesta de longitud exponencial (ejemplo: pedir todos los circuitos hamiltonianos de longitud a lo sumo k).
- El problema es indecidible (ejemplo: problema de la detención).
- El problema es decidible pero no se conocen algoritmos polinomiales que lo resuelvan.

Problemas intratables

One day your boss calles you into his office and confides that the company is about to enter the highly competitive "bandersnatch" market. A good method is needed for determining whether or not any given set of specifications for a new bandersnatch component can be met and, if so, for constructing a design that meets them. Since you are the company's chief algorithm designer, your charge is to find an efficient algorithm for doing this.

After consulting with the bandersnatch department to determine exactly what the problem is, you eagerly hurry back to your office, and plunge into the task. Some weeks later, you have not been able to come up with any algorithm substantially better than searching through all possible designs. This would involve years of computation time for just one set of specifications. You certainly don't want to return to your boss's office and report:

Problemas intratables



"I can't find an efficient algorithm,
I guess I'm just too dumb."

Problemas intratables

To avoid serious damage to your position within the company, it would be much better if you could prove that the bandersnatch problem is inherently intractable, that no algorithm could possibly solve it quickly. Then you could stride confidently into the boss's office and proclaim:

Problemas intratables

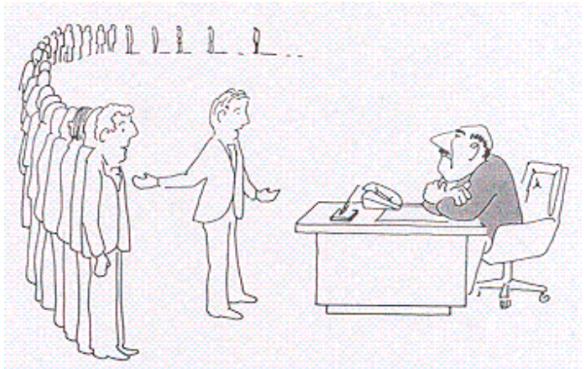
"I can't find an efficient algorithm,
because no such algorithm is possible! "

Problemas intratables

Unfortunately, proving inherent intractibility can be just as hard as finding efficient algorithms.

The theory of NP-completeness provides many straightforward techniques for proving that a given problem is "just as hard" as a large number of other problems that are widely recognize as being difficult and that have been confounding the experts for years. Armed with these techniques, you might be able to prove that the bandersnatch problem is NP-complete and march into your boss's office and announce:

Problemas NP-completos



"I can't find an efficient algorithm,
but neither can all these famous people."

Problemas NP-completos

At the very least, this would inform your boss that it would do no good to fire you and hire another expert on algorithms.

Now you can spend you time looking for efficient algorithms that solve various special cases of the general problem. You might look for algorithms that, though not guaranteed to run quickly, seem likely to do so most of the time. Or you might even relax the problem somewhat, looking for a fast algorithm that merely finds designs that meet most of the component specifications. Thus, the primary application of the theory of NP-completeness is to assist algorithm designers in directing their problem-solving efforts toward those approaches that have the greatest likelihood of leading to useful algorithms.

Las clases P y NP

Definiciones:

- Un problema de decisión pertenece a la clase **P (polinomial)** si existe un algoritmo polinomial para resolverlo.
- Un problema de decisión pertenece a la clase **NP (polinomial no-determinísticamente)** si dada una instancia con respuesta "sí" se puede dar un certificado que garantiza que la respuesta es "sí", y esta garantía puede ser verificada en tiempo polinomial.

Relaciones entre las clases:

- $P \subseteq NP$
- **Problema abierto:** ¿Es $P = NP$?

Ejemplos de problemas en NP

- Suma de enteros.
- Multiplicación de enteros.
- Árbol generador mínimo.
- Clique máxima.
- Camino mínimo entre un par de nodos.
- Problema del viajante de comercio.
- Conjunto independiente de cardinal máximo.
- Problema de satisfactibilidad (SAT): Dado un conjunto de claúsulas $C_1, \ldots, C_m$ formadas por literales basados en las variables booleanas $X = \{x_1, \ldots, x_n\}$, determinar si hay una asignación de valores de verdad a las variables de X tal que la expresión $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ sea verdadera.

Máquinas de Turing no-determinísticas (MTND)

- Una MTND tiene los mismos componentes que vimos para una MTD, con la siguiente excepción.
- Un programa correspondiente en una MTND es una tabla que mapea un par (q_i, t_i) a un conjunto de ternas $(q_f, t_f, \{0, +1, -1\})$.
- Esto admite dos interpretaciones equivalentes:
  1. En cada paso se selecciona una de las alternativas posibles.
  2. En cada paso se continúa la ejecución en paralelo de las distintas alternativas, generando una copia de la MTND por cada alternativa.

Máquinas de Turing no-determinísticas (MTND)

- Una MTND resuelve un problema de decisión si ...
  1. existe una secuencia de alternativas seguidas que lleva a un estado de aceptación si y sólo si la respuesta es "sí", o bien
  2. alguna de las copias se detiene en un estado de aceptación si y sólo si la respuesta es "sí".
- La complejidad temporal de una MTND se define como el máximo número de pasos que toma reconocer una entrada aceptable en función de su tamaño.

Clase NP - Otra caracterización

- Un problema de decisión está en la clase NP si las instancias "sí" son reconocidas por una MTND polinomial.
- La clase NP se puede definir como el conjunto de problemas de decisión que se pueden resolver por un algoritmo polinomial no-determinístico.
- **Lema:** Si Π es un problema de decisión que pertence a la clase NP, entonces Π puede ser resuelto por un algoritmo determinístico en tiempo exponencial respecto del tamaño de la entrada.

Ejemplo: Conjunto independiente máximo

Dado un grafo $G = (V, X)$ y un entero k, ¿tiene G un conjunto independiente de tamaño mayor o igual a k?

guess(S): función multivaluada que retorna un nuevo elemento de S.

$I := \emptyset$
mientras $S \neq \emptyset$ **hacer**
  $v :=$ *guess*(S)
  $S := S \setminus \{v\}$
  si $\Gamma(v) \cap I = \emptyset$ **entonces** $I := I \cup \{v\}$
  si $|I| \geq k$ **entonces retornar** ''sí''
fin mientras
retornar ''no''

Transformaciones polinomiales

Deficiones:

- Una transformación o reducción polinomial de un problema de decisión Π_1 a uno Π_2 es una función polinomial que transforma una instancia I_1 de Π_1 en una instancia I_2 de Π_2 tal que I_1 tiene respuesta "sí" para Π_1 si y sólo si I_2 tiene respuesta "sí" para Π_2.
- El problema de decisión Π_1 se reduce polinomialmente a otro problema de decisión Π_2, $\Pi_1 \leq_p \Pi_2$, si existe una transformación polinomial de Π_1 a Π_2.

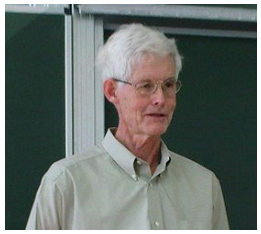
Las reducciones polinomiales son transitivas, es decir, si $\Pi_1 \leq_p \Pi_2$ y $\Pi_2 \leq_p \Pi_3$, entonces $\Pi_1 \leq_p \Pi_3$.

Problemas NP-completos

Definición: Un problema de decisión Π es NP-completo si:

1. $\Pi \in NP$
2. $\forall \bar{\Pi} \in NP$, $\bar{\Pi} \leq_p \Pi$

Problemas NP-completos



Stephen Cook

Leonid Levin

- **Teorema (Cook, 1971 – Levin, 1973).**
  SAT es NP-completo.