Complejidad Computacional

MLG521

Cristobal Rojas
Pamela Alvarez

Departamento de Ciencias de de la Ingeniería
Departamento de Ingeniería Matemática
Universidad Andrés Bello

MLG521

Motivación

- ¿ Cómo medir la dificultad de un problema ?
- ¿ En qué sentido podemos decir que un problema es más difícil que otro ?
- Si sabemos que un problema tiene solución, ¿ podemos siempre calcularla ?
- Si sabemos como calcular una solución, ¿ podemos siempre calcularla ?

Ejemplo

Problema 1

Sean $\{a_1, a_2, \ldots, a_n\}$ n números naturales. Ordenarlos de manera creciente

Problema 2

Sean $\{a_1, a_2, \ldots, a_n\}$, n ciudades . Encontrar un tour que las recorra todas a costo mínimo.

Ejemplo

Problema 1

Sean $\{a_1, a_2, \ldots, a_n\}$ n números naturales. Ordenarlos de manera creciente

Problema 2

Sean $\{a_1, a_2, \ldots, a_n\}$, n ciudades . Encontrar un tour que las recorra todas a costo mínimo.

Problemas Decidibles

Problema

Un problema esta formado por una instancia o input y una pregunta sobre la instancia(Podemos pensar con respuesta sí o no).

Decidible

Un problema es decidible si existe un algoritmo que, dada cualquier instancia, responde la pregunta en tiempo finito.

Existen problemas que no son decidibles (Ej: saber si un programa se va a "quedar pegado").

Problemas Decidibles

Problema

Un problema esta formado por una instancia o input y una pregunta sobre la instancia(Podemos pensar con respuesta sí o no).

Decidible

Un problema es decidible si existe un algoritmo que, dada cualquier instancia, responde la pregunta en tiempo finito.

Existen problemas que no son decidibles (Ej: saber si un programa se va a “quedar pegado”).

Notación Asintótica

La *calidad* de un algoritmo (para resolver un cierto problema) se refleja en su **tiempo de ejecución**, que mide el tiempo máximo que tarda el algoritmo en resolver una instancia, como función del *tamaño* de la instancia. Para expresar esto se usa la **notación asintótica**:

Big O

Sean $T(n)$ el tiempo de ejecución de un cierto algoritmo. Decimos que T es de orden $g(n)$, denotado por $T(n) = O(g(n))$ si existe una constante C tal que

$$T(n) \leq C \cdot g(n) \quad \forall n$$

Ex: si $T(n) = 5n^3 + 4n^2 + 4n + 10$, entonces $T(n) = O(n^3)$.

Análisis de Algortimos

Para un problema existen varios algoritmos, con distintos tiempos de ejecución. El tiempo de ejecución de un algoritmo dado entrega una **cota superior a la complejidad computacional** del problema.

- Ordenar una lista tiene complejidad $O(n^2)$ (porque sabemos como hacerlo en ese tiempo – podría tomar menos con otro algoritmo).
- El vendedor viajero tiene complejidad $O(n!)$ (porque usando fuerza bruta toma ese tiempo – podría tomar menos)

Comparación

Comparemos el tiempo que tomaría ejecutar algoritmos con distintos tiempos de ejecución, en instancias de distintos tamaños (en un computador de 2.5 GH)

$T(n)\backslash$ tam.	30	40	50	60
$O(n)$	0.00003 seg	0.00004 seg	0.00005 seg	0.00006 seg
$O(n^2)$	0.0009 seg.	0.0016 seg.	0.0025 seg.	0.0036 seg.
$O(n^5)$	24.3 seg	1.7 min.	5.2 min	13 min.
$O(2^n)$	17.9 min	12.7 dias	35.7 años	366 siglos
$O(3^n)$	6.5 años	3855 siglos	2×10^8 siglos	1.3×10^13 siglos

Las clases P y NP

Definición

Un problema pertenece a la clase P si su complejidad computacional es $O(n^k)$.

Ej: Decidir si existe un tour euleriano (visita todas las aristas una única vez)

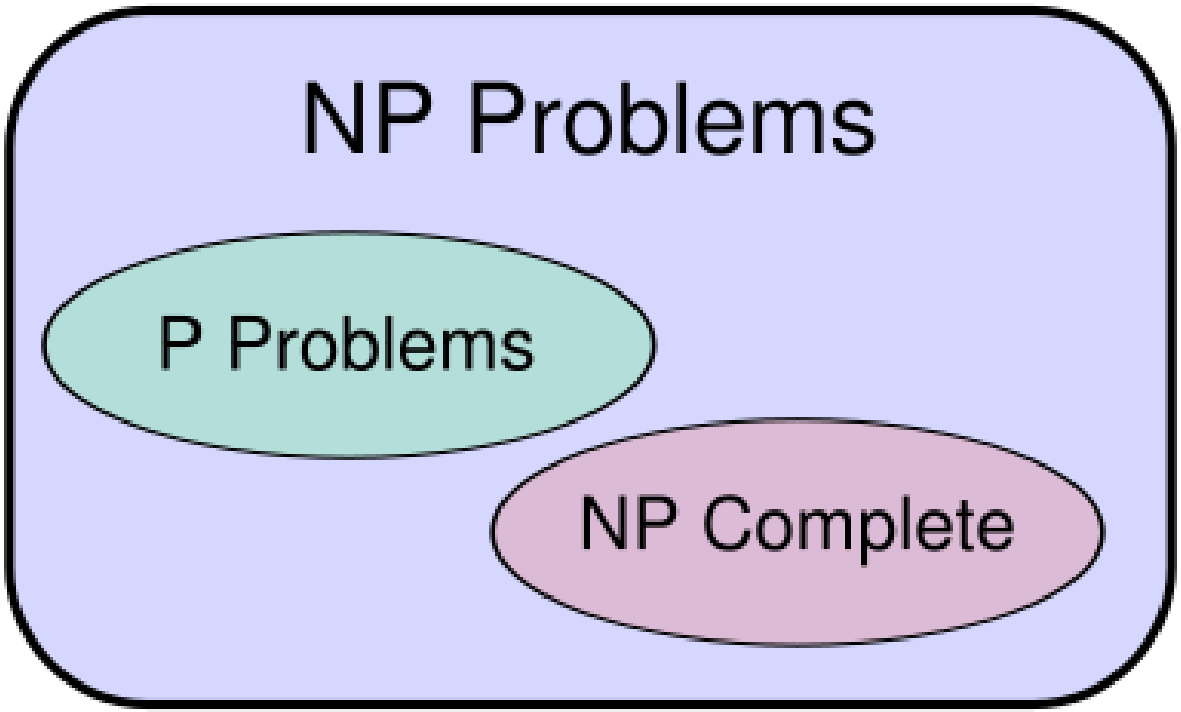
Definición

Un problema pertenece a la clase NP si existe un algoritmo que tarda a lo mas tiempo $O(n^k)$ *en verificar* si una respuesta es correcta o no.

Ej:

- Decidir si existe un tour euleriano
- Decidir si existe un tour hamiltoniano (visita todos los nodos una única vez)

P v/s NP



Reducciones polinomiales

Reducciones

Un problema A es reducible a un problema B si existe un algoritmo polinomial que transforma una instancia de A en una de B y si x es una solución para A si y sólo si su transfromación es una solución para B.

Esto quiere decir que el problema B es al menos tán dificil como el problema A.

NP-completo

NP-completitud

Un problema es NP-completo si todo problema en NP se puede reducir a él.

Ej: Vendedor Viajero, tour Hamiltoneano (son teoremas difíciles)

¿Qué hacemos?

- Existe una amplia literatura sobre este tipo de problemas.
- Buscamos soluciones para nuestro problema particular.
- Algoritmos de aproximación.
- Heurísticas.

Algoritmos de Aproximación

Definición

Para un problema de optimización A un algoritmo de α-aprox. es un algoritmo que encuentra una solución x' para A en tiempo polinomial tal que:

$$x' \leq \alpha x \ (Min)$$

$$x' \geq \alpha x \ (Max)$$

Heurísticas

En computación, dos objetivos fundamentales son encontrar algoritmos con buenos tiempos de ejecución y buenas soluciones, usualmente las óptimas. Una heurística es un algoritmo que abandona uno o ambos objetivos; por ejemplo, normalmente encuentran buenas soluciones, aunque en ocasiones no hay pruebas de que la solución no pueda ser arbitrariamente errónea; o se ejecuta razonablemente rápido, aunque no existe tampoco prueba de que deba ser así.