

INF155: Informática Teórica

Ayudantía 9 “Problemas Intratables”

Aldo Berrios Valenzuela

Miércoles 22 de Junio de 2016

Los lenguajes recursivamente enumerables nos dicen cuando aceptan palabras σ , pero no nos garantiza si rechaza ciertas palabras (por no responde “No”)

9. Problemas Intratables

- Que un problema sea decidable no implica que sea “fácil de resolver”.
- Problemas “intratables”: muy costosos en función del “input” (demoran más)
- Máquinas de Turing determinista (DTM) y no deterministas (NTM):
 - Reconocen los mismos lenguajes.
 - Costo natural de leer NTM es una cota inferior del costo
 - Simular una NTM por medio de una DTM requiere explorar todos los casos \Rightarrow exponencial.
- Clase P: los problemas que una DTM puede resolver en tiempo polinomial.
- Clase NP: Los problemas que una NTM puede resolver en tiempo polinomial. Otra definición: verificación de solución en tiempo polinomial por DTM.
- $P \subseteq NP$ ($P \subset NP$ o $P = NP$?)
- Habían problemas que se pensaban que estaban en NP, pero en realidad estaban en P. Por ejemplo, la programación lineal se pensó que estaban en NP.
- Si $P = NP$, problemas que pensamos que eran muy difíciles de resolver, en realidad eran “más fáciles de resolver”.
- Recursivos ($P \subseteq NP \subseteq$ Recursivo)
- Reducción polinomial:

$$P_1 \rightsquigarrow_p P_2 : x \text{ aceptado por } P_1 \Leftrightarrow f(x) \text{ aceptado por } P_2$$
$$f(x)$$

- Transformación polinomial y resolver P_2 en polinomial: entonces P_1 también se puede resolver en tiempo polinomial (las funciones son cerradas respecto a la composición de funciones).
- P_2 tiene solución $\Rightarrow P_1$ tiene solución.
- P_1 es indecidible $\Rightarrow P_2$ es indecidible.
- P_2 al menos tan difícil como P_1 .

9.1. P vs NP

- NP-duro (NP-hard): un problema P está en NP-duro si todo problema en NP es reducible polinomialmente a P .

$$x \rightsquigarrow P, \forall x \in \text{NP}$$

- NP-completo: un problema P está en NP-completo si:
 1. Es NP-hard.
 2. Está en NP.
- Para demostrar que un problema está en P, sólo tenemos que construir un algoritmo que resuelva el problema en tiempo polinomial.
- Para demostrar que un problema está en NP, tenemos que ver que una instancia es evaluable polinomialmente por una NTM y luego, verificar solución polinomial TM.
- Para demostrar que un problema es NP-completo:
 - Demostrar que está en NP.
 - Demostrar que está en NP-hard.
- Otra forma de demostrar que un problema está en NP-completo es demostrar es tomar este problema y reducirlo a otro problema NP-completo (todos los problemas NP-completo son reducibles entre sí).

9.2. Ejercicios

Ejemplo 9.1. Encontrar un número mayor de un arreglo.

Solución. Seguimos el siguiente algoritmo:

1. Tomar un elemento $i \in \{1, \dots, n\}$
2. Lo comparo con n elementos del arreglo.
3. Si es mayor a todos, entonces tenemos una solución.

La complejidad de este algoritmo está dado por el polinomio $n \cdot n = n^2$, por lo tanto este problema está en P. \square

Ejemplo 9.2 (SAT). Una expresión booleana es satisfacible si la expresión es true (con alguna asignación). Por ejemplo:

$$(x_1 \vee \bar{x}_2 \vee \dots \vee \bar{x}_n) \wedge (\dots) \wedge (\dots)$$

$$(x_1 \vee) \wedge (x_1 \vee \bar{x}_2)$$

Solución. Para dar solución a este problema, tenemos que probar todas las combinaciones posibles hasta encontrar aquellos valores que hagan que la expresión booleana sea true.

Evaluar este problema está en P, pero resolverlo está en NP, dado que la solución de este algoritmo tiene una complejidad de 2^n , que no es polinomio (y obviamente está en NP). \square

Ejemplo 9.3. Problema de la mochila¹. Este problema consiste en que un ladrón entra a robar a una casa y empieza a extraer especies de valor. Cada objeto tiene un peso a_i y puede venderse en el mercado negro a un valor c_i . La mochila del ladrón no soporta más de A kilogramos, y por lo tanto, no puede llevarse todos los objetos del hogar.

¹Se ve en IO1 (ILI292) y Optimización (INF292)

Sea z la ganancia obtenida al vender los objetos sustraídos en el mercado negro, entonces, lo que se busca es maximizar z . Luego, la función objetivo del caso es:

$$\text{máx } z = \sum_{i=1}^n c_i \cdot x_i$$

Sujeta a la restricción:

$$\sum_{i=1}^n a_i x_i \leq A$$

Donde:

- A : capacidad máxima (en kilogramos) de la mochila.
- a_i : peso del objeto i
- c_i : ganancia del objeto i .
- x_i : si tomo objeto el objeto i . $x_i \in \{0, 1\}$.

Determine si el problema de la mochila está en P o en NP.

Solución. Para dar solución a este problema, tenemos que probar todas las combinaciones posibles. A cada variable x_i podemos asignarle sólo 2 valores posibles, y como tenemos n variables, resolver este problema tiene una complejidad de 2^n . Este valor no es polinomial, por lo tanto, resolver este problema está en NP.

Ojo: existe un algoritmo para resolver el problema de la mochila, pero este algoritmo **no tiene solución en tiempo polinomial** y por eso está en NP. □

10. Observaciones de la Ayudantía

- Un problema itratable es aquel que se puede resolver, pero en temas de tiempo es tan costoso que es mejor no crear un algoritmo para resolverlo.
- Los problemas NP-completos son reducibles entre todos ellos (por transitividad).