NP-completitud y casos tratables

Semana $(7)_2 = 111$

Lógica para Ciencia de la Computación - IIC2213

Prof. Sebastián Bugedo

Programa

Obertura

Acto único
- NP-completitud
- Más problemas NP-completos
- Casos tratables

Epílogo

Programa

Obertura

Acto único
- NP-completitud
- Más problemas NP-completos
- Casos tratables

Epílogo

Clases de complejidad no deterministas

Definición (clases de tiempo)

Dado un alfabeto A, se define

$$\begin{aligned} \mathrm{NTIME}(g) \quad = \quad \{L \subseteq A^* \quad & | \; L \text{ puede ser aceptado en tiempo } g \\ & \text{por una máquina no determinista}\} \end{aligned}$$

Una clase fundamental

$$\mathrm{NP} \quad = \quad \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$$

NP es la clase de lenguajes aceptados por NTM polinomiales

La definición de NP

Teorema (def. alternativa de NP)

Sea L un lenguaje. $L \in \text{NP}$ si, y solo si, existe una máquina determinista $\mathcal{M}$ que funciona en tiempo polinomial y un polinomio $p(\cdot)$ tales que

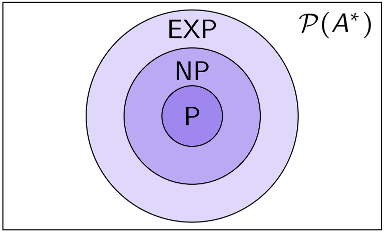
$$L = \{x \mid \text{existe } y \text{ tal que } |y| = p(|x|) \text{ y } \mathcal{M} \text{ acepta } (x, y)\}$$

El teorema nos da una nueva forma de demostrar que $L \in \text{NP}$:
buscar testigo y verificador ambos polinomiales

¿Cómo se relaciona NP con P y EXP?

Teorema

Las clases P, NP y EXP cumplen que $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{EXP}$



Al menos una de las inclusiones es estricta

Problemas NP-completos

Teorema (Cook-Levin)

El problema SAT es NP-completo

El teorema de Cook-Levin es clave por varias razones

- Nos entrega un primer problema NP-completo
- Cualquier $L \in \mathrm{NP}$ tal que $\mathrm{SAT} \leq_p L$ es también NP-completo
- Si probamos que $\mathrm{SAT} \in \mathrm{P}$, por teorema de reducciones tenemos que $\mathrm{P} = \mathrm{NP}$

SAT es clave para relacionar problemas con solución eficiente y solución verificable de forma eficiente

Hacia dónde vamos

Estudiaremos en detalle otros problemas NP-completos y sus potencialidades

¿Hay versiones de problemas NP-completos que sí se puedan resolver eficientemente?

¿Todo problema en NP es NP-completo?

¿Qué implicancias tendría si P=NP?
¿Se acaba el mundo?

Hoy: problemas NP-completos y consecuencias de P=NP

Playlist Unidad II y Orquesta

Playlist: LogiWawos #2

Además sigan en instagram:
@orquesta_tamen

Objetivos de la clase

- Demostrar que SAT es NP-completo
- Conocer casos de SAT NP-completos y polinomiales
- Comprender la relación entre las clases P y NP
- Conocer casos tratables de problemas NP-completos

Programa

Obertura

Acto único

- NP-completitud
- Más problemas NP-completos
- Casos tratables

Epílogo

Problemas NP-completos

Teorema (Cook-Levin)

El problema SAT es NP-completo

Demostración

Ya demostramos que $\mathrm{SAT} \in \mathrm{NP}$, por lo que debemos probar que SAT es NP-hard. Para esto, sea $L \in \mathrm{NP}$.

P.D. $L \leq_p \mathrm{SAT}$, i.e. existe una función $f : \{0,1\}^* \to \{0,1\}^*$ tal que

1. f es computable en tiempo polinomial
2. para $w \in \{0,1\}^*$, se tiene que $w \in L$ si, y solo si, $f(w) \in \mathrm{SAT}$

Denotaremos a tal reducción como $f(w) = \varphi_w$

Dado L fijo, ¿qué representará la fórmula φ_w?

Problemas NP-completos

Demostración

Como $L \in \mathrm{NP}$, existe una máquina no determinista $\mathcal{M}$ tal que

- $L = L(\mathcal{M})$
- $t_{\mathcal{M}}$ es $\mathcal{O}(n^k)$, para algún $k > 0$ fijo

Disponemos de $\mathcal{M}$ y construiremos una fórmula que codificará el funcionamiento de $\mathcal{M}$ con entrada cualquiera w:

$$\mathcal{M} \text{ acepta } w \quad \Leftrightarrow \quad \varphi_w \text{ es satisfacible}$$

¿Qué principios de funcionamiento debemos codificar en φ_w?

Demostración

Sin pérdida de generalidad, supondremos que $\mathcal{M}$ cumple ciertas condiciones que facilitarán la codificación

- $\mathcal{M} = (Q, \{0,1\}, q_0, \{q_m\}, \Delta)$ y no existe transición en Δ para el estado final q_m
- Para cada combinación de estado $q \in (Q \setminus \{q_m\})$ y símbolo $a \in \{0, 1, \sqcup\}$ existe al menos una transición en Δ
- La palabra cumple $w = a_0 \cdots a_{n-1}$, con $a_i \in \{0,1\}$. Si $n = 0$, entonces $w = \epsilon$

¿Qué variables proposicionales necesitamos?

Problemas NP-completos

Demostración

Codificaremos mediante variables los siguientes componentes de $\mathcal{M}$

- Símbolos en la cinta

$$s_{t,p,a} \quad \text{con } 0 \leq t \leq t_{\mathcal{M}}(n),\ -t_{\mathcal{M}}(n) \leq p \leq t_{\mathcal{M}}(n),\ a \in \{0, 1, \sqcup\}$$

- Posición de la cabeza

$$c_{t,p} \quad \text{con } 0 \leq t \leq t_{\mathcal{M}}(n),\ -t_{\mathcal{M}}(n) \leq p \leq t_{\mathcal{M}}(n)$$

- Estado de la cabeza

$$e_{t,q} \quad \text{con } 0 \leq t \leq t_{\mathcal{M}}(n),\ q \in Q$$

Notemos que t indica el tiempo como paso y p una posición en la cinta infinita.

Con estas variables, ¿qué restricciones codificaremos?

Demostración

1. Inicialización: fórmula φ_i

$$\left(c_{0,0}\right) \wedge \left(e_{0,q_0}\right) \wedge \left(\bigwedge_{p=-t_{\mathcal{M}}(n)}^{-1} s_{0,p,\sqcup}\right) \wedge \left(\bigwedge_{p=0}^{n-1} s_{0,p,a_p}\right) \wedge \left(\bigwedge_{p=n}^{t_{\mathcal{M}}(n)} s_{0,p,\sqcup}\right)$$

Cada paréntesis codifica respectivamente:

- Cabeza en posición inicial (arbitraria)
- Estado inicial
- Blancos antes de la palabra
- Palabra escrita desde posición 0 a $n-1$
- Blancos después de la palabra

¿Cómo se ve φ_i cuando $w = \epsilon$?

Demostración

2. Funcionamiento correcto: fórmula φ_c

   Primero, pedimos que cada celda tenga un único símbolo

$$\bigwedge_{t=0}^{t_{\mathcal{M}}(n)} \bigwedge_{p=-t_{\mathcal{M}}(n)}^{t_{\mathcal{M}}(n)} \Bigg(\big(s_{t,p,0} \vee s_{t,p,1} \vee s_{t,p,\sqcup} \big) \wedge \big(s_{t,p,0} \rightarrow (\neg s_{t,p,1} \wedge \neg s_{t,p,\sqcup}) \big) \wedge \big(s_{t,p,1} \rightarrow (\neg s_{t,p,0} \wedge \neg s_{t,p,\sqcup}) \big) \wedge \big(s_{t,p,\sqcup} \rightarrow (\neg s_{t,p,0} \wedge \neg s_{t,p,1}) \big) \Bigg)$$

   Segundo, la máquina tiene un único estado en todo momento

$$\bigwedge_{t=0}^{t_{\mathcal{M}}(n)} \Bigg(\bigvee_{q \in Q} \Bigg(e_{t,q} \wedge \bigwedge_{k \in (Q \setminus \{q\})} \neg e_{t,k} \Bigg) \Bigg)$$

Demostración

2. Funcionamiento correcto: fórmula φ_c

Tercero, la cabeza siempre está en una única posición

$$\bigwedge_{t=0}^{t_{\mathcal{M}}(n)} \left(\bigvee_{p=-t_{\mathcal{M}}(n)}^{t_{\mathcal{M}}(n)} \left(c_{t,p} \wedge \bigwedge_{k \in ([-t_{\mathcal{M}}(n), t_{\mathcal{M}}(n)] \setminus \{p\})} \neg c_{t,k} \right) \right)$$

Cuarto, el valor de una celda no cambia si no es apuntada por la cabeza

$$\bigwedge_{t=0}^{t_{\mathcal{M}}(n)-1} \bigwedge_{p=-t_{\mathcal{M}}(n)}^{t_{\mathcal{M}}(n)} \Bigg(\neg c_{t,p} \to \Big[(s_{t,p,0} \wedge s_{t+1,p,0}) \vee (s_{t,p,1} \wedge s_{t+1,p,1}) \vee (s_{t,p,\sqcup} \wedge s_{t+1,p,\sqcup}) \Big] \Bigg)$$

La fórmula φ_c se define como la conjunción estas cuatro subfórmulas.

Problemas NP-completos

Demostración

3. Transiciones legales: fórmula φ_Δ

   Representando $\triangleleft$ como -1 y $\triangleright$ como $+1$

$$\bigwedge_{t=0}^{t_{\mathcal{M}}(n)-1} \bigwedge_{p=-(t_{\mathcal{M}}(n)-1)}^{t_{\mathcal{M}}(n)} \left(\bigwedge_{(q,a)\in Q\times\{0,1,\sqcup\}} \left[(e_{t,q} \wedge c_{t,p} \wedge s_{t,p,a}) \rightarrow \bigvee_{(q',a',k):(q,a,q',a',k)\in\Delta} (e_{t+1,q'} \wedge c_{t+1,p+k} \wedge s_{t+1,p,a'}) \right] \right)$$

4. La máquina acepta w: fórmula φ_a

$$\bigvee_{t=0}^{t_{\mathcal{M}}(n)} e_{t,q_m}$$

La fórmula buscada es $\varphi_w = \varphi_i \wedge \varphi_c \wedge \varphi_\Delta \wedge \varphi_a$ (¿por qué?). $\square$

Problemas NP-completos

Genial, tenemos un problema NP-completo... ¿es el único?

Lema

Sea $\mathcal{C}$ una clase de complejidad y L_1, L_2 dos lenguajes. Si L_1 es $\mathcal{C}$-hard y además $L_1 \leq_p L_2$, entonces L_2 es $\mathcal{C}$-hard.

Para probar que L es NP-completo:

1. Demostrar que $L \in \mathrm{NP}$
2. Demostrar existe una reducción polinomial desde un **problema NP-completo conocido** a L

Esta estrategia es la que permite probar que muchos problemas de interés práctico son NP-completos

Programa

Obertura

Acto único
- NP-completitud
- Más problemas NP-completos
- Casos tratables

Epílogo

Los familiares de SAT

La demostración del Teorema de Cook nos sugiere una pregunta

- Sabemos que toda fórmula φ tiene una equivalente en CNF y DNF...
- ¿$3\mathrm{SAT}$ y $\mathrm{DNF}\text{-}\mathrm{SAT}$ son NP-completos?
- ¿Podríamos reciclar la demo para probar que otros problemas son NP-completos?

Los familiares de SAT

Teorema

El siguiente problema es NP-completo

$$\mathrm{CNF\text{-}SAT} = \{\varphi \mid \varphi \text{ es una fórmula en CNF satisfacible}\}$$

¿Cómo se demuestra?

- Adaptación de la demostración del Teorema de Cook

¿Se puede hacer lo mismo con DNF?

Los familiares de SAT

Teorema

El siguiente problema está en P

$$\mathrm{DNF\text{-}SAT} = \{\varphi \mid \varphi \text{ es una fórmula en DNF satisfacible}\}$$

¿Es NP-completo? Nadie sabe cómo demostrarlo/refutarlo

Si $\mathrm{DNF\text{-}SAT}$ es NP-completo, concluímos que P=NP!!

Los familiares de SAT

Teorema

El siguiente problema es NP-completo

$$3\mathrm{SAT} = \{\varphi \mid \varphi \text{ es una fórmula en 3CNF satisfacible}\}$$

¿Cómo se demuestra?

- Dada una fórmula φ en CNF...
- se construye una fórmula ψ en 3CNF **con más variables** tal que

$$\varphi \text{ es satisfacible} \quad \Leftrightarrow \quad \psi \text{ es satisfacible}$$

- Esto toma tiempo polinomial!

El método de conversión de CNF a 3CNF es la reducción

$$\mathrm{CNF\text{-}SAT} \leq_p 3\mathrm{SAT}$$

Más problemas NP-completos

NP-completos hasta ahora

- SAT
- CNF-SAT
- 3SAT

¿Tenemos más?

Hace unas clases probamos que un problema X cumple $\text{3SAT} \leq_p X$

Más problemas NP-completos

Ya demostramos que existe una reducción de $3\mathrm{SAT}$ a

$$3\mathrm{COL} = \{G \mid G \text{ grafo no dirigido 3-coloreable}\}$$

Ahora basta probar que $3\mathrm{COL} \in \mathsf{NP}$ para concluir:

Teorema

$3\mathrm{COL}$ es NP-completo

¿Cómo probamos que $3\mathrm{COL} \in \mathsf{NP}$?

Más problemas NP-completos

Demostración

Usando la definición alternativa de NP (testigos y verificadores) tenemos que para un grafo G 3 coloreable

- existe un testigo y de tamaño polinomial: la asignación de colores
- existe una máquina $\mathcal{M}$ que en tiempo polinomial verifica que la asignación de colores sea válida

Concluímos que $3\mathrm{COL} \in$ NP. $\square$

¿Cómo se demuestra si usamos la definición inicial de NP?

Más problemas NP-completos

Demostración (segunda forma)

Para probar que $3\mathrm{COL} \in \mathsf{NP}$, debemos mostrar una máquina **no determinista** $\mathcal{N}$ que acepta $3\mathrm{COL}$ y funciona en **tiempo polinomial**. La definimos según:

1. Si el input *w* no es grafo, se rechaza.
2. Se *adivina* una coloración de forma no determinista
3. Se enlaza con una máquina $\mathcal{M}$ que en tiempo polinomial sobre el número de aristas, revisa que la coloración sea válida
4. Si es válida, se acepta el input *w*

Si el grafo es 3-coloreable, existe una coloración que es *adivinable* en el paso 2 y que satisface el paso 4. Concluímos que $3\mathrm{COL} \in \mathsf{NP}$. □

Notemos que solo nos interesa la aceptación debido a la definición de lenguaje aceptado en máquinas no deterministas

Más problemas NP-completos

Se puede extender este resultado para el problema

$$k\text{-}\mathrm{COL} = \{\, G \mid G \text{ grafo no dirigido } k\text{-coloreable}\}$$

Teorema

k-COL es NP-completo para $k \geq 3$

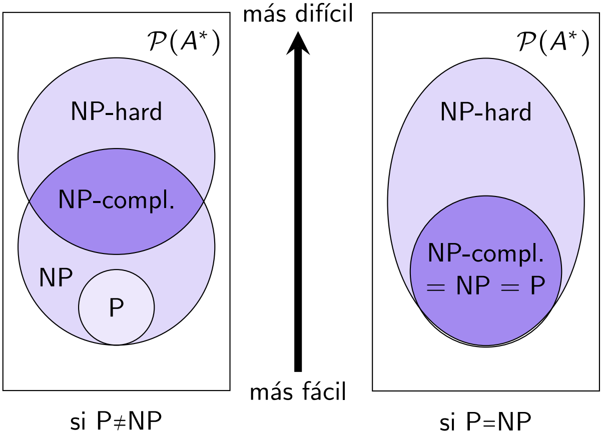
Demostración (Propuesta)

Relaciones de complejidad

Dado que la pregunta ¿P=NP? es un problema abierto. . .

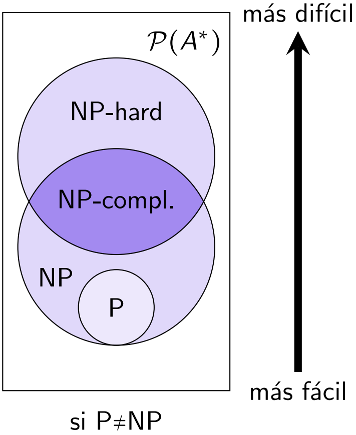
. . . la relación entre problemas en P, en NP, NP-completos y NP-hard puede tomar dos formas

Relaciones de complejidad



¿Dónde se ubican problemas que conocemos?

Relaciones de complejidad

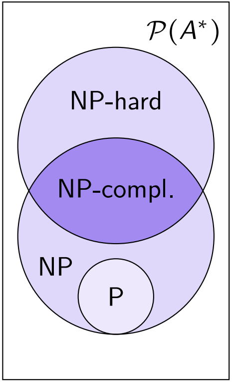


Si P ≠ NP, tenemos ejemplos en cada sector

- P: EVAL
- NP-completo: SAT
- NP-hard fuera de NP: HALTING
- NP fuera de P y no NP-completo???

¿Hay algún ejemplo en ese último sector?

Relaciones de complejidad



si P≠NP

El problema de isomorfismo de grafos

$$\mathrm{GI} = \{(G_1, G_2) \mid G_1 \cong G_2\}$$

ha sido muy estudiado

- No sabemos si está en P
- No sabemos si es NP-completo
- Podría estar en una clase intermedia

Lo importante: no todo problema en NP se sabe NP-completo

En la práctica,

¿qué hacer con un problema NP-completo?

Programa

Obertura

Acto único

- NP-completitud
- Más problemas NP-completos
- Casos tratables

Epílogo

Casos tratables de problemas NP-completos

Dado un problema NP-completo L...

- ¿existe algún caso particular que se puede resolver eficientemente?
- ¿son interesantes tales casos? (i.e. no triviales)

Veremos dos ejemplos de casos tratables

Casos tratables de problemas NP-completos

Considere el siguiente caso de k-COL

$$2\text{-}\mathrm{COL} = \{G \mid G \text{ grafo no dirigido 2-coloreable}\}$$

Teorema

2-COL está en P

¿Cómo se demuestra?

Casos tratables de problemas NP-completos

Considere el siguiente caso de $\mathrm{CNF\text{-}SAT}$

$$\mathrm{2\text{-}CNF\text{-}SAT} = \{\varphi \mid \varphi \text{ fórmula en 2CNF satisfacible}\}$$

Teorema

$\mathrm{2\text{-}CNF\text{-}SAT}$ está en P

¿Cómo se demuestra?

Programa

Obertura

Acto único
- NP-completitud
- Más problemas NP-completos
- Casos tratables

Epílogo

Objetivos de la clase

- ☐ Demostrar que SAT es NP-completo
- ☐ Conocer casos de SAT NP-completos y polinomiales
- ☐ Comprender la relación entre las clases P y NP
- ☐ Conocer casos tratables de problemas NP-completos

¿Qué aprendí hoy? ¿Comentarios?

Ve a

www.menti.com

Introduce el código

6925 2034

O usa el código QR