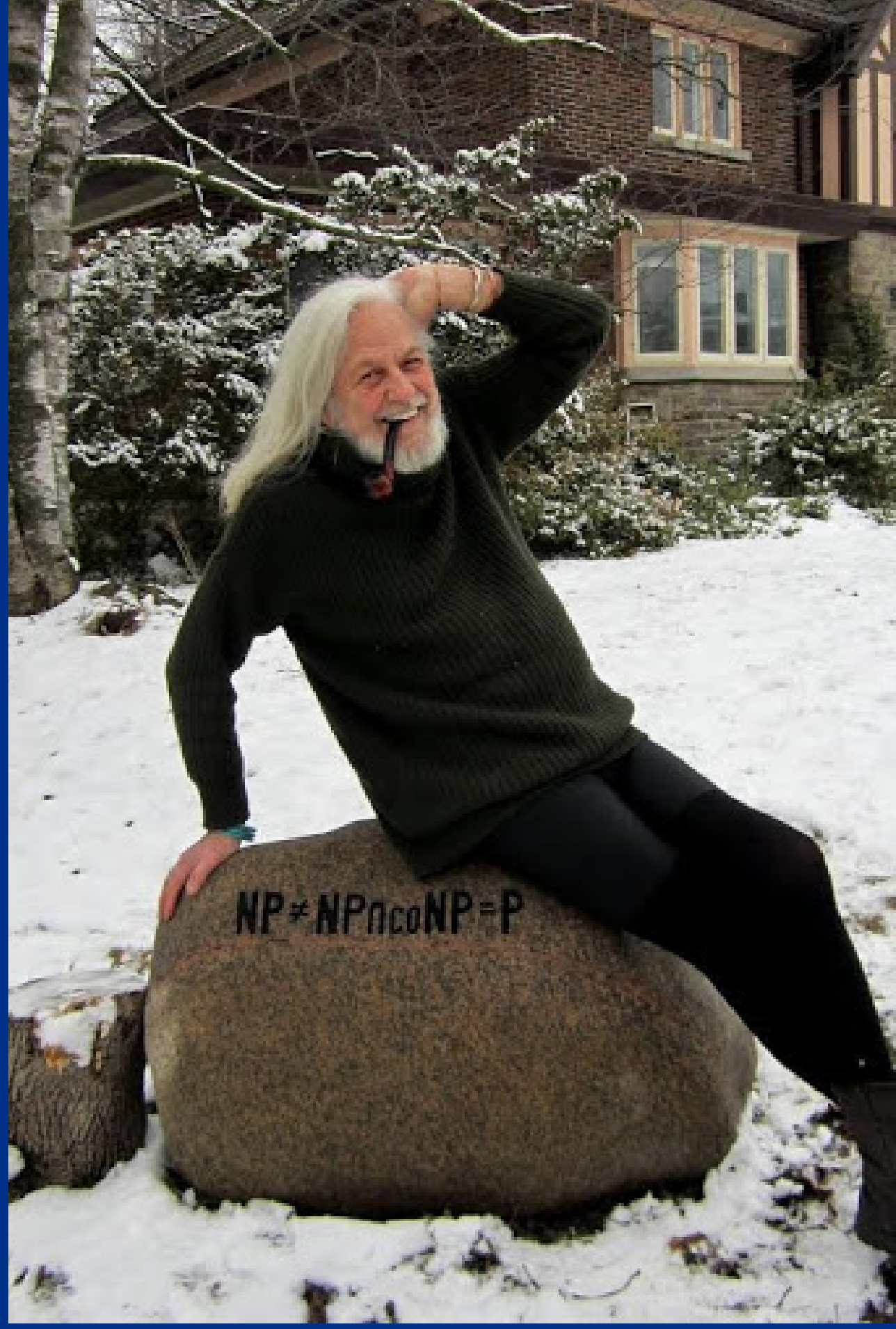
Tiempo y eficiencia



UNIVERSIDAD DE CHILE

Arturo Merino

Plan

Recuerdo y plan

- **Recuerdo:**

Recuerdo y plan

- **Recuerdo:**
  - Máquinas de Turing y sus variantes

Recuerdo y plan

- **Recuerdo:**
  - Máquinas de Turing y sus variantes
  - Lenguajes decidibles e indecidibles

Recuerdo y plan

- **Recuerdo:**
  - Máquinas de Turing y sus variantes
  - Lenguajes decidibles e indecidibles

- **Plan:**

Recuerdo y plan

- **Recuerdo:**
  - Máquinas de Turing y sus variantes
  - Lenguajes decidibles e indecidibles
- **Plan:**
  - Eficiencia

Recuerdo y plan

- **Recuerdo:**
  - Máquinas de Turing y sus variantes
  - Lenguajes decidibles e indecidibles
- **Plan:**
  - Eficiencia
  - La clase P

Recuerdo y plan

- **Recuerdo:**
  - Máquinas de Turing y sus variantes
  - Lenguajes decidibles e indecidibles

- **Plan:**
  - Eficiencia
  - La clase P
  - La clase NP

Eficiencia

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?
  - Una transición $\rightarrow$ un paso

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?
  - Una transición $\rightarrow$ un paso
- El tiempo depende del input

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?
  - Una transición $\rightarrow$ un paso
- El tiempo depende del input

Def.

Sea M una MT determinista que siempre se detiene. Su *tiempo en el peor caso* es

$$T_M(n) = \max_{w \in \Sigma^n} \{\text{número de pasos de } M \text{ sobre } w\}.$$

El tamaño del input es $n = |w|$.

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?
  - Una transición $\rightarrow$ un paso
- El tiempo depende del input

Def.

Sea M una MT determinista que siempre se detiene. Su *tiempo en el peor caso* es

$$T_M(n) = \max_{w \in \Sigma^n} \{\text{número de pasos de } M \text{ sobre } w\}.$$

El tamaño del input es $n = |w|$.

- **Ej:**

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?
  - Una transición $\rightarrow$ un paso
- El tiempo depende del input

Def.

Sea M una MT determinista que siempre se detiene. Su *tiempo en el peor caso* es

$$T_M(n) = \max_{w \in \Sigma^n} \{\text{número de pasos de } M \text{ sobre } w\}.$$

El tamaño del input es $n = |w|$.

- **Ej:**
  - Leer el input y volver al inicio. ¿Cuántos pasos?

Tiempo de cómputo

- ¿Cómo medimos la eficiencia de una MT?
  - Una transición $\rightarrow$ un paso
- El tiempo depende del input

Def.

Sea M una MT determinista que siempre se detiene. Su *tiempo en el peor caso* es

$$T_M(n) = \max_{w \in \Sigma^n} \{\text{número de pasos de } M \text{ sobre } w\}.$$

El tamaño del input es $n = |w|$.

- **Ej:**
  - Leer el input y volver al inicio. ¿Cuántos pasos?
  - Un entero m en binario. ¿Cuál es el largo del input?

Clases de tiempo

- Un mismo lenguaje, distintas MT

Clases de tiempo

- Un mismo lenguaje, distintas MT
  - ¿Qué tan rápido podemos decidirlo?

Clases de tiempo

- Un mismo lenguaje, distintas MT
  - ¿Qué tan rápido podemos decidirlo?

Def.

Sea $t : \mathbb{N} \to \mathbb{N}$. La *clase de tiempo* $\mathrm{TIME}(t(n))$ es

$$\{L : \text{una MT determinista de una cinta decide } L \text{ en tiempo } O(t(n))\}.$$

Clases de tiempo

- Un mismo lenguaje, distintas MT
  - ¿Qué tan rápido podemos decidirlo?

Def.

Sea $t : \mathbb{N} \to \mathbb{N}$. La *clase de tiempo* $\mathrm{TIME}(t(n))$ es

$$\{L : \text{una MT determinista de una cinta decide } L \text{ en tiempo } O(t(n))\}.$$

- ¿Y si usamos varias cintas?

Clases de tiempo

- Un mismo lenguaje, distintas MT
  - ¿Qué tan rápido podemos decidirlo?

Def.

Sea $t : \mathbb{N} \to \mathbb{N}$. La *clase de tiempo* $\mathrm{TIME}(t(n))$ es

$$\{L : \text{una MT determinista de una cinta decide } L \text{ en tiempo } O(t(n))\}.$$

- ¿Y si usamos varias cintas?

Teo.

Sea $t(n) \geq n$. Toda MT determinista multicinta de tiempo $t(n)$ tiene una MT de una cinta equivalente de tiempo $O(t(n)^2)$.

Clases de tiempo

- Un mismo lenguaje, distintas MT
  - ¿Qué tan rápido podemos decidirlo?

Def.

Sea $t : \mathbb{N} \to \mathbb{N}$. La *clase de tiempo* $\mathrm{TIME}(t(n))$ es

$$\{L : \text{una MT determinista de una cinta decide } L \text{ en tiempo } O(t(n))\}.$$

- ¿Y si usamos varias cintas?

Teo.

Sea $t(n) \geq n$. Toda MT determinista multicinta de tiempo $t(n)$ tiene una MT de una cinta equivalente de tiempo $O(t(n)^2)$.

- **Dem:**

La clase P

La clase P

- ¿Lenguajes eficientemente computables?

La clase P

- ¿Lenguajes eficientemente computables?
- *Un programa que sólo llama subrutinas eficientes es eficiente*

La clase P

- ¿Lenguajes eficientemente computables?
- *Un programa que sólo llama subrutinas eficientes es eficiente*

Def.

La clase P contiene los lenguajes decidibles en tiempo polinomial por una MT determinista de una cinta.

$$\mathrm{P} = \bigcup_{k \geq 1} \mathrm{TIME}(n^k).$$

La clase P

- ¿Lenguajes eficientemente computables?
- *Un programa que sólo llama subrutinas eficientes es eficiente*

Def.

La clase P contiene los lenguajes decidibles en tiempo polinomial por una MT determinista de una cinta.

$$\mathrm{P} = \bigcup_{k \geq 1} \mathrm{TIME}(n^k).$$

- *Tesis fuerte de Church–Turing*

La clase P

- ¿Lenguajes eficientemente computables?
- *Un programa que sólo llama subrutinas eficientes es eficiente*

Def.

La clase P contiene los lenguajes decidibles en tiempo polinomial por una MT determinista de una cinta.

$$\mathrm{P} = \bigcup_{k \geq 1} \mathrm{TIME}(n^k).$$

- *Tesis fuerte de Church–Turing*
  - Todo modelo razonable de cómputo clásico, secuencial y determinista se puede simular con una MT con overhead polinomial.

La clase P

- ¿Lenguajes eficientemente computables?
- *Un programa que sólo llama subrutinas eficientes es eficiente*

Def.

La clase P contiene los lenguajes decidibles en tiempo polinomial por una MT determinista de una cinta.

$$\mathrm{P} = \bigcup_{k \geq 1} \mathrm{TIME}(n^k).$$

- *Tesis fuerte de Church–Turing*
  - Todo modelo razonable de cómputo clásico, secuencial y determinista se puede simular con una MT con overhead polinomial.

- n^{100}. ¿Eficiente?

Un camino entre dos vértices

- Probar que $L \in \mathsf{P} \rightarrow$ diseñar un algoritmo polinomial

Un camino entre dos vértices

- Probar que $L \in \mathrm{P} \rightarrow$ diseñar un algoritmo polinomial
  - Número de etapas y costo por etapa

Un camino entre dos vértices

- Probar que $L \in \mathsf{P} \rightarrow$ diseñar un algoritmo polinomial
  - Número de etapas y costo por etapa
- **Ej:** G dirigido, s, t vértices de G.

Un camino entre dos vértices

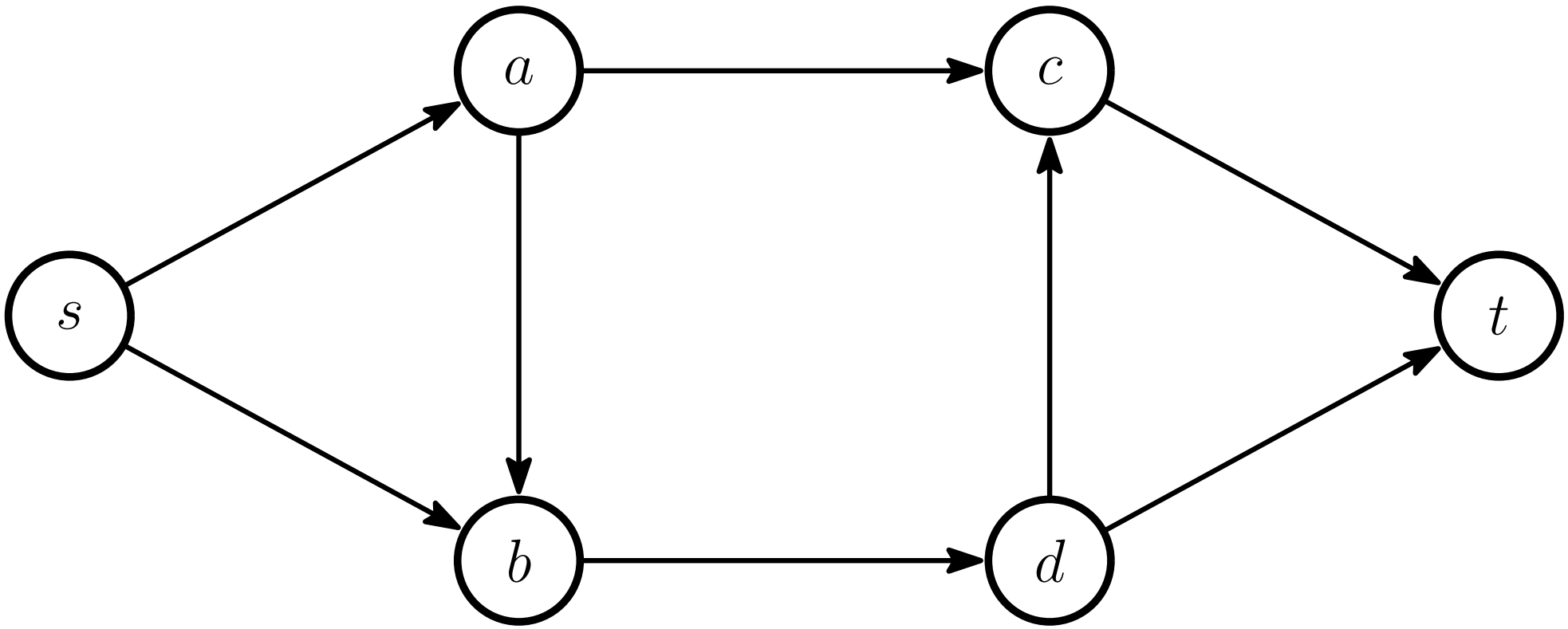
- Probar que $L \in \mathsf{P} \rightarrow$ diseñar un algoritmo polinomial
  - Número de etapas y costo por etapa
- **Ej:** G dirigido, s, t vértices de G.
  - $\mathrm{PATH} = \{\langle G, s, t\rangle$: hay un camino de s a t en $G\}$.

Un camino entre dos vértices

- Probar que $L \in \mathrm{P} \rightarrow$ diseñar un algoritmo polinomial
  - Número de etapas y costo por etapa
- **Ej:** G dirigido, s, t vértices de G.
  - $\mathrm{PATH} = \{\langle G, s, t\rangle$: hay un camino de s a t en $G\}$.
- Demostremos que $\mathrm{PATH} \in \mathrm{P}$

Un camino entre dos vértices

- Probar que $L \in \mathrm{P} \rightarrow$ diseñar un algoritmo polinomial
  - Número de etapas y costo por etapa
- **Ej:** G dirigido, s, t vértices de G.
  - $\mathrm{PATH} = \{\langle G, s, t\rangle$: hay un camino de s a t en $G\}$.
- Demostremos que $\mathrm{PATH} \in \mathrm{P}$
- **Dem:**



Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.

Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.
  - $\text{RELPRIME} = \{\langle a, b \rangle : \text{mcd}(a, b) = 1\}$.

Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.
  - $\text{RELPRIME} = \{\langle a, b \rangle : \text{mcd}(a, b) = 1\}$.
- Demostremos que $\text{RELPRIME} \in$ P

Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.
  - $\text{RELPRIME} = \{\langle a, b \rangle : \text{mcd}(a, b) = 1\}$.
- Demostremos que $\text{RELPRIME} \in$ P
- **Dem:**

Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.
  - $\text{RELPRIME} = \{\langle a, b \rangle : \text{mcd}(a, b) = 1\}$.
- Demostremos que $\text{RELPRIME} \in$ P
- **Dem:**

- ¿Lenguajes libres de contexto?

Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.
  - $\text{RELPRIME} = \{\langle a, b\rangle : \text{mcd}(a, b) = 1\}$.
- Demostremos que $\text{RELPRIME} \in$ P
- **Dem:**

- ¿Lenguajes libres de contexto?

Teo.

Todo lenguaje libre de contexto pertenece a P.

Más ejemplos

- **Ej:** $a, b \geq 1$, escritos en binario.
  - $\text{RELPRIME} = \{\langle a, b\rangle : \text{mcd}(a, b) = 1\}$.
- Demostremos que $\text{RELPRIME} \in$ P
- **Dem:**

- ¿Lenguajes libres de contexto?

Teo.

Todo lenguaje libre de contexto pertenece a P.

- **Dem:**

La clase NP

Encontrar y verificar

- **Ej:** G dirigido, s, t vértices de G.

Encontrar y verificar

- **Ej:** G dirigido, s, t vértices de G.
  - Antes: un camino de s a t

Encontrar y verificar

- **Ej:** G dirigido, s, t vértices de G.
  - Antes: un camino de s a t
  - Ahora: pasar por todos los vértices, sin repetir

Encontrar y verificar

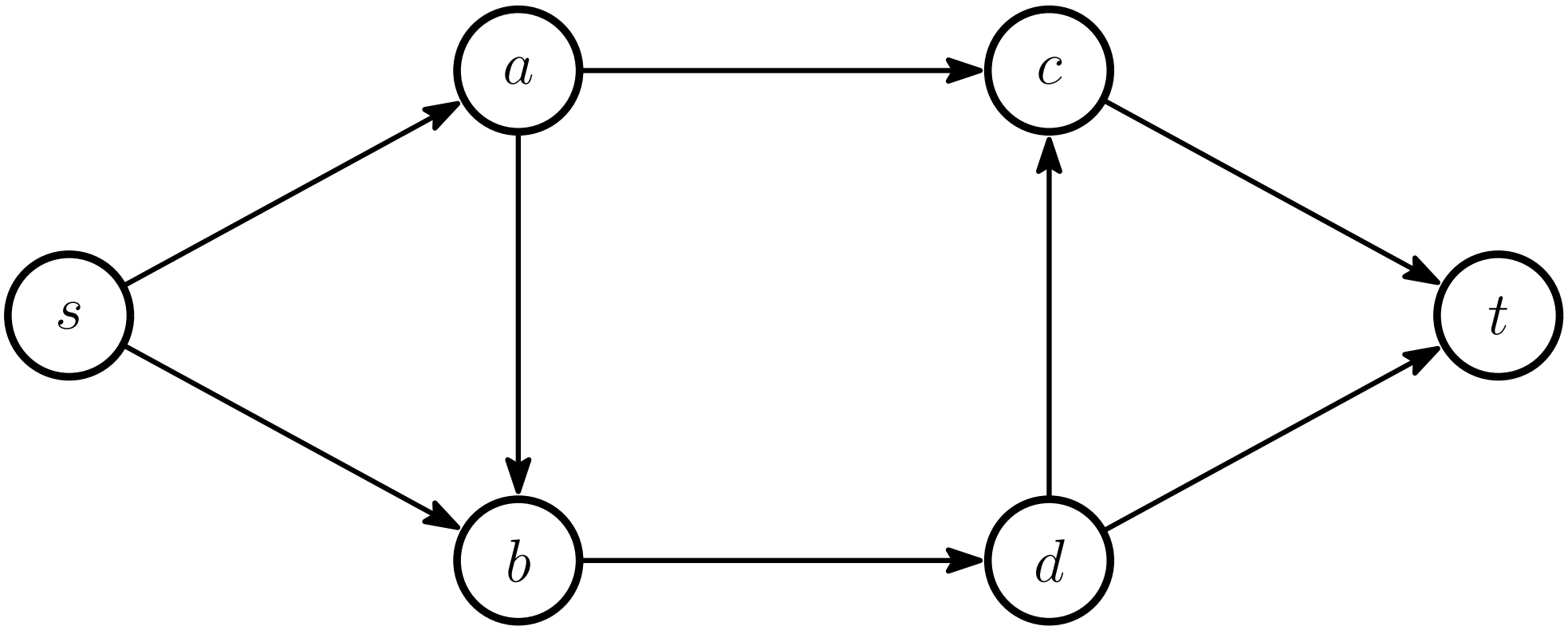
- **Ej:** G dirigido, s, t vértices de G.
  - Antes: un camino de s a t
  - Ahora: pasar por todos los vértices, sin repetir
  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t \rangle$ con tal camino

Encontrar y verificar

- **Ej:** G dirigido, s, t vértices de G.
  - Antes: un camino de s a t
  - Ahora: pasar por todos los vértices, sin repetir
  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t \rangle$ con tal camino
- ¿Resolver?

Encontrar y verificar

- **Ej:** G dirigido, s, t vértices de G.
  - Antes: un camino de s a t
  - Ahora: pasar por todos los vértices, sin repetir
  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t \rangle$ con tal camino
- ¿Resolver?
- Verificar



Verificadores

- Formalicemos la idea de verificar una solución

Verificadores

- Formalicemos la idea de verificar una solución

Def.

Un *verificador polinomial* para $L \subseteq \Sigma^*$ es una MT determinista V, de tiempo polinomial en $|w| + |c|$, junto con un polinomio p, tal que

$$w \in L \iff \exists c \in \{0,1\}^*, \quad |c| \leq p(|w|), \quad V(w,c) = 1.$$

c es un *certificado*.

Verificadores

- Formalicemos la idea de verificar una solución

Def.

Un *verificador polinomial* para $L \subseteq \Sigma^*$ es una MT determinista V, de tiempo polinomial en $|w| + |c|$, junto con un polinomio p, tal que

$$w \in L \iff \exists c \in \{0,1\}^*, \quad |c| \leq p(|w|), \quad V(w,c) = 1.$$

c es un *certificado*.

- NP $\leftarrow$ clase de lenguajes verificables eficientemente

Verificadores

- Formalicemos la idea de verificar una solución

Def.

Un *verificador polinomial* para $L \subseteq \Sigma^*$ es una MT determinista V, de tiempo polinomial en $|w| + |c|$, junto con un polinomio p, tal que

$$w \in L \iff \exists c \in \{0,1\}^*, \quad |c| \leq p(|w|), \quad V(w,c) = 1.$$

c es un *certificado*.

- NP $\leftarrow$ clase de lenguajes verificables eficientemente

Def.

La clase NP contiene los lenguajes que tienen un verificador polinomial.

Verificadores

- Formalicemos la idea de verificar una solución

Def.

Un *verificador polinomial* para $L \subseteq \Sigma^*$ es una MT determinista V, de tiempo polinomial en $|w| + |c|$, junto con un polinomio p, tal que

$$w \in L \iff \exists c \in \{0,1\}^*, \quad |c| \leq p(|w|), \quad V(w,c) = 1.$$

c es un *certificado*.

- NP $\leftarrow$ clase de lenguajes verificables eficientemente

Def.

La clase NP contiene los lenguajes que tienen un verificador polinomial.

- ¿NP?

Verificadores

- Formalicemos la idea de verificar una solución

Def.

Un *verificador polinomial* para $L \subseteq \Sigma^*$ es una MT determinista V, de tiempo polinomial en $|w| + |c|$, junto con un polinomio p, tal que

$$w \in L \iff \exists c \in \{0,1\}^*, \quad |c| \leq p(|w|), \quad V(w,c) = 1.$$

c es un *certificado*.

- NP $\leftarrow$ clase de lenguajes verificables eficientemente

Def.

La clase NP contiene los lenguajes que tienen un verificador polinomial.

- ¿NP?
- Tiempo polinomial **no determinista**

Tiempo no determinista

- ¿Cómo medimos el tiempo cuando hay varias ramas?

Tiempo no determinista

- ¿Cómo medimos el tiempo cuando hay varias ramas?

Def.

Para una MTND que siempre termina, el *tiempo* es el máximo de pasos sobre todas las ramas y todos los inputs de largo n.

$$\mathrm{NTIME}(t(n)) = \{L : \text{una MTND decide } L \text{ en tiempo } O(t(n))\}.$$

Tiempo no determinista

- ¿Cómo medimos el tiempo cuando hay varias ramas?

Def.

Para una MTND que siempre termina, el *tiempo* es el máximo de pasos sobre todas las ramas y todos los inputs de largo n.

$$\mathrm{NTIME}(t(n)) = \{L : \text{una MTND decide } L \text{ en tiempo } O(t(n))\}.$$

Teo.

Un lenguaje está en NP si y solo si alguna MTND lo decide en tiempo polinomial. En particular,

$$\mathsf{NP} = \bigcup_{k \geq 1} \mathrm{NTIME}(n^k).$$

Tiempo no determinista

- ¿Cómo medimos el tiempo cuando hay varias ramas?

Def.

Para una MTND que siempre termina, el *tiempo* es el máximo de pasos sobre todas las ramas y todos los inputs de largo n.

$$\mathrm{NTIME}(t(n)) = \{L : \text{una MTND decide } L \text{ en tiempo } O(t(n))\}.$$

Teo.

Un lenguaje está en NP si y solo si alguna MTND lo decide en tiempo polinomial. En particular,

$$\mathsf{NP} = \bigcup_{k \geq 1} \mathrm{NTIME}(n^k).$$

- **Dem:**

Certificados

- **Ej:** Demostremos que los siguientes problemas están en NP

Certificados

- **Ej:** Demostremos que los siguientes problemas están en NP

  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t \rangle$ con s-t camino Hamiltoniano.

Certificados

- **Ej:** Demostremos que los siguientes problemas están en NP

  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t \rangle$ con s-t camino Hamiltoniano.

  - $a_1, \ldots, a_m, t \in \mathbb{N}$, escritos en binario.

Certificados

- **Ej:** Demostremos que los siguientes problemas están en NP

  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t\rangle$ con s-t camino Hamiltoniano.

  - $a_1, \ldots, a_m, t \in \mathbb{N}$, escritos en binario.

$$\mathrm{SUBSET\text{-}SUM} = \Big\{\langle a_1, \ldots, a_m, t\rangle : \exists I \subseteq \{1, \ldots, m\},\ \sum_{i \in I} a_i = t\Big\}.$$

Certificados

- **Ej:** Demostremos que los siguientes problemas están en NP

  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t\rangle$ con s-t camino Hamiltoniano.

  - $a_1, \ldots, a_m, t \in \mathbb{N}$, escritos en binario.

$$\mathrm{SUBSET\text{-}SUM} = \Big\{\langle a_1, \ldots, a_m, t\rangle : \exists I \subseteq \{1, \ldots, m\},\ \sum_{i \in I} a_i = t\Big\}.$$

  - $\mathrm{CLIQUE} = \{\langle G, k\rangle : G$ tiene k vértices todos adyacentes entre sí$\}$.

Certificados

- **Ej:** Demostremos que los siguientes problemas están en NP

  - $\mathrm{HAMPATH}$: codificaciones $\langle G, s, t\rangle$ con s-t camino Hamiltoniano.

  - $a_1, \ldots, a_m, t \in \mathbb{N}$, escritos en binario.

$$\mathrm{SUBSET\text{-}SUM} = \Big\{\langle a_1, \ldots, a_m, t\rangle : \exists I \subseteq \{1, \ldots, m\},\ \sum_{i \in I} a_i = t\Big\}.$$

  - $\mathrm{CLIQUE} = \{\langle G, k\rangle : G$ tiene k vértices todos adyacentes entre sí$\}$.

  - $\mathrm{SAT} = \{\langle \varphi\rangle : \varphi$ es satisfacible$\}$.

P versus NP

- Si podemos decidir, podemos verificar.

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

- Problema del Milenio: premio de US$1 millón.

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

- Problema del Milenio: premio de US$1 millón.
- Pregunta central de la teoría de la computación.

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

- Problema del Milenio: premio de US$1 millón.
- Pregunta central de la teoría de la computación.

- ¿Qué inclusiones podemos probar?

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

- Problema del Milenio: premio de US$1 millón.
- Pregunta central de la teoría de la computación.

- ¿Qué inclusiones podemos probar?

Prop.

$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{EXPTIME}$, donde

$$\mathsf{EXPTIME} = \bigcup_{k \geq 1} \mathrm{TIME}(2^{n^k}).$$

P versus NP

- Si podemos decidir, podemos verificar.
- ¿Si podemos verificar, podemos decidir?

$$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$$

- Problema del Milenio: premio de US$1 millón.
- Pregunta central de la teoría de la computación.

- ¿Qué inclusiones podemos probar?

Prop.

$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{EXPTIME}$, donde

$$\mathsf{EXPTIME} = \bigcup_{k \geq 1} \mathrm{TIME}(2^{n^k}).$$

- **Dem:**

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

- P $\subseteq$ coNP

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

- P $\subseteq$ coNP
- **Dem:**

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

- P $\subseteq$ coNP
- **Dem:**
- ¿NP = coNP? También abierto

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

- P $\subseteq$ coNP
- **Dem:**
- ¿NP = coNP? También abierto
- Conjeturas habituales: P $\neq$ NP y NP $\neq$ coNP.

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

- P $\subseteq$ coNP
- **Dem:**
- ¿NP = coNP? También abierto

- Conjeturas habituales: P $\neq$ NP y NP $\neq$ coNP.
- Survey de Williams: 80% de P $\neq$ NP.

¿Y si la respuesta es no?

- ¿Cómo convencemos a alguien de que no hay un clique de tamaño k?

Def.

La clase coNP contiene los complementos de lenguajes de NP:

$$\text{coNP} = \{L^c : L \in \text{NP}\}.$$

- P $\subseteq$ coNP
- **Dem:**
- ¿NP = coNP? También abierto

- Conjeturas habituales: P $\neq$ NP y NP $\neq$ coNP.
- Survey de Williams: 80% de P $\neq$ NP.
- Knuth: P = NP.

Dudas/Preguntas