

Clases de complejidad aleatorizadas

IIC3810

Un ejemplo: equivalencia de polinomios

Consideramos polinomios en varias variables en $\mathbb{Q}$

Un monomio es una expresión de la forma $cx_1^{\ell_1} \cdots x_n^{\ell_n}$, donde $c \in \mathbb{Q}$ y cada $\ell_i \in \mathbb{N}$.

Un monomio $cx_1^{\ell_1} \cdots x_n^{\ell_n}$ es nulo si $c = 0$

► No es nulo si $c \neq 0$

El grado de un monomio $cx_1^{\ell_1} \cdots x_n^{\ell_n}$ no nulo es $\ell_1 + \cdots + \ell_n$.

Polinomios en varias variables

Un polinomio es una expresión de la forma:

$$p(x_1, \dots, x_n) = \sum_{i=1}^{\ell} \prod_{j=1}^{m_i} \left(\sum_{k=1}^n a_{i,j,k} x_k + a_{i,j,n+1} \right)$$

donde cada $a_{i,j,k} \in \mathbb{Q}$ y cada $a_{i,j,n+1} \in \mathbb{Q}$

Polinomios en varias variables

La forma canónica de un polinomio $p(x_1, \dots, x_n)$ es única, y es igual a 0 o a una suma de monomios que satisface las siguiente propiedades:

- ▶ cada monomio en la forma canónica es de la forma $cx_1^{\ell_1} \cdots x_n^{\ell_n}$ con $c \neq 0$
- ▶ si $cx_1^{\ell_1} \cdots x_n^{\ell_n}$ y $dx_1^{m_1} \cdots x_n^{m_n}$ son dos monomios distintos en la forma canónica, entonces $\ell_i \neq m_i$ para algún $i \in \{1, \dots, n\}$

Un polinomio $p(x_1, \dots, x_n)$ es nulo si su forma canónica es 0

El grado de un polinomio $p(x_1, \dots, x_n)$ no nulo es el mayor grado de los monomios en su forma canónica.

Equivalencia de polinomios en varias variables

Dos polinomios $p(x_1, \dots, x_n)$ y $q(x_1, \dots, x_n)$ son idénticos si para cada secuencia $a_1, \dots, a_n \in \mathbb{Q}$ se tiene que:

$$p(a_1, \dots, a_n) = q(a_1, \dots, a_n)$$

Queremos verificar si dos polinomios son idénticos, para lo cual definimos el siguiente lenguaje:

$$\text{EQUIV-POL} = \{(p(x_1, \dots, x_n), q(x_1, \dots, x_n)) \mid \\ p(x_1, \dots, x_n) \text{ y } q(x_1, \dots, x_n) \text{ son polinomios idénticos}\}$$

Equivalencia de polinomios en varias variables

¿Podemos resolver EQUIV-POL en tiempo polinomial?

Tenemos un problema: calcular la forma canónica de un polinomio toma tiempo exponencial

Vamos a construir un algoritmo aleatorizado para EQUIV-POL

- El ingrediente principal del algoritmo es el lema de Schwartz-Zippel

El ingrediente principal

Lema de Schwartz-Zippel

Sea $p(x_1, \dots, x_n)$ un polinomio no nulo de grado k , y sea A un subconjunto finito y no vacío de $\mathbb{Q}$. Si $a_1, \dots, a_n$ son elegidos de manera uniforme e independiente desde A , entonces

$$\Pr(p(a_1, \dots, a_n) = 0) \leq \frac{k}{|A|}$$

Ejercicio

Demuestre el lema de Schwartz-Zippel por inducción en n

Un algoritmo aleatorizado para EQUIV-POL

Vamos a dar un algoritmo aleatorizado para el problema de verificar si dos polinomios en varias variables son equivalentes.

Suponga que la entrada del algoritmo está dada por los siguientes polinomios:

$$p(x_1, \dots, x_n) = \sum_{i=1}^{\ell} \prod_{j=1}^{m_i} \left(\sum_{k=1}^n a_{i,j,k} x_k + a_{i,j,n+1} \right)$$
$$q(x_1, \dots, x_n) = \sum_{i=1}^r \prod_{j=1}^{s_i} \left(\sum_{k=1}^n b_{i,j,k} x_k + b_{i,j,n+1} \right)$$

Un algoritmo aleatorizado para EQUIV-POL

EquivPolAleatorizado($p(x_1, \dots, x_n)$, $q(x_1, \dots, x_n)$)

$k := 1 + \text{máx} \{m_1, \dots, m_\ell, s_1, \dots, s_r\}$

$A := \{1, \dots, 100 \cdot k\}$

sea $a_1, \dots, a_n$ una secuencia de números elegidos de
manera uniforme e independiente desde A

if $p(a_1, \dots, a_n) = q(a_1, \dots, a_n)$ **then return** sí

else return no

Utilizando el lema de Schwartz-Zippel

Vamos a calcular la probabilidad de error del algoritmo.

- ▶ Si los polinomios $p(x_1, \dots, x_n)$ y $q(x_1, \dots, x_n)$ son equivalentes, entonces el algoritmo responde **sí** sin cometer error
- ▶ Si los polinomios $p(x_1, \dots, x_n)$ y $q(x_1, \dots, x_n)$ no son equivalentes, el algoritmo puede responder **sí** al escoger una secuencia de números $a_1, \dots, a_n$ desde A tales que $p(a_1, \dots, a_n) = q(a_1, \dots, a_n)$
 - ▶ Donde $A = \{1, \dots, 100 \cdot k\}$

Esto significa que $(a_1, \dots, a_n)$ es una raíz del polinomio $r(x_1, \dots, x_n) = p(x_1, \dots, x_n) - q(x_1, \dots, x_n)$

Utilizando el lema de Schwartz-Zippel

$r(x_1, \dots, x_n)$ no es el polinomio nulo y es de grado t con $t < k$

► Dado que $k = 1 + \max\{m_1, \dots, m_\ell, s_1, \dots, s_r\}$

Utilizando el lema de Schwartz-Zippel obtenemos:

$$\Pr(r(a_1, \dots, a_n) = 0) \leq \frac{t}{|A|} \leq \frac{k}{|A|} = \frac{k}{100 \cdot k} = \frac{1}{100}$$

La probabilidad de error del algoritmo está entonces acotada por $\frac{1}{100}$

Un mejor algoritmo aleatorizado para el problema general

Ejercicio

De un algoritmo aleatorizado que resuelva el problema de equivalencia de polinomios en varias variables.

- ▶ La probabilidad de error del algoritmo debe estar acotada por $\frac{1}{100^{10}}$
- ▶ Debe existir una constante k tal que el algoritmo en el peor caso es $O(m^k)$, donde m es el tamaño de la entrada
 - ▶ Si consideramos $p(x_1, \dots, x_n)$ y $q(x_1, \dots, x_n)$ como palabras sobre un cierto alfabeto, entonces $m = |p(x_1, \dots, x_n)| + |q(x_1, \dots, x_n)|$

Una solución para el ejercicio

EquivPolAleatorizado($p(x_1, \dots, x_n)$, $q(x_1, \dots, x_n)$)

$k := 1 + \text{máx} \{m_1, \dots, m_\ell, s_1, \dots, s_r\}$

$A := \{1, \dots, 100 \cdot k\}$

for $i := 1$ **to** 10 **do**

 sea $a_1, \dots, a_n$ una secuencia de números elegidos de
 manera uniforme e independiente desde A

if $p(a_1, \dots, a_n) \neq q(a_1, \dots, a_n)$ **then return** no

else return sí

Algoritmos probabilísticos y Máquinas de Turing

¿Cómo podemos formalizar la idea de un algoritmo probabilístico utilizando la noción de MT?

¿Podemos definir clases de complejidad basados en los algoritmos probabilísticos?

Vamos a responder a estas preguntas en las siguientes transparencias.

MT probabilística

Definición

Una MT probabilística es una tupla $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ tal que:

- ▶ *Q es un conjunto finito de estados*
- ▶ *Σ es un alfabeto finito tal que $\vdash, \sqcup \notin \Sigma$*
- ▶ *Γ es un alfabeto finito tal que $\Sigma \cup \{\vdash, \sqcup\} \subseteq \Gamma$*
- ▶ *$q_0 \in Q$ es el estado inicial*
- ▶ *$F \subseteq Q$ es un conjunto de estados finales*
- ▶ *δ es una función parcial:*

$$\delta : Q \times \Gamma \times \{0, 1\} \rightarrow Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\}$$

MT probabilística: Funcionamiento

La entrada de una MT probabilística M consiste de un string $w \in \Sigma^*$ y un string $s \in \{0, 1\}^\omega$

- ▶ w es el input que se quiere aceptar o rechazar
- ▶ s es un string infinito de símbolos 0 y 1, el cual es considerado como un string de bits aleatorios

En el estado inicial:

- ▶ M tiene en la primera cinta $\vdash wB \cdots$ y en la segunda cinta $\vdash s$
- ▶ M está en el estado q_0
- ▶ Las cabezas lectoras de ambas cintas están en la posición 1

MT probabilística: Funcionamiento

En cada instante la máquina se encuentra en un estado q y sus cabezas lectoras están en posiciones p_1 y p_2

- ▶ Si el símbolo en la posición p_i ($i = 1, 2$) es a_i y $\delta(q, a_1, a_2) = (q', b, X)$, entonces:
 - ▶ La máquina escribe el símbolo b en la posición p_1 de la primera cinta
 - ▶ Cambia de estado desde q a q'
 - ▶ Mueve la cabeza lectora de la primera cinta a la posición $p_1 - 1$ si X es $\leftarrow$, y a la posición $p_1 + 1$ si X es $\rightarrow$. Si X es $\square$, entonces esta cabeza lectora permanece en la posición p_1
 - ▶ Mueve la cabeza lectora de la segunda cinta a la posición $p_2 + 1$

El tiempo de ejecución de una MT probabilística

La entrada de una MT probabilística M con alfabeto Σ consiste de dos strings $w \in \Sigma^*$ y $s \in \{0, 1\}^\omega$

- ▶ Utilizamos la notación $M(w, s)$ para indicar las entradas de M
- ▶ Decimos que $M(w, s)$ acepta si M con entrada (w, s) se detiene en un estado final
 - ▶ El caso en que $M(w, s)$ rechaza se define de forma similar

Primer supuesto

Consideramos una MT probabilística M que se detiene en todas sus entradas (w, s)

El tiempo de ejecución de una MT probabilística

Un paso de una MT probabilística M consiste en ejecutar una instrucción de la función de transición

- Definimos $\text{tiempo}_M(w, s)$ como el número de pasos ejecutados por M con entrada (w, s)

Segundo supuesto

Existe una función $f : \Sigma^* \rightarrow \mathbb{N}$ tal que para cada $w \in \Sigma^*$ y $s \in \{0, 1\}^\omega$:

$$\text{tiempo}_M(w, s) \leq f(w)$$

Vale decir, hay una cantidad máxima de bits aleatorios que deben ser utilizados con entrada w , la cual sólo depende de w

El tiempo de ejecución de una MT probabilística

Para estudiar el peor caso necesitamos la siguiente definición:

$$tiempo_M(w) = \max\{tiempo_M(w, s) \mid s \in \{0, 1\}^\omega\}$$

Con esto tenemos que el tiempo de funcionamiento de M en el peor caso es definido por la función t_M :

$$t_M(n) = \max\{tiempo_M(w) \mid w \in \Sigma^* \text{ y } |w| = n\}$$

La probabilidad de aceptar en una MT probabilística

Tercer supuesto

Si para una MT probabilística M con alfabeto Σ se tiene que $t_M(n) \leq g(n)$ para todo $n \in \mathbb{N}$, entonces suponemos que las entradas de M son de la forma (w, s) con $w \in \Sigma^*$, $s \in \{0, 1\}^*$ y $|s| = g(n)$.

Dado el tiempo de ejecución de M no podemos usar más de $g(n)$ bits aleatorios para una entrada w de largo n .

La probabilidad de aceptar en una MT probabilística

Sea M una MT probabilística con alfabeto Σ y tal que $t_M(n) \leq g(n)$ para todo $n \in \mathbb{N}$.

Definición

Para cada $w \in \Sigma^$ tal que $|w| = n$, la probabilidad de que M acepte w es definida de la siguiente forma:*

$$\mathbf{Pr}_s(M \text{ acepte } w) = \frac{|\{s \in \{0, 1\}^* \mid |s| = g(n) \text{ y } M(w, s) \text{ acepta}\}|}{2^{g(n)}}$$

Clases de complejidad probabilísticas

Vamos a definir una primera clase de complejidad considerando los algoritmos probabilísticos

- ▶ Esto nos va a permitir decir cuando un lenguaje es *aceptado* por una MT probabilística

Definición

Sea L un lenguaje sobre un alfabeto Σ . Entonces L está en RP si existe una MT probabilística M tal que $t_M(n)$ es $O(n^k)$ y para cada $w \in \Sigma^*$:

- ▶ Si $w \in L$, entonces $\Pr(M \text{ acepte } w) \geq \frac{3}{4}$
- ▶ Si $w \notin L$, entonces $\Pr(M \text{ acepte } w) = 0$

Vale decir, para los lenguaje en RP tenemos algoritmos probabilísticos que pueden cometer errores sólo para los elementos que están en L

La clase RP: un ejemplo

Ejercicio

Muestre que $\overline{\text{EQUIV-POL}} \in \text{RP}$

¿Por qué utilizamos la probabilidad $\frac{3}{4}$?

El valor $\frac{3}{4}$ es arbitrario

- Podemos utilizar valores arbitrariamente más pequeños

Lema de amplificación

Sea L un lenguaje sobre un alfabeto Σ . Si $L \in \text{RP}$, entonces para cada $\ell \in \mathbb{N}$, existe una MT probabilística M tal que $t_M(n)$ es $O(n^k)$ y para cada $w \in \Sigma^*$:

- Si $w \in L$, entonces $\Pr(M \text{ acepte } w) \geq 1 - \frac{1}{4^\ell}$
- Si $w \notin L$, entonces $\Pr(M \text{ acepte } w) = 0$

Ejercicio

Demuestre el lema de amplificación

¿Dónde está la clase RP?

Teorema

$$PTIME \subseteq RP \subseteq NP$$

Ejercicio

Demuestre el teorema.

Corolario

$$PTIME \subseteq co-RP \subseteq co-NP$$

¿Qué sabemos sobre RP y co-RP?

Son problemas abiertos si $\text{PTIME} = \text{RP}$ o $\text{RP} = \text{co-RP}$

Pero se cree que $\text{PTIME} = \text{RP}$

- ▶ Puesto que si $L \in \text{RP}$, entonces hay un algoritmo para resolver L puede ser usado en la *práctica* como un algoritmo de tiempo polinomial
- ▶ De esto se concluiría que $\text{RP} = \text{co-RP} = \text{PTIME}$

$\overline{\text{EQUIV-POL}}$ es un ejemplo de un problema que está en RP y para el cual no se sabe si está en PTIME.

¿Qué sabemos sobre $RP \cap co-RP$?

Tenemos que $PTIME \subseteq RP \cap co-RP$

¿Podemos demostrar que $PTIME = RP \cap co-RP$?

- ▶ Este es un problema abierto

Pero podemos demostrar que para cada problema en $RP \cap co-RP$ existe un algoritmo que lo decide en tiempo polinomial *esperado*.

Un algoritmo de tiempo polinomial esperado

Sea $L \in \text{RP} \cap \text{co-RP}$ con alfabeto Σ

Entonces existen MTs probabilísticas M_1 y M_2 tales que:

- ▶ $t_{M_1}(n)$ es $O(n^k)$ y para cada $w \in \Sigma^*$:
 - ▶ Si $w \in L$, entonces $\Pr(M_1 \text{ acepte } w) \geq \frac{3}{4}$
 - ▶ Si $w \notin L$, entonces $\Pr(M_1 \text{ acepte } w) = 0$
- ▶ $t_{M_2}(n)$ es $O(n^\ell)$ y para cada $w \in \Sigma^*$:
 - ▶ Si $w \in \bar{L}$, entonces $\Pr(M_2 \text{ acepte } w) \geq \frac{3}{4}$
 - ▶ Si $w \notin \bar{L}$, entonces $\Pr(M_2 \text{ acepte } w) = 0$

Un algoritmo de tiempo polinomial esperado

Sea $g(n) = \max\{t_{M_1}(n), t_{M_2}(n)\}$

Considere el siguiente algoritmo para decidir si $w \in L$:

1. Escoja al azar y con distribución uniforme un string $s \in \{0, 1\}^*$ tal que $|s| = g(|w|)$
2. Verifique si $M_1(w, s)$ acepta. Si es así retorne **sí**, sino vaya al paso 3
3. Verifique si $M_2(w, s)$ acepta. Si es así retorne **no**, sino vaya al paso 1

Un algoritmo de tiempo polinomial esperado

El algoritmo anterior puede no detenerse.

- ▶ Si se detiene entrega el resultado correcto

¿Cuál es el tiempo *esperado* de funcionamiento del algoritmo?

- ▶ Calcular el número esperado de veces que se ejecuta la secuencia de pasos 1 al 3 se reduce a calcular la esperanza de una variable aleatoria con distribución geométrica de parámetro $\frac{3}{4}$

Concluimos que el algoritmo funciona en tiempo polinomial esperado.

Una clase de complejidad probabilística más general

Definición

Sea L un lenguaje sobre un alfabeto Σ . Entonces L está en BPP si existe una MT probabilística M tal que $t_M(n)$ es $O(n^k)$ y para cada $w \in \Sigma^$:*

- ▶ *Si $w \in L$, entonces $\mathbf{Pr}(M \text{ acepte } w) \geq \frac{3}{4}$*
- ▶ *Si $w \notin L$, entonces $\mathbf{Pr}(M \text{ acepte } w) \leq \frac{1}{4}$*

¿Dónde está la clase BPP?

Teorema

$$BPP = co-BPP$$

Ejercicio

Demuestre el teorema.

Corolario

$$RP \subseteq BPP \text{ y } co-RP \subseteq BPP$$

¿Dónde está la clase BPP?

Es un problema abierto si $\text{PTIME} = \text{BPP}$

- ▶ De esto se concluiría que $\text{BPP} = \text{RP} = \text{co-RP} = \text{PTIME}$

Vamos a demostrar que BPP está contenida en la jerarquía polinomial.

- ▶ En esta demostración vamos a considerar una versión equivalente pero más simple de la definición de BPP

Simplificando la definición de BPP

Sea M una MT probabilística con alfabeto de entrada Σ

Dado $w \in \Sigma^*$ y $s \in \{0, 1\}^*$ tal que $t_M(|w|) \leq |s|$, decimos que $M(w, s)$ es incorrecto si:

$$\begin{aligned} &w \in L \text{ y } M(w, s) \text{ rechaza} \\ &\quad \text{o} \\ &w \notin L \text{ y } M(w, s) \text{ acepta} \end{aligned}$$

Vamos a utilizar esta noción para dar una definición más simple de BPP

Una definición equivalente de BPP

Definición

Sea L un lenguaje sobre un alfabeto Σ . Entonces L está en BPP si existe una MT probabilística M tal que $t_M(n)$ es $O(n^k)$ y para cada $w \in \Sigma^$:*

$$\Pr_s(M(w, s) \text{ es incorrecto}) \leq \frac{1}{4}$$

Un lema de amplificación para BPP

Al igual que para el caso de RP, el valor $\frac{1}{4}$ en la definición de BPP pueden ser reemplazado por un valor arbitrariamente más pequeño.

Lema de amplificación para BPP

Sea L un lenguaje sobre un alfabeto Σ . Si $L \in \text{BPP}$, entonces para cada $\ell \in \mathbb{N}$, existe una MT probabilística M tal que $t_M(n)$ es $O(n^k)$ y para cada $w \in \Sigma^*$:

$$\Pr_s(M(w, s) \text{ es incorrecto}) \leq \left(\frac{3}{4}\right)^\ell$$

Demostración del lema de amplificación para BPP

Como $L \in \text{BPP}$, existe una MT probabilística M_1 tal que $t_{M_1}(n)$ es $O(n^{k_1})$ y para cada $w \in \Sigma^*$:

$$\Pr_s(M_1(w, s) \text{ es incorrecto}) \leq \frac{1}{4}$$

Utilizamos la MT M_1 para construir la MT M mencionada en el enunciado del lema de amplificación.

Demostración del lema de amplificación para BPP

Considere el siguiente algoritmo que recibe como entrada $w \in \Sigma^*$:

1. Escoja al azar con distribución uniforme y de manera independiente strings $s_1, s_2, \dots, s_{2\ell+1}$ en $\{0, 1\}^*$ y tales que $|s_1| = |s_2| = \dots = |s_{2\ell+1}| = t_{M_1}(|w|)$
2. Construya el conjunto $A = \{i \in \{1, \dots, 2\ell + 1\} \mid M_1(w, s_i) \text{ acepta}\}$
3. Si $|A| \geq \ell + 1$, entonces retorne **sí**, sino retorne **no**

Suponga que M es una MT probabilística que implementa este algoritmo

- Se tiene que $t_M(n)$ es $O(n^k)$ para una constante k ya que $t_{M_1}(n)$ es $O(n^{k_1})$ para una constante k_1

Vamos a demostrar que M satisface la condición del lema

Demostración del lema de amplificación para BPP

Dado $w \in \Sigma^*$, tenemos que:

$$\Pr_s(M_1(w, s) \text{ es incorrecto}) = q$$

con $q \leq \frac{1}{4}$

Además suponemos que $0 < q$

- Si $q = 0$ entonces $\Pr_s(M(w, s) \text{ es incorrecto}) = 0 \leq (\frac{3}{4})^\ell$.
¿Por qué?

Tenemos entonces que:

$$\Pr_s(M(w, s) \text{ es incorrecto}) = \sum_{i=0}^{\ell} \binom{2\ell+1}{i} (1-q)^i q^{2\ell+1-i}$$

Demostración del lema de amplificación para BPP

Como $0 < q \leq \frac{1}{4}$, tenemos que $1 \leq \frac{1-q}{q}$

Entonces para $i \in \{0, \dots, \ell\}$ tenemos que:

$$\begin{aligned} (1-q)^i q^{2\ell+1-i} &\leq (1-q)^i q^{2\ell+1-i} \left(\frac{1-q}{q} \right)^{\ell-i} \\ &= (1-q)^\ell q^{\ell+1} \end{aligned}$$

Demostración del lema de amplificación para BPP

Por lo tanto:

$$\begin{aligned}\sum_{i=0}^{\ell} \binom{2\ell+1}{i} (1-q)^i q^{2\ell+1-i} &\leq \sum_{i=0}^{\ell} \binom{2\ell+1}{i} (1-q)^{\ell} q^{\ell+1} \\ &= (1-q)^{\ell} q^{\ell+1} \sum_{i=0}^{\ell} \binom{2\ell+1}{i} \\ &\leq (1-q)^{\ell} q^{\ell+1} 2^{2\ell+1}\end{aligned}$$

El mayor valor de la función $f(x) = x(1-x)$ en el intervalo $(0, \frac{1}{4}]$ se alcanza en $x = \frac{1}{4}$

► Concluimos que $(1-q)q \leq \frac{3}{4} \cdot \frac{1}{4}$

Demostración del lema de amplificación para BPP

Concluimos que:

$$\begin{aligned}\sum_{i=0}^{\ell} \binom{2\ell+1}{i} (1-q)^i q^{2\ell+1-i} &\leq (1-q)^{\ell} q^{\ell+1} 2^{2\ell+1} \\ &= ((1-q)q)^{\ell} q 2^{2\ell+1} \\ &\leq \left(\frac{3}{4} \cdot \frac{1}{4}\right)^{\ell} \cdot \frac{1}{4} 2^{2\ell+1} \\ &= \frac{3^{\ell}}{4^{2\ell+1}} 2^{2\ell+1} \\ &= 2 \frac{3^{\ell}}{4^{2\ell+1}} 4^{\ell} \\ &= \frac{1}{2} \cdot \frac{3^{\ell}}{4^{\ell}} \\ &< \left(\frac{3}{4}\right)^{\ell}\end{aligned}$$

Demostración del lema de amplificación para BPP

Poniendo todo junto concluimos que:

$$\begin{aligned}\Pr_s(M(w, s) \text{ es incorrecto}) &= \sum_{i=0}^{\ell} \binom{2\ell+1}{i} (1-q)^i q^{2\ell+1-i} \\ &\leq \left(\frac{3}{4}\right)^{\ell}\end{aligned}$$



BPP está en la jerarquía polinomial

Teorema (Gács-Sipser-Lautemann)

$$BPP \subseteq \Sigma_2^P \cap \Pi_2^P$$

Como sabemos que $BPP = \text{co-BPP}$, nos basta demostrar que $BPP \subseteq \Sigma_2^P$

- Antes de realizar esta demostración vamos a ver dos ingredientes necesarios para ella

Una generalización de la noción de lenguaje

Sea Σ un alfabeto.

Un lenguaje L puede tener como elementos pares de strings sobre Σ

- ▶ Tenemos entonces que $L \subseteq \Sigma^* \times \Sigma^*$

L es un lenguaje como los que habíamos definido antes puesto que un par de strings también es un string

- ▶ Podemos representar (x, y) como un string $x\#y$ donde $\#$ es un símbolo nuevo

En las siguientes transparencias vamos a considerar lenguajes que consisten de pares o tuplas de strings.

Una caracterización de NP

Proposición

Sea L un lenguaje sobre un alfabeto Σ . Entonces L está en NP si y sólo si existe un lenguaje $A \subseteq \Sigma^ \times \Sigma^*$ y un polinomio $p(n)$ tales que $A \in PTIME$ y para todo $u \in \Sigma^*$:*

$$u \in L \text{ si y sólo si } (\exists v \in \Sigma^*, |v| = p(|u|)) : (u, v) \in A$$

Ejercicio

Demuestre la proposición.

Primer ingrediente: una caracterización de Σ_2^P

Proposición

Sea L un lenguaje sobre un alfabeto Σ . Entonces L está en Σ_2^P si y sólo si existe un lenguaje $B \subseteq \Sigma^* \times \Sigma^* \times \Sigma^*$ y un polinomio $q(n)$ tales que $B \in PTIME$ y para todo $u \in \Sigma^*$:

$u \in L$ si y sólo si

$$(\exists v_1 \in \Sigma^*, |v_1| = q(|u|))(\forall v_2 \in \Sigma^*, |v_2| = q(|u|)) : (u, v_1, v_2) \in B$$

Ejercicio

Demuestre la dirección ($\Leftarrow$) de la proposición.

- Esta es la dirección que vamos a utilizar para demostrar que $BPP \subseteq \Sigma_2^P$