

# Clases de complejidad aleatorizadas

IIC3810

# Un primer 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 siguientes 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:

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

# 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?

# 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

# 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

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

# 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  $n$

- Puede utilizar como apoyo las transparencias de mi curso Diseño y Análisis de Algoritmos

# 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.

# 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**( $p(x_1, \dots, x_n)$ ,  $q(x_1, \dots, x_n)$ )

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

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

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.

# 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

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

# 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  $t \leq k$

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

# Utilizando el lema de Schwartz-Zippel

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

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

Utilizando el lema de Schwartz-Zippel obtenemos:

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

# Utilizando el lema de Schwartz-Zippel

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

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

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( $p(x_1, \dots, x_n)$ ,  $q(x_1, \dots, x_n)$ )  
   $k := \text{máx} \{m_1, \dots, m_\ell, s_1, \dots, s_r\}$   
   $A := \{1, \dots, 100 \cdot k\}$   
  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?

# 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*
- ▶  *$\Sigma$  es un alfabeto finito tal que  $\vdash, \sqcup \notin \Sigma$*
- ▶  *$\Gamma$  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*
- ▶  *$\delta$  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  $\vdash wB \dots$  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  $p_1$

# 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  $\Sigma$  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

# El tiempo de ejecución de una MT probabilística

La entrada de una MT probabilística  $M$  con alfabeto  $\Sigma$  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

# 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  $(w, s)$

# 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  $s \in \{0, 1\}^\omega$ :

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

# 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:

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

# 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  $\Sigma$  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  $\Sigma$  y tal que  $t_M(n) \leq g(n)$  para todo  $n \in \mathbb{N}$ .

# La probabilidad de aceptar en una MT probabilística

Sea  $M$  una MT probabilística con alfabeto  $\Sigma$  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:*

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

# 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

# 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  $\Sigma$ . 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  $\Pr(M \text{ acepte } w) = 0$

# 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  $\Sigma$ . 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  $\overline{\text{EQUIV-POL}} \in \text{RP}$