Teoría de la Computación

Juan Gutiérrez

March 27, 2023

Resumen

Información útil

El profesor

- Juan Gutiérrez

El profesor

- Juan Gutiérrez
- Dr. Ciencia de la Computación, Instituto de Matemática y Estadística (IME), Universidad de Sao Paulo (USP)

El profesor

- Juan Gutiérrez
- Dr. Ciencia de la Computación, Instituto de Matemática y Estadística (IME), Universidad de Sao Paulo (USP)
- Área: Computación teórica. Grupo: Teoria da Computação, Combinatória e Otimização

El profesor

- Juan Gutiérrez
- Dr. Ciencia de la Computación, Instituto de Matemática y Estadística (IME), Universidad de Sao Paulo (USP)
- Área: Computación teórica. Grupo: Teoria da Computação, Combinatória e Otimização
- Subáreas: Teoría de grafos, Matemática discreta, Análisis de algoritmos, Optimización combinatoria.

El profesor

- Juan Gutiérrez
- Dr. Ciencia de la Computación, Instituto de Matemática y Estadística (IME), Universidad de Sao Paulo (USP)
- Área: Computación teórica. Grupo: Teoria da Computação, Combinatória e Otimização
- Subáreas: Teoría de grafos, Matemática discreta, Análisis de algoritmos, Optimización combinatoria.
- Tesis: Transversals of graphs

Sistema de evaluación

- $N_F = 0.25 * E_1 + 0.25 * E_2 + 0.25 * E_3 + 0.25 * C$
- E_i: Exámenes (3)
- C: Continua (3)
- Para aprobar el curso hay que obtener 11 o más en la nota final N_F.

Bibliografía

- **Sipser**: Introducción a la teoria de la computación
- **Hopcroft**: Introducción a la teoria de autómatas lenguajes y computación

Teoría de la computacion

- En el sentido amplio: https://en.wikipedia.org/wiki/Theoretical_computer_science (Computación teórica)
- Más estrictamente: https://en.wikipedia.org/wiki/Theory_of_computation

Objetivos

La pregunta principal

¿Cuales son las capacidades y limitaciones fundamentales de las computadoras?

La pregunta principal

3 tipos de respuestas dependiendo del enfoque

- Autómatas

La pregunta principal

3 tipos de respuestas dependiendo del enfoque

- Autómatas
- Computabilidad

La pregunta principal

3 tipos de respuestas dependiendo del enfoque

- Autómatas
- Computabilidad
- Complejidad computacional

La pregunta principal

Complejidad computacional

- Problemas fáciles vs problemas dificiles

La pregunta principal

Complejidad computacional

- Problemas fáciles vs problemas dificiles
- Problema fácil: Ordenar elementos de un vector

La pregunta principal

Complejidad computacional

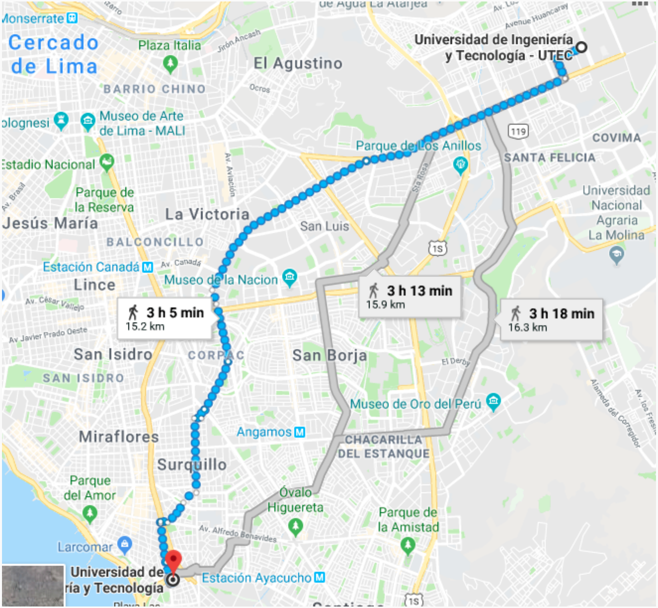
- Problemas fáciles vs problemas dificiles
- Problema fácil: Ordenar elementos de un vector
- Problema dificil: Asignar horarios en una universidad

La pregunta principal

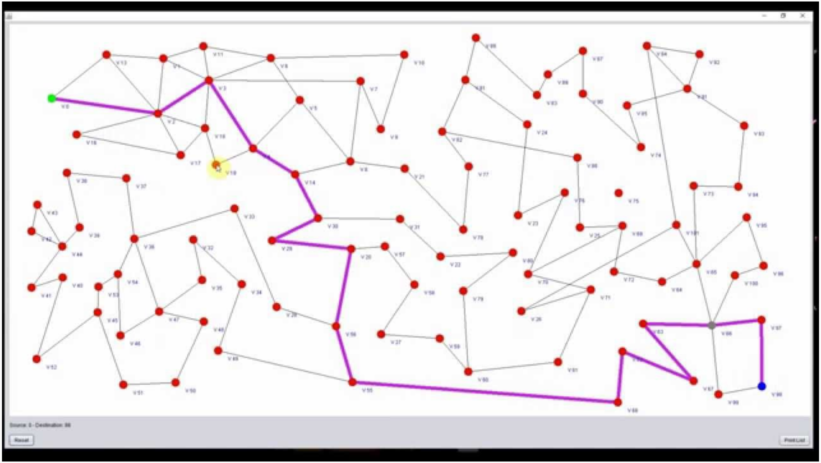
Complejidad computacional

- Problemas fáciles vs problemas dificiles
- Problema fácil: Ordenar elementos de un vector
- Problema dificil: Asignar horarios en una universidad
- Pregunta reformulada: ¿Qué hace que algunos problemas sean fáciles y otros difíciles?

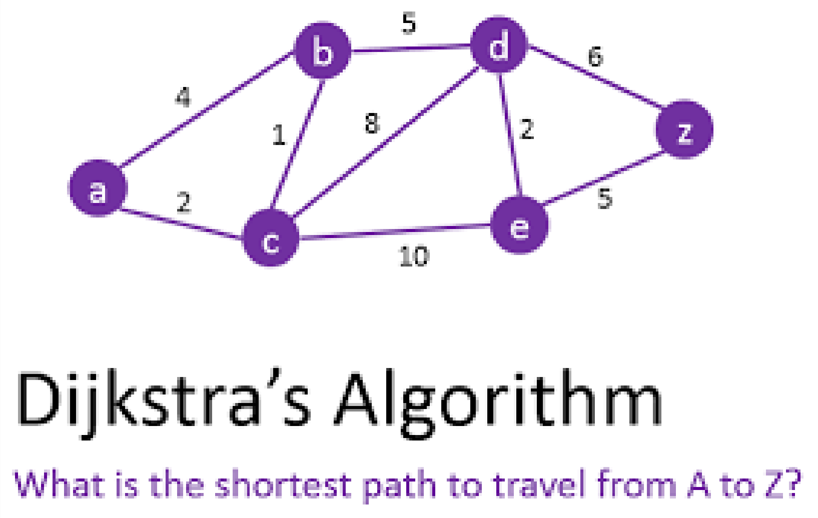
Más problemas ...



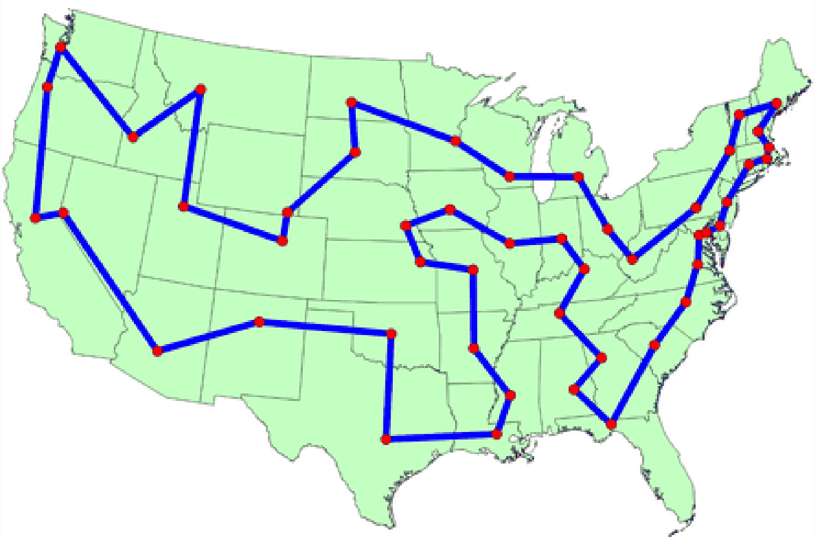
Más problemas ...



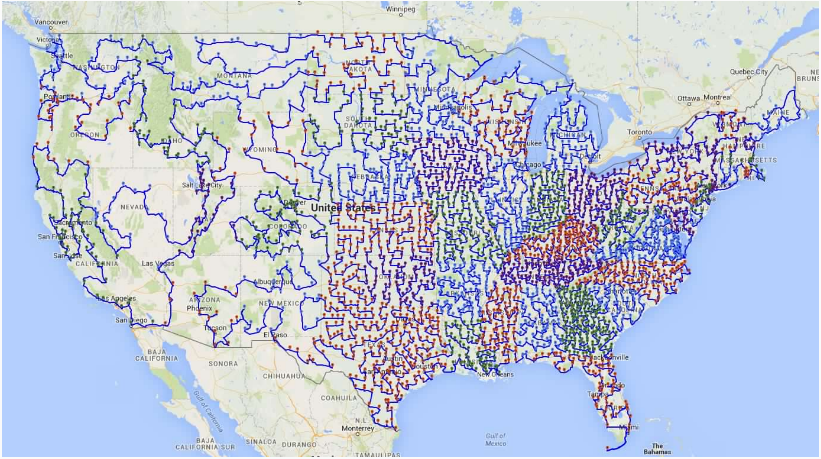
Más problemas ...



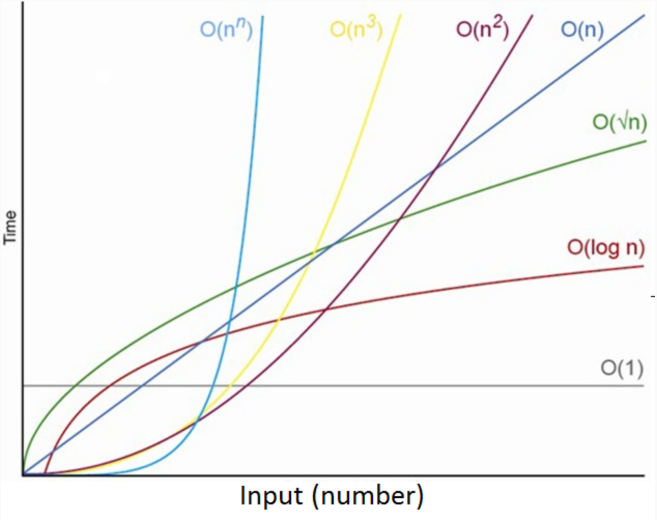
Más problemas ...



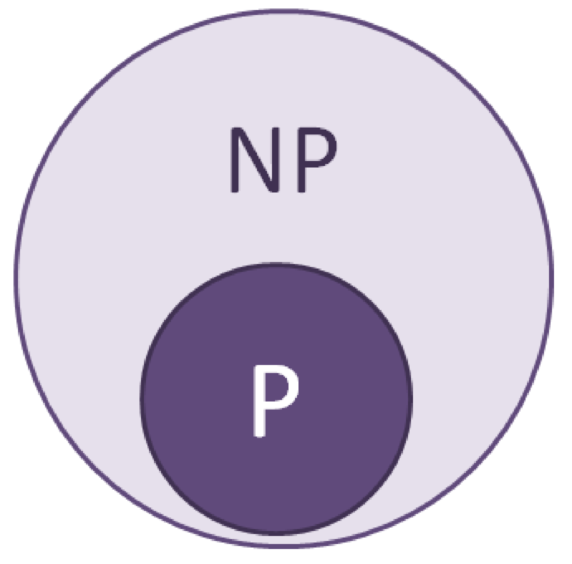
Más problemas ...



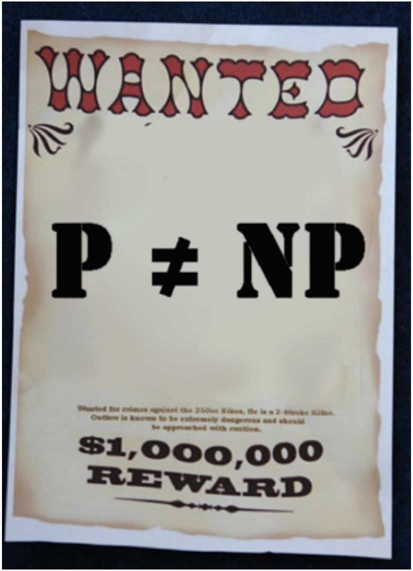
P vs NP



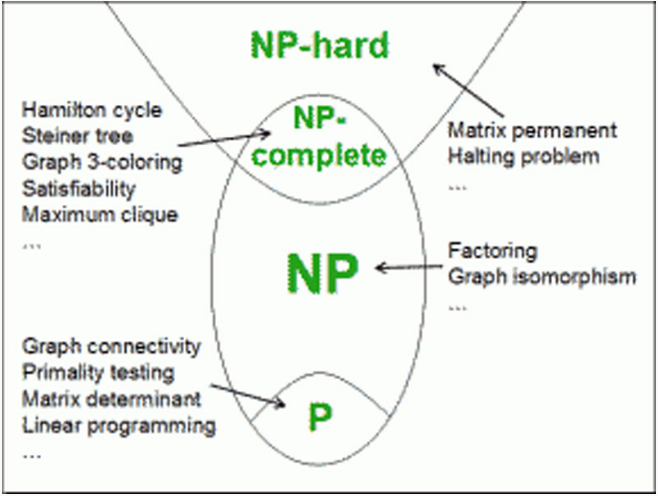
P vs NP



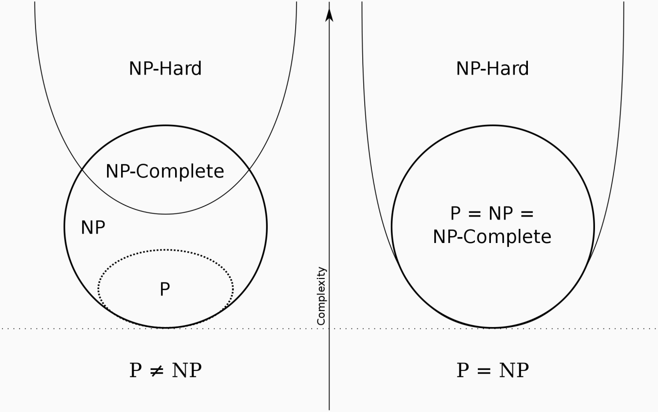
P vs NP



P vs NP


NP-hard
NP-complete
NP
P
Hamilton cycle
Steiner tree
Graph 3-coloring
Satisfiability
Maximum clique
...
Matrix permanent
Halting problem
...
Factoring
Graph isomorphism
...
Graph connectivity
Primality testing
Matrix determinant
Linear programming
...

P vs NP



NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP =
NP-Complete
P ≠ NP
P = NP

La pregunta principal

Computabilidad

- Problemas que se pueden resolver vs problemas que no se pueden resolver (mediante el computador)

La pregunta principal

Computabilidad

- Problemas que se pueden resolver vs problemas que no se pueden resolver (mediante el computador)
- Problema se puede resolver: Varios

La pregunta principal

Computabilidad

- Problemas que se pueden resolver vs problemas que no se pueden resolver (mediante el computador)
- Problema se puede resolver: Varios
- Problema que no se puede resolver: Determinar si una expresión matemática es verdadera o falsa

La pregunta principal

Computabilidad

- Problemas que se pueden resolver vs problemas que no se pueden resolver (mediante el computador)
- Problema se puede resolver: Varios
- Problema que no se puede resolver: Determinar si una expresión matemática es verdadera o falsa
- Problema que no se puede resolver: Determinar si un programa termina al recibir cierta entrada

La pregunta principal

Computabilidad

- Problemas que se pueden resolver vs problemas que no se pueden resolver (mediante el computador)
- Problema se puede resolver: Varios
- Problema que no se puede resolver: Determinar si una expresión matemática es verdadera o falsa
- Problema que no se puede resolver: Determinar si un programa termina al recibir cierta entrada
- ▸ Problema de la parada

La pregunta principal

Teoría de Autómatas

- Ofrece diversos modelos de computación

La pregunta principal

Teoría de Autómatas

- Ofrece diversos modelos de computación
- Autómatas finitos

La pregunta principal

Teoría de Autómatas

- Ofrece diversos modelos de computación
- Autómatas finitos
- Autómatas de pila

La pregunta principal

Teoría de Autómatas

- Ofrece diversos modelos de computación
- Autómatas finitos
- Autómatas de pila
- Máquinas de turing

La pregunta principal

Teoría de Autómatas

- Ofrece diversos modelos de computación
- Autómatas finitos
- Autómatas de pila
- Máquinas de turing
- Pregunta reformulada: ¿Todos estos modelos tienen el mismo poder? O algunos modelos resuelven más problemas que otros.

Para poder dar respuestas a la pregunta inicial...

- Debemos formalizar el concepto de computador

Para poder dar respuestas a la pregunta inicial...

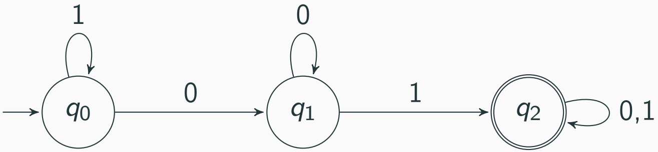
- Debemos formalizar el concepto de computador
- La teoría de autómatas ayuda a esta formalización

Para poder dar respuestas a la pregunta inicial...

- Debemos formalizar el concepto de computador
- La teoría de autómatas ayuda a esta formalización
- Comenzaremos estudiando autómatas!

Autómatas

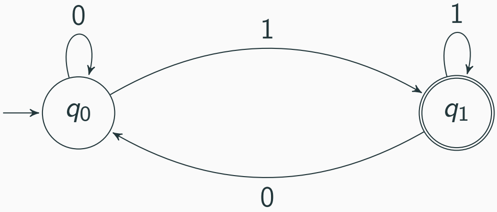
Un autómata



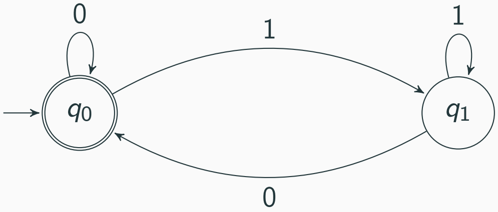
Otro autómata

¿Qué cadenas reconoce el autómata?

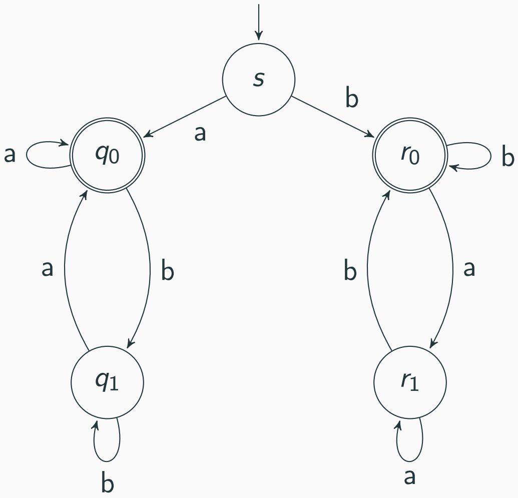
Más autómatas

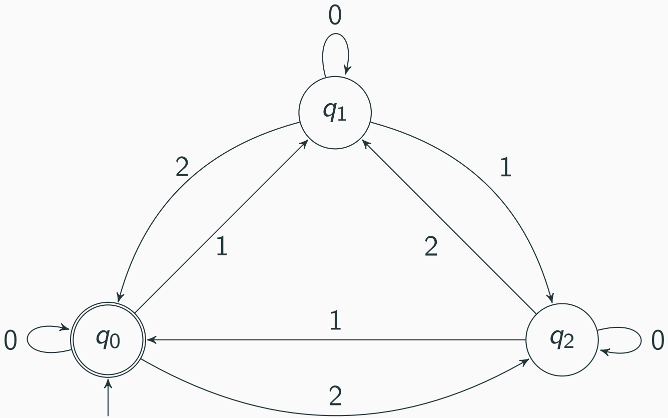


Más autómatas



Más autómatas




0
q_1
2
1
1
2
1
0
q_0
q_2
0
2

Más autómatas

¿Cómo generalizar el autómata anterior?

Definición formal de autómata

Un autómata finito es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1. Q es un conjunto finito de elementos llamados *estados*,

Definición formal de autómata

Un autómata finito es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1 Q es un conjunto finito de elementos llamados *estados*,

2 $\sum$ es un conjunto finito llamado *alfabeto*,

Definición formal de autómata

Un autómata finito es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1 Q es un conjunto finito de elementos llamados *estados*,

2 $\sum$ es un conjunto finito llamado *alfabeto*,

3 $\delta : Q \times \Sigma \rightarrow Q$ es la *función de transición*,

Definición formal de autómata

Un autómata finito es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1 Q es un conjunto finito de elementos llamados *estados*,

2 $\sum$ es un conjunto finito llamado *alfabeto*,

3 $\delta : Q \times \Sigma \to Q$ es la *función de transición*,

4 $q_0 \in Q$ es el *estado inicial*, y

Definición formal de autómata

Un autómata finito es una 5-tupla $(Q, \sum, \delta, q_0, F)$, donde

1 Q es un conjunto finito de elementos llamados *estados*,

2 $\sum$ es un conjunto finito llamado *alfabeto*,

3 $\delta : Q \times \Sigma \to Q$ es la *función de transición*,

4 $q_0 \in Q$ es el *estado inicial*, y

5 $F \subseteq Q$ es el conjunto de *estados de aceptación*,

Definición formal de computación

Sea $A = (Q, \sum, \delta, q_0, F)$ un autómata finito. Sea $w = w_1 w_2 \ldots w_n$ una cadena, donde cada w_i está en el alfabeto $\sum$. Entonces A acepta w si existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 = q_0$,

Definición formal de computación

Sea $A = (Q, \sum, \delta, q_0, F)$ un autómata finito. Sea $w = w_1 w_2 \ldots w_n$ una cadena, donde cada w_i está en el alfabeto $\sum$. Entonces A acepta w si existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, y

Definición formal de computación

Sea $A = (Q, \sum, \delta, q_0, F)$ un autómata finito. Sea $w = w_1 w_2 \ldots w_n$ una cadena, donde cada w_i está en el alfabeto $\sum$. Entonces A acepta w si existe una secuencia de estados $r_0, r_1 \ldots r_n$ en Q con las siguientes condiciones:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, para $i = 0, \ldots, n-1$, y
3. $r_n \in F$

Definición formal de computación

Decimos que el autómata A reconoce el lenguaje L si
$L = \{w : A \text{ acepta } w\}$.

Dicho lenguaje suele ser denotado por $L(A)$.

Definición formal de computación

Un lenguaje es llamado ***regular*** *si existe algún autómata finito que lo reconoce*

Definición formal de computación

Pero qué es exactamente un lenguaje?

Algunas definiciones básicas

- **Alfabeto:** conjunto finito y no vacío de elementos llamados **símbolos**

Algunas definiciones básicas

- **Alfabeto:** conjunto finito y no vacío de elementos llamados **símbolos**
- $\sum_1 = \{0, 1\}$

Algunas definiciones básicas

- **Alfabeto:** conjunto finito y no vacío de elementos llamados **símbolos**
- $\sum_1 = \{0, 1\}$
- $\sum_2 = \{\text{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}\}$

Algunas definiciones básicas

- **Alfabeto:** conjunto finito y no vacío de elementos llamados **símbolos**
- $\sum_1 = \{0, 1\}$
- $\sum_2 = \{$a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z$\}$
- $\Gamma = \{0, 1, x, y, z\}$

Algunas definiciones básicas

- **Cadena** sobre un alfabeto: secuencia finita de **símbolos** del alfabeto

Algunas definiciones básicas

- **Cadena** sobre un alfabeto: secuencia finita de **símbolos** del alfabeto
- 01011 es una cadena sobre $\sum_1 = \{0, 1\}$

Algunas definiciones básicas

- **Cadena** sobre un alfabeto: secuencia finita de **símbolos** del alfabeto
- 01011 es una cadena sobre $\sum_1 = \{0, 1\}$
- abracadabra es una cadena sobre $\sum_2 = \{$a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z$\}$

Algunas definiciones básicas

- **Cadena** sobre un alfabeto: secuencia finita de **símbolos** del alfabeto
- 01011 es una cadena sobre $\sum_1 = \{0, 1\}$
- abracadabra es una cadena sobre $\sum_2 = \{\text{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}\}$
- $|w|$: tamaño de una cadena w

Algunas definiciones básicas

- **Cadena** sobre un alfabeto: secuencia finita de **símbolos** del alfabeto
- 01011 es una cadena sobre $\sum_1 = \{0, 1\}$
- abracadabra es una cadena sobre $\sum_2 = \{$a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z$\}$
- $|w|$: tamaño de una cadena w
- ϵ: cadena vacía (tamaño 0)

Algunas definiciones básicas

- $\sum^k$: conjunto de cadenas de longitud k sobre el alfabeto $\sum$

Algunas definiciones básicas

- $\sum^{k}$: conjunto de cadenas de longitud k sobre el alfabeto $\sum$
- $\sum = \{0, 1\}$

Algunas definiciones básicas

- $\sum^k$: conjunto de cadenas de longitud k sobre el alfabeto $\sum$
- $\sum = \{0, 1\}$
- $\sum^0 = \{\epsilon\}$

Algunas definiciones básicas

- $\sum^k$: conjunto de cadenas de longitud k sobre el alfabeto $\sum$
- $\sum = \{0, 1\}$
- $\sum^0 = \{\epsilon\}$
- $\sum^1 = \{0, 1\}$

Algunas definiciones básicas

- $\sum^k$: conjunto de cadenas de longitud k sobre el alfabeto $\sum$
- $\sum = \{0, 1\}$
- $\sum^0 = \{\epsilon\}$
- $\sum^1 = \{0, 1\}$
- $\sum^2 = \{00, 01, 10, 11\}$

Algunas definiciones básicas

- $\sum^*$: conjunto de todas las cadenas sobre el alfabeto $\sum$

Algunas definiciones básicas

- $\sum^*$: conjunto de todas las cadenas sobre el alfabeto $\sum$
- $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

Algunas definiciones básicas

- $\sum^*$: conjunto de todas las cadenas sobre el alfabeto $\sum$
- $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
- $\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \ldots$

Algunas definiciones básicas

- $\sum^*$: conjunto de todas las cadenas sobre el alfabeto $\sum$
- $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
- $\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \ldots$
- $\sum^+$: conjunto de todas las cadenas no vacías sobre el alfabeto $\sum$

Algunas definiciones básicas

- $\sum^*$: conjunto de todas las cadenas sobre el alfabeto $\sum$
- $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
- $\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \ldots$
- $\sum^+$: conjunto de todas las cadenas no vacías sobre el alfabeto $\sum$
- $\sum^+ = \sum^1 \cup \sum^2 \cup \ldots$

Algunas definiciones básicas

- $\sum^*$: conjunto de todas las cadenas sobre el alfabeto $\sum$
- $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
- $\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \ldots$
- $\sum^+$: conjunto de todas las cadenas no vacías sobre el alfabeto $\sum$
- $\sum^+ = \sum^1 \cup \sum^2 \cup \ldots$
- $\sum^* = \sum^+ \cup \{\epsilon\}$

Algunas definiciones básicas

- cadenas x e y

Algunas definiciones básicas

- cadenas x e y
- xy: **concatenación** de x e y

Algunas definiciones básicas

- cadenas x e y
- xy: **concatenación** de x e y
- $x = 01011$, $y = 1110$

Algunas definiciones básicas

- cadenas x e y
- xy: **concatenación** de x e y
- $x = 01011$, $y = 1110$
- $xy = 010111110$

Algunas definiciones básicas

- cadenas x e y
- xy: **concatenación** de x e y
- $x = 01011$, $y = 1110$
- $xy = 010111110$
- Para cualquier cadena w, $\epsilon w = w \epsilon = w$

Algunas definiciones básicas

- **Lenguaje** sobre $\sum$: subconjunto de $\sum^*$

Algunas definiciones básicas

- **Lenguaje** sobre $\sum$: subconjunto de $\sum^*$
- $\sum^*$ es un lenguaje sobre $\sum$

Algunas definiciones básicas

- **Lenguaje** sobre $\sum$: subconjunto de $\sum^*$
- $\sum^*$ es un lenguaje sobre $\sum$
- $\emptyset$ es un lenguaje sobre $\sum$

Algunas definiciones básicas

- **Lenguaje** sobre $\sum$: subconjunto de $\sum^*$
- $\sum^*$ es un lenguaje sobre $\sum$
- $\emptyset$ es un lenguaje sobre $\sum$
- $\{\epsilon\}$ es un lenguaje sobre $\sum$
- $\emptyset \neq \{\epsilon\}$

Algunas definiciones básicas

- El lenguaje de todas las cadenas de 0's y 1's que consisten en n 0's seguidos de n 1's, para algún $n \geq 0$

Algunas definiciones básicas

- El lenguaje de todas las cadenas de 0's y 1's que consisten en n 0's seguidos de n 1's, para algún $n \geq 0$
- $\{\epsilon, 01, 0011, 000111, \ldots\}$

Algunas definiciones básicas

- El lenguaje de todas las cadenas de 0's y 1's que consisten en n 0's seguidos de n 1's, para algún $n \geq 0$
- $\{\epsilon, 01, 0011, 000111, \ldots\}$
- El lenguaje de todas las cadenas de 0's y 1's que tienen la misma cantidad de 0's y 1's

Algunas definiciones básicas

- El lenguaje de todas las cadenas de 0's y 1's que consisten en n 0's seguidos de n 1's, para algún $n \geq 0$
- $\{\epsilon, 01, 0011, 000111, \ldots\}$
- El lenguaje de todas las cadenas de 0's y 1's que tienen la misma cantidad de 0's y 1's
- $\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$

Algunas definiciones básicas

- El lenguaje de todas las cadenas de 0's y 1's que consisten en n 0's seguidos de n 1's, para algún $n \geq 0$
- $\{\epsilon, 01, 0011, 000111, \ldots\}$
- El lenguaje de todas las cadenas de 0's y 1's que tienen la misma cantidad de 0's y 1's
- $\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$
- El lenguaje de todas las cadenas de 0's y 1's que representan números primos

Algunas definiciones básicas

- El lenguaje de todas las cadenas de 0's y 1's que consisten en n 0's seguidos de n 1's, para algún $n \geq 0$
- $\{\epsilon, 01, 0011, 000111, \ldots\}$
- El lenguaje de todas las cadenas de 0's y 1's que tienen la misma cantidad de 0's y 1's
- $\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$
- El lenguaje de todas las cadenas de 0's y 1's que representan números primos
- $\{10, 11, 101, 1011, \ldots\}$

Actividad en clase

Diseñe autómatas finitos para los siguientes lenguajes. El alfabeto es siempre $\{0, 1\}$

(a) Cadenas con un número impar de 1's
(b) Cadenas con un número par de 1's
(c) Cadenas con un número de unos múltiplo de 3
(d) Cadenas que tienen un número par de ceros siguiendo al último 1, y que tienen por lo menos un 1
(e) Cadenas que tienen al 001 como subcadena
(f) Cadenas que tienen al 011 como subcadena
(g) Cadenas con tres ceros consecutivos
(h) Cadenas cuyo segundo dígito empezando desde la derecha es 1

Diseñar autómatas para los siguientes lenguajes

Cadenas con un número impar de 1's

Diseñar autómatas para los siguientes lenguajes

Cadenas con un número par de 1's

Diseñar autómatas para los siguientes lenguajes

Cadenas con un número de unos múltiplo de 3

Diseñar autómatas para los siguientes lenguajes

Cadenas que tienen un número par de ceros siguiendo al último 1, y que tienen por lo menos un 1

Diseñar autómatas para los siguientes lenguajes

Cadenas que tienen al 001 como subcadena

Diseñar autómatas para los siguientes lenguajes

Cadenas que tienen al 011 como subcadena

Diseñar autómatas para los siguientes lenguajes

Cadenas con tres ceros consecutivos

Diseñar autómatas para los siguientes lenguajes

Cadenas cuyo segundo dígito empezando desde la derecha es 1

Actividad en clase

Diseñe autómatas finitos para los siguientes lenguajes. El alfabeto es siempre $\{0, 1\}$

(a) Cadenas que comienzan en 1 y terminan en 0

(b) Cadenas que tienen al 0101 como subcadena

(c) Todas las cadenas excepto la cadena vacía

(d) Cadenas cuyo tercer dígito empezando desde la derecha es 1

(e) $\{\epsilon, 0\}$

(f) El conjunto vacío

(g) Cadenas cuya longitud es máximo 5

(h) Cadenas con un número impar de 1's y un número impar de 0's

Otro autómata

Cadenas con exactamente dos a's y con por lo menos dos b's

Otro autómata

Como proceder cuando es pedida la unión de dos lenguajes?

Operaciones regulares

- Sean A y B dos lenguajes

Operaciones regulares

- Sean A y B dos lenguajes
- **Unión**: $A \cup B = \{x : x \in A \text{ o } x \in B\}$

Operaciones regulares

- Sean A y B dos lenguajes
- **Unión**: $A \cup B = \{x : x \in A \text{ o } x \in B\}$
- **Concatenación**: $A \cdot B = \{xy : x \in A \text{ , } y \in B\}$

Operaciones regulares

- Sean A y B dos lenguajes
- **Unión**: $A \cup B = \{x : x \in A \text{ o } x \in B\}$
- **Concatenación**: $A \cdot B = \{xy : x \in A \ , \ y \in B\}$
- **Estrella**: $A^* = \{x_1 x_2 \ldots x_k : x_i \in A\}$

Operaciones regulares

- $A = \{good, bad\}$ y $B = \{boy, girl\}$

Operaciones regulares

- $A = \{good, bad\}$ y $B = \{boy, girl\}$
- $A \cup B = \{good, bad, boy, girl\}$

Operaciones regulares

- $A = \{good, bad\}$ y $B = \{boy, girl\}$
- $A \cup B = \{good, bad, boy, girl\}$
- $A \cdot B = \{goodboy, goodgirl, badboy, badgirl\}$

Operaciones regulares

- $A = \{good, bad\}$ y $B = \{boy, girl\}$
- $A \cup B = \{good, bad, boy, girl\}$
- $A \cdot B = \{goodboy, goodgirl, badboy, badgirl\}$
- $A^* = \{\epsilon, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, \ldots\}$

Operaciones regulares

- $\mathbb{N} = \{1, 2, 3, \ldots\}$

Operaciones regulares

- $\mathbb{N} = \{1, 2, 3, \ldots\}$
- $\mathbb{N}$ es **cerrado** bajo la operación de multiplicación, osea

Operaciones regulares

- $\mathbb{N} = \{1, 2, 3, \ldots\}$
- $\mathbb{N}$ es **cerrado** bajo la operación de multiplicación, osea
- si $x, y \in \mathbb{N}$, entonces $x \times y \in \mathbb{N}$

Operaciones regulares

- $\mathbb{N} = \{1, 2, 3, \ldots\}$
- $\mathbb{N}$ es **cerrado** bajo la operación de multiplicación, osea
- si $x, y \in \mathbb{N}$, entonces $x \times y \in \mathbb{N}$
- $\mathbb{N}$ NO es cerrado bajo la operación de división, ya que

Operaciones regulares

- $\mathbb{N} = \{1, 2, 3, \ldots\}$
- $\mathbb{N}$ es **cerrado** bajo la operación de multiplicación, osea
- si $x, y \in \mathbb{N}$, entonces $x \times y \in \mathbb{N}$
- $\mathbb{N}$ NO es cerrado bajo la operación de división, ya que
- $1, 2 \in \mathbb{N}$, pero $1/2 \notin \mathbb{N}$

Operaciones regulares

La clase de lenguajes regulares es cerrada bajo la operación de unión? Y bajo las operaciónes de concatenación y cerradura?

Operaciones regulares

Teorema
La clase de lenguajes regulares es cerrada bajo la operación de unión. Es decir, si L_1 *y* L_2 *son lenguajes regulares, entonces* $L_1 \cup L_2$ *es también un lenguaje regular.*

Operaciones regulares

- Prueba por construcción

Operaciones regulares

- Prueba por construcción
- Como L_1 es regular, existe un AFD A_1 que reconoce L_1

Operaciones regulares

- Prueba por construcción
- Como L_1 es regular, existe un AFD A_1 que reconoce L_1
- Como L_2 es regular, existe un AFD A_2 que reconoce L_2

Operaciones regulares

- Prueba por construcción
- Como L_1 es regular, existe un AFD A_1 que reconoce L_1
- Como L_2 es regular, existe un AFD A_2 que reconoce L_2
- Construimos A a partir de A_1 y A_2

Operaciones regulares

- Prueba por construcción
- Como L_1 es regular, existe un AFD A_1 que reconoce L_1
- Como L_2 es regular, existe un AFD A_2 que reconoce L_2
- Construimos A a partir de A_1 y A_2
- No podemos simular primero A_1 y luego, si es que no funciona, simular A_2

Operaciones regulares

- Prueba por construcción
- Como L_1 es regular, existe un AFD A_1 que reconoce L_1
- Como L_2 es regular, existe un AFD A_2 que reconoce L_2
- Construimos A a partir de A_1 y A_2
- No podemos simular primero A_1 y luego, si es que no funciona, simular A_2
- Debemos simular A_1 y A_2 simultáneamente

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$
- Sea $A_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$
- Sea $A_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$
- Construímos $A = \{Q, \Sigma, \delta, q, F\}$ como

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$
- Sea $A_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$
- Construímos $A = \{Q, \Sigma, \delta, q, F\}$ como
- $Q = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$
- Sea $A_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$
- Construímos $A = \{Q, \Sigma, \delta, q, F\}$ como
- $Q = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$
- $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$
- Sea $A_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$
- Construímos $A = \{Q, \Sigma, \delta, q, F\}$ como
- $Q = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$
- $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $q_0 = (q_1, q_2)$

Operaciones regulares

- Sea $A_1 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$
- Sea $A_2 = \{Q_2, \Sigma, \delta_2, q_2, F_2\}$
- Construímos $A = \{Q, \Sigma, \delta, q, F\}$ como
- $Q = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\}$
- $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $q_0 = (q_1, q_2)$
- $F = \{(r_1, r_2) : r_1 \in F_1 \text{ o } r_2 \in F_2)\}$

Operaciones regulares

Y la concatenación, es cerrada?

Teorema
La clase de lenguajes regulares es cerrada bajo la operación de concatenación. Es decir, si L_1 *y* L_2 *son lenguajes regulares, entonces* $L_1 \cdot L_2$ *es también un lenguaje regular.*

Operaciones regulares

Como probar que la concatenación es cerrada?

Operaciones regulares

- Como antes, intentaremos construir un autómata A a partir de A_1 y A_2

Operaciones regulares

- Como antes, intentaremos construir un autómata A a partir de A_1 y A_2
- A acepta si su entrada puede ser dividida en dos partes, la primera aceptada por A_1, y la segunda aceptada por A_2

Operaciones regulares

- Como antes, intentaremos construir un autómata A a partir de A_1 y A_2
- A acepta si su entrada puede ser dividida en dos partes, la primera aceptada por A_1, y la segunda aceptada por A_2
- El problema es que A no sabe en qué momento dividir su entrada

Operaciones regulares

- Como antes, intentaremos construir un autómata A a partir de A_1 y A_2
- A acepta si su entrada puede ser dividida en dos partes, la primera aceptada por A_1, y la segunda aceptada por A_2
- El problema es que A no sabe en qué momento dividir su entrada
- Necesitamos un modelo con más posibilidades: **NO DETERMINISMO**

Gracias