Introducción a la Complejidad Computacional

IIC2213

Complejidad Computacional

Objetivo: Medir la complejidad computacional de un problema.

Vale decir: Medir la cantidad de recursos computacionales necesarios para solucionar un problema.

- Tiempo
- Espacio
- ...

Para hacer esto primero tenemos que introducir la noción de problema.

Problemas de decisión

Alfabeto A: Conjunto finito de símbolos.

- Ejemplo: $A = \{0, 1\}$

Problemas de decisión

Alfabeto A: Conjunto finito de símbolos.

- Ejemplo: $A = \{0, 1\}$

Palabra w: Secuencia finita de símbolos de A.

- Ejemplo: $w = 01101$

Problemas de decisión

Alfabeto A: Conjunto finito de símbolos.

- Ejemplo: $A = \{0, 1\}$

Palabra w: Secuencia finita de símbolos de A.

- Ejemplo: $w = 01101$

A^*: Conjunto de todas las palabras construidas con símbolos de A.

Problemas de decisión

Alfabeto A: Conjunto finito de símbolos.

- Ejemplo: $A = \{0, 1\}$

Palabra w: Secuencia finita de símbolos de A.

- Ejemplo: $w = 01101$

A^*: Conjunto de todas las palabras construidas con símbolos de A.

Lenguaje L: Conjunto de palabras.

- Ejemplo: $L = \{0^n 1^n \mid n \in \mathbb{N}\}$

Problemas de decisión

Problema de decisión asociado a un lenguaje L: Dado $w \in A^*$, decidir si $w \in L$

Problemas de decisión

Problema de decisión asociado a un lenguaje L: Dado $w \in A^*$, decidir si $w \in L$

Ejemplo

Podemos ver SAT como un problema de decisión. Suponga que $P = \{p, q\}$:

- $A = \{p, q, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$
  Algunas palabras de A^* representan fórmulas de $L(P)$, mientras que otras tales como $\neg\neg$ y $p\neg q \wedge \wedge \vee q$ no representan fórmulas
- SAT $= \{w \in A^* \mid w$ representa una fórmula de $L(P)$ y w es satisfacible$\}$

Complejidad de un problema de decisión

La complejidad de un lenguaje L es la complejidad del problema de decisión asociado a L.

¿Cuándo decimos que L puede ser solucionado eficientemente?

- Cuando existe un algoritmo eficiente que decide L

Complejidad de un problema de decisión

La complejidad de un lenguaje L es la complejidad del problema de decisión asociado a L.

¿Cuándo decimos que L puede ser solucionado eficientemente?

- Cuando existe un algoritmo eficiente que decide L

Ejercicio

Muestre que $L = \{w \in \{0,1\}^* \mid \text{número de 0s y 1s en } w \text{ es el mismo}\}$ puede ser resuelto eficientemente.

Complejidad de un problema de decisión

La complejidad de un lenguaje L es la complejidad del problema de decisión asociado a L.

¿Cuándo decimos que L puede ser solucionado eficientemente?

- Cuando existe un algoritmo eficiente que decide L

Ejercicio

Muestre que $L = \{w \in \{0,1\}^* \mid \text{número de 0s y 1s en } w \text{ es el mismo}\}$ puede ser resuelto eficientemente.

¿Cuándo decimos que L es un problema difícil?

- Cuando **no existe** un algoritmo eficiente que decide L

Máquinas de Turing

¿Cómo podemos demostrar que un problema es difícil?

- Para hacer esto, primero tenemos que formalizar la noción de algoritmo

Máquinas de Turing

¿Cómo podemos demostrar que un problema es difícil?

- Para hacer esto, primero tenemos que formalizar la noción de algoritmo

¿Qué es un algoritmo? ¿Podemos formalizar este concepto?

- Máquinas de Turing: Intento por formalizar este concepto

Máquinas de Turing

¿Cómo podemos demostrar que un problema es difícil?

- Para hacer esto, primero tenemos que formalizar la noción de algoritmo

¿Qué es un algoritmo? ¿Podemos formalizar este concepto?

- Máquinas de Turing: Intento por formalizar este concepto

¿Podemos demostrar que las máquinas de Turing capturan la noción de algoritmo?

- No, el concepto de algoritmo es intuitivo

Máquinas de Turing

¿Por qué creemos que las máquinas de Turing son una buena formalización del concepto de algoritmo?

- Porque cada programa de una máquina de Turing puede ser implementado
- Porque todos los algoritmos conocidos han podido ser implementados en máquinas de Turing
- Porque todos los otros intentos por formalizar este concepto fueron reducidos a las máquinas de Turing
  - Los mejores intentos resultaron ser equivalentes a las máquinas de Turing
  - Todos los intentos "razonables" fueron reducidos eficientemente

Máquinas de Turing

¿Por qué creemos que las máquinas de Turing son una buena formalización del concepto de algoritmo?

- Porque cada programa de una máquina de Turing puede ser implementado
- Porque todos los algoritmos conocidos han podido ser implementados en máquinas de Turing
- Porque todos los otros intentos por formalizar este concepto fueron reducidos a las máquinas de Turing
  - Los mejores intentos resultaron ser equivalentes a las máquinas de Turing
  - Todos los intentos "razonables" fueron reducidos eficientemente
- Tesis de Church: **Algoritmo = Máquina de Turing**

Máquinas de Turing: Componentes

Dado: Alfabeto A que no contiene símbolo especial $\mathtt{B}$.

Componentes:

- Memoria: Arreglo infinito (en ambas direcciones) que en cada posición almacena un símbolo de $A \cup \{\mathtt{B}\}$
- Control: Conjunto finitos de estados, una cabeza lectora, un estado inicial y un conjunto de estados finales
- Programa: Conjunto de instrucciones

Máquinas de Turing: Funcionamiento

Inicialmente:

- La máquina recibe como entrada una palabra w
- Coloca esta palabra en alguna parte del arreglo. En las posiciones restantes el arreglo contiene símbolo blanco `B`
- La cabeza lectora se coloca en la primera posición de w y la máquina queda en el estado inicial q_0

Máquinas de Turing: Funcionamiento

En cada paso:

- La máquina lee el símbolo a en la celda apuntada por la cabeza lectora y determina en que estado q se encuentra
- Busca en el programa una instrucción para (q, a). Si esta instrucción no existe, entonces la máquina se detiene
- Si la instrucción existe, entonces la ejecuta: Escribe un símbolo en la posición apuntada por la cabeza lectora, pasa a un nuevo estado y mueve la cabeza lectora a la derecha o a la izquierda

Máquinas de Turing: Funcionamiento

En cada paso:

- La máquina lee el símbolo a en la celda apuntada por la cabeza lectora y determina en que estado q se encuentra
- Busca en el programa una instrucción para (q, a). Si esta instrucción no existe, entonces la máquina se detiene
- Si la instrucción existe, entonces la ejecuta: Escribe un símbolo en la posición apuntada por la cabeza lectora, pasa a un nuevo estado y mueve la cabeza lectora a la derecha o a la izquierda

Si la máquina se detiene en un estado final, entonces la máquina acepta w

Máquinas de Turing: Formalización

Definición

Máquina de Turing (determinista): (Q, A, q_0, δ, F)

- Q *es un conjunto finito de estados*
- A *es un alfabeto tal que* $\mathtt{B} \notin A$
- q_0 *es el estado inicial* $(q_0 \in Q)$
- F *es un conjunto de estados finales* $(F \neq \emptyset)$
- δ *es una función parcial:*

$$\delta \quad : \quad Q \times (A \cup \{\mathtt{B}\}) \to Q \times (A \cup \{\mathtt{B}\}) \times \{\leftarrow, \rightarrow\}$$

δ *es llamada función de transición. Suponemos que su dominio no es vacío*

Máquinas de Turing: Ejemplo

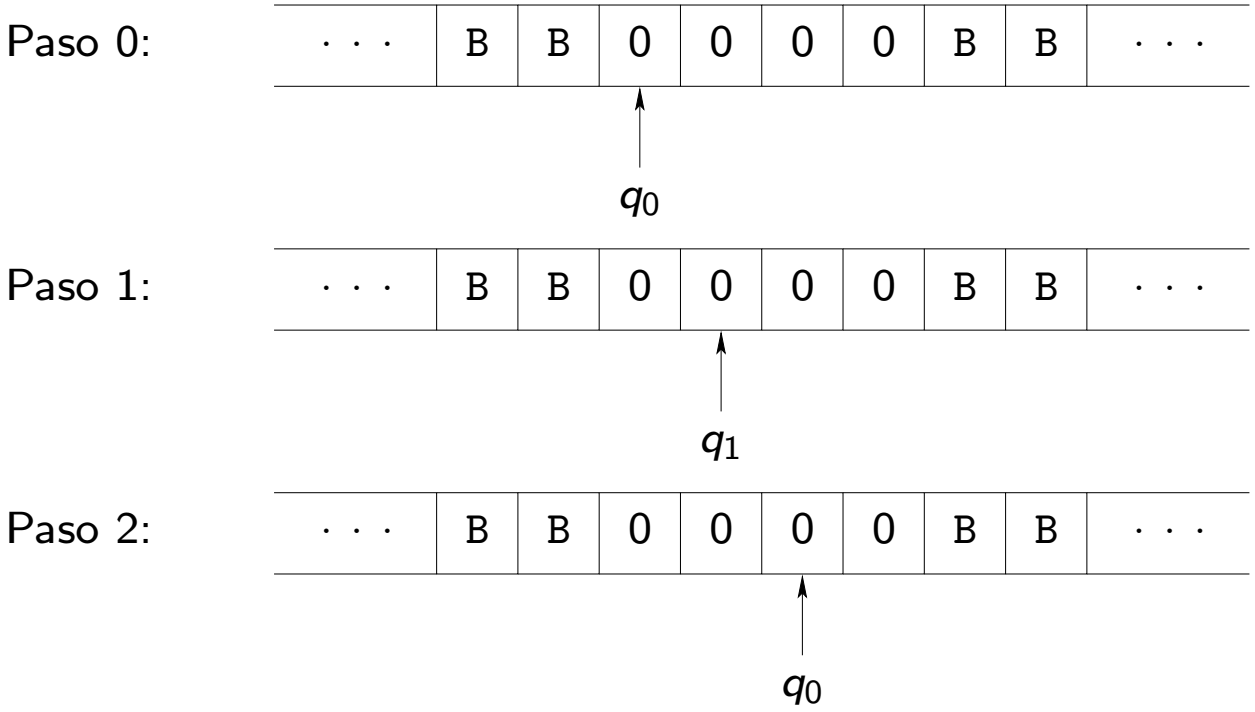
Queremos construir una máquina que verifique si el largo de una palabra de 0s es par: $M = (Q, A, q_0, \delta, F)$

- $A = \{0\}$
- $Q = \{q_0, q_1, q_S, q_N\}$
- $F = \{q_S\}$
- δ es definida como:

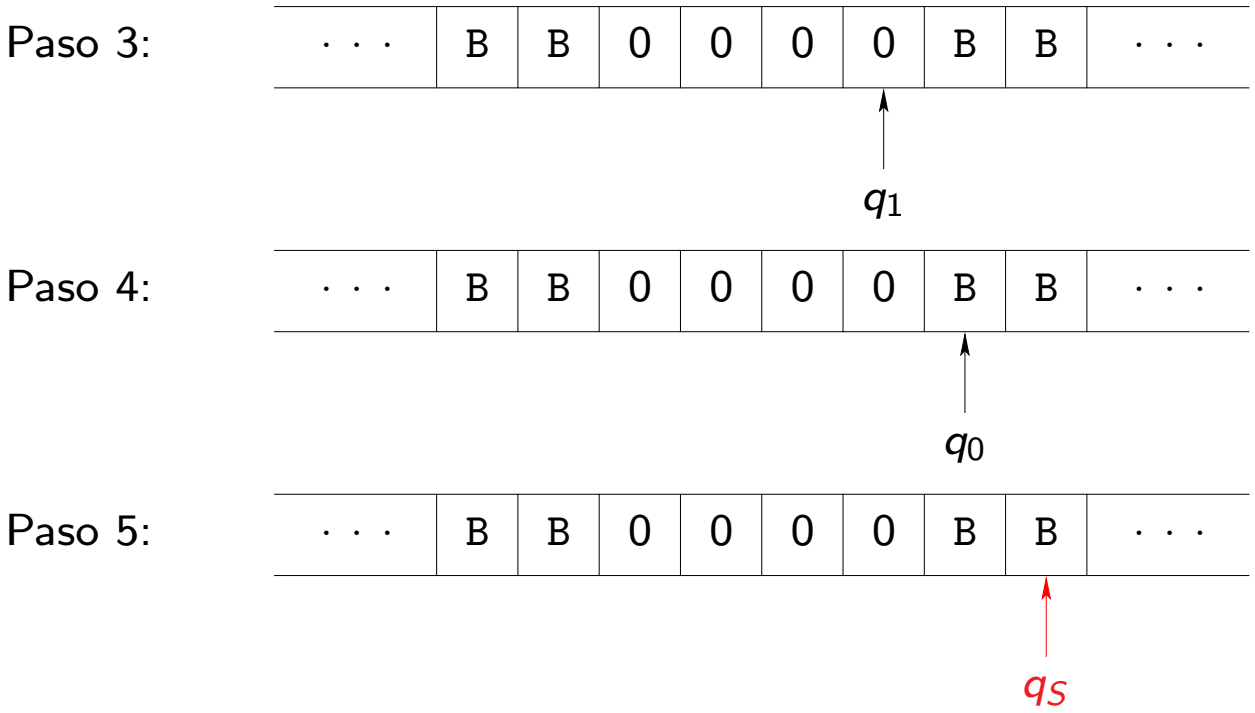
$$\begin{aligned}
\delta(q_0, 0) &= (q_1, 0, \rightarrow) \\
\delta(q_0, \mathrm{B}) &= (q_S, \mathrm{B}, \rightarrow) \\
\delta(q_1, 0) &= (q_0, 0, \rightarrow) \\
\delta(q_1, \mathrm{B}) &= (q_N, \mathrm{B}, \rightarrow)
\end{aligned}$$

Máquinas de Turing: Ejecución

Supongamos que $w = 0000$:

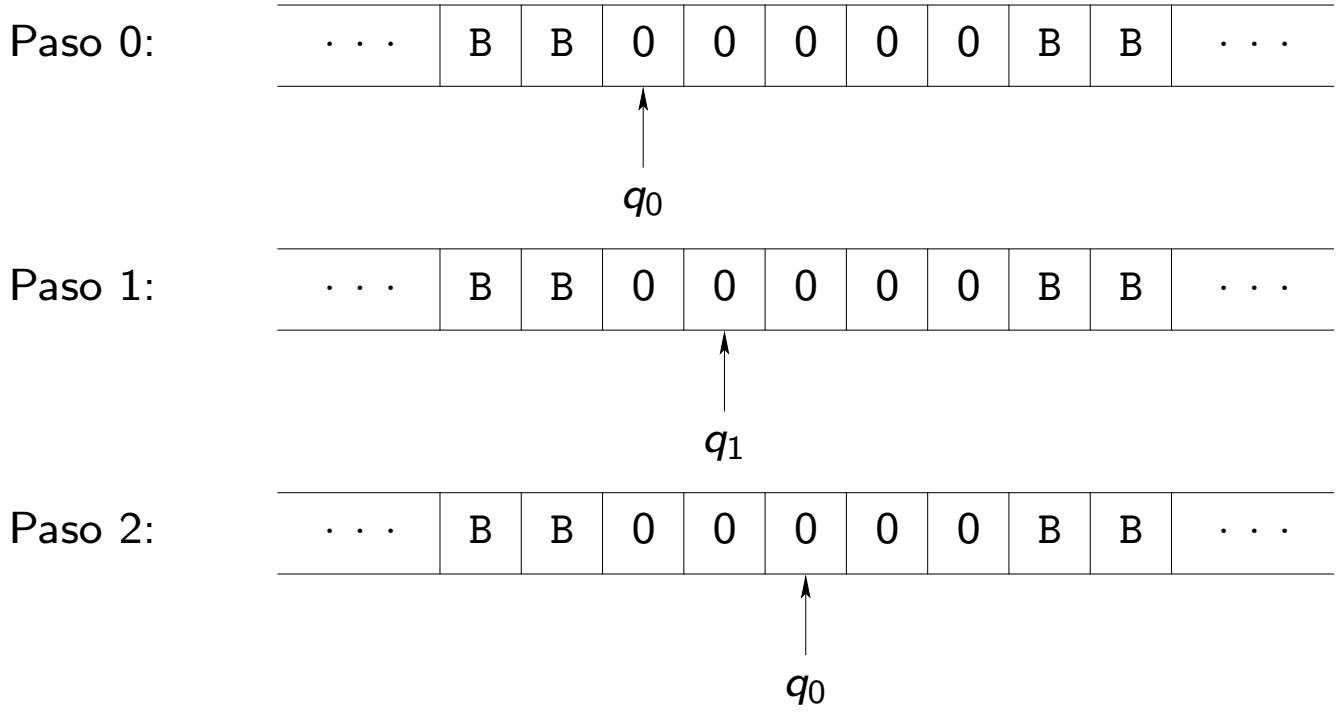


Máquinas de Turing: Ejecución

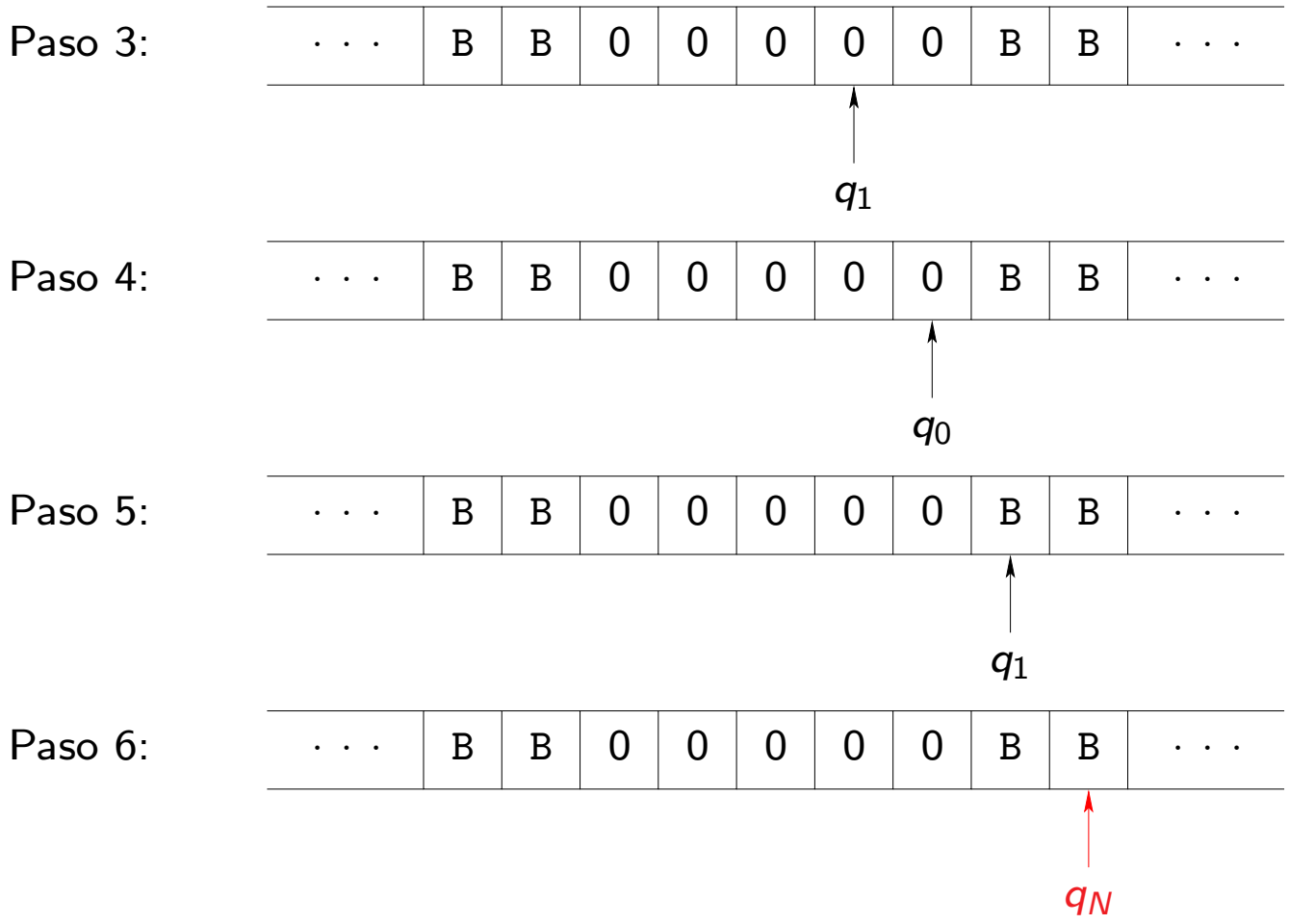


Máquinas de Turing: Otra ejecución

Ahora supongamos que $w = 00000$:

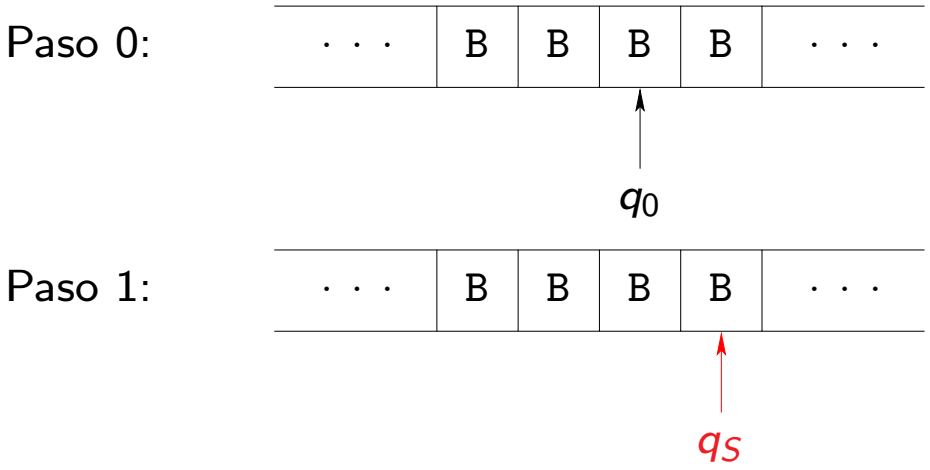


Máquinas de Turing: Otra ejecución



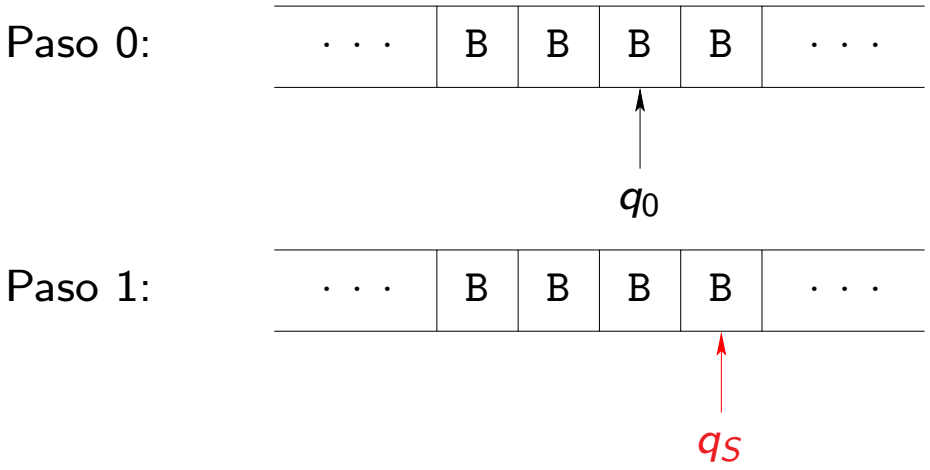
Máquinas de Turing: Palabra vacía

Finalmente supongamos que $w = \varepsilon$:



Máquinas de Turing: Palabra vacía

Finalmente supongamos que $w = \varepsilon$:



En este case la cabeza lectora se encuentra en una posición arbitraria en el paso inicial

El lenguaje aceptado por una máquina de Turing

Definición

Dada una máquina de Turing $M = (Q, A, q_0, \delta, F)$*:*

$$L(M) \quad = \quad \{w \in A^* \mid M \text{ acepta } w\}$$

$L(M)$ *es el lenguaje aceptado por* M

El lenguaje aceptado por una máquina de Turing

Definición

Dada una máquina de Turing $M = (Q, A, q_0, \delta, F)$*:*

$$L(M) \quad = \quad \{w \in A^* \mid M \text{ acepta } w\}$$

$L(M)$ *es el lenguaje aceptado por* M

En el ejemplo anterior: $L(M) = \{w \in \{0\}^* \mid \text{largo de } w \text{ es par}\}$

El lenguaje aceptado por una máquina de Turing

Ejercicio

1. Construya una máquina de Turing que acepta el lenguaje de las palabras de 0s que tienen largo divisible por 3.

2. Construya una máquina de Turing que acepta el lenguaje de las palabras de 0s que tienen largo divisible por 6.

3. Construya una máquina de Turing que acepta el lenguaje de las palabras de 0s y 1s que tienen el mismo número de 0s y 1s.

Detención de una máquina de Turing

Una máquina de Turing puede no detenerse en alguna entrada.

Detención de una máquina de Turing

Una máquina de Turing puede no detenerse en alguna entrada.

Ejemplo

$M = (Q, A, q_0, \delta, F)$, donde $Q = \{q_0, q_1\}$, $A = \{0\}$, $F = \{q_1\}$ y δ es definida de la siguiente forma:

$$\begin{aligned} \delta(q_0, 0) &= (q_1, 0, \rightarrow) \\ \delta(q_0, \mathrm{B}) &= (q_0, \mathrm{B}, \rightarrow) \end{aligned}$$

¿Con qué palabra no se detiene esta máquina?

Problemas decidibles

Si una máquina de Turing no se detiene en alguna entrada, entonces no puede ser utilizada como un algoritmo.

- Un buen programa en C, Java o Python no se debería quedar en un loop infinito

Problemas decidibles

Si una máquina de Turing no se detiene en alguna entrada, entonces no puede ser utilizada como un algoritmo.

- Un buen programa en C, Java o Python no se debería quedar en un loop infinito

Definición

Un lenguaje L es decidible si existe una máquina de Turing M que se detiene en todas las entradas y tal que $L = L(M)$.

Problemas decidibles

Si una máquina de Turing no se detiene en alguna entrada, entonces no puede ser utilizada como un algoritmo.

- Un buen programa en C, Java o Python no se debería quedar en un loop infinito

Definición

Un lenguaje L es decidible si existe una máquina de Turing M que se detiene en todas las entradas y tal que $L = L(M)$.

¿Existen problemas para los cuales no hay algoritmos? ¿Se puede demostrar esto?

Existencia de problemas no decidibles

Sea $A = \{0, 1\}$.

- ¿Cuántos lenguajes con alfabeto A existen?
- ¿Cuántas máquinas de Turing con alfabeto A existen?
- ¿Existen lenguajes no decidibles?
  - Si lanzamos un dardo al azar sobre el espacio de todos los lenguajes con alfabeto A, ¿cuál es la probabilidad de que el dardo caiga en un problema decidible?

Existencia de problemas no decidibles

La demostración anterior no es constructiva.

- Sabemos que existe un problema indecidible, pero no sabemos como es este lenguaje

Vamos a construir un lenguaje indecidible.

- Usamos diagonalización

Codificación de una máquina de Turing

Primero tenemos que introducir algo de terminología.

Codificación de una máquina de Turing

Primero tenemos que introducir algo de terminología.

Supongamos que $M = (Q, A, q_1, \delta, F)$, donde

- $Q = \{q_1, \ldots, q_n\}$
- $A = \{0, 1\}$
- $F = \{q_{i_1}, q_{i_2}, \ldots, q_{i_m}\}$, $1 \leq i_1 < i_2 < \cdots < i_m \leq n$

Definimos una palabra $C(M) = w_Q 00 w_\delta 00 w_F$ que representa a M.

Codificación de una máquina de Turing

- w_Q:

$$001001100\cdots00\underbrace{1\cdots1}_{n \text{ veces}}$$

Codificación de una máquina de Turing

- w_Q:

$$001001100\cdots00\underbrace{1\cdots1}_{n \text{ veces}}$$

- w_δ: Usamos 1 para representar símbolo 0, 11 para símbolo 1 y 111 para símbolo B. También usamos 1 para representar $\leftarrow$ y 11 para $\rightarrow$.

  w_δ es de la forma $w_1 \cdots w_k$, donde cada w_i representa una transición. Por ejemplo, si $\delta(q_i, 0) = (q_j, \mathrm{B}, \leftarrow)$, entonces la siguiente palabra representa esta transición:

$$00\underbrace{1\cdots1}_{i \text{ veces}}00100\underbrace{1\cdots1}_{j \text{ veces}}00111001$$

Codificación de una máquina de Turing

En cambio, si $\delta(q_i, 1) = (q_j, 0, \rightarrow)$, entonces la siguiente palabra representa la transición:

$$00\underbrace{1 \cdots 1}_{i \text{ veces}}001100\underbrace{1 \cdots 1}_{j \text{ veces}}0010011$$

Codificación de una máquina de Turing

En cambio, si $\delta(q_i, 1) = (q_j, 0, \rightarrow)$, entonces la siguiente palabra representa la transición:

$$00\underbrace{1\cdots1}_{i \text{ veces}}001100\underbrace{1\cdots1}_{j \text{ veces}}0010011$$

- w_F:

$$00\underbrace{1\cdots1}_{i_1 \text{ veces}}00\underbrace{1\cdots1}_{i_2 \text{ veces}}00\cdots00\underbrace{1\cdots1}_{i_m \text{ veces}}$$

Codificación de una máquina de Turing: Ejemplo

Sea $M = (Q, A, q_1, \delta, F)$, donde $Q = \{q_1, q_2, q_3, q_4\}$, $A = \{0, 1\}$, $F = \{q_3\}$ y δ es definido de la siguiente forma:

$$\begin{array}{lcl} \delta(q_1, 0) & = & (q_1, 0, \rightarrow) \\ \delta(q_1, 1) & = & (q_2, 1, \rightarrow) \\ \delta(q_2, 1) & = & (q_4, 1, \leftarrow) \end{array} \qquad \begin{array}{lcl} \delta(q_2, \mathtt{B}) & = & (q_3, \mathtt{B}, \leftarrow) \\ \delta(q_2, 0) & = & (q_4, 0, \leftarrow) \end{array}$$

Entonces: $C(M) = w_Q 00 w_\delta 00 w_F$, donde:

$$\begin{array}{lcl} w_Q & = & 0010011001110011111 \\ w_F & = & 00111 \\ w_\delta & = & 00100100100100110010011001100110011 \\ & & 00110011100111001110010011001001111001001 \\ & & 0011001100111110011001 \end{array}$$

Un problema indecidible: Problema de la parada

Problema natural

Dada una máquina de Turing M y una palabra w, ¿se detiene M con entrada w?

Un problema indecidible: Problema de la parada

Problema natural

Dada una máquina de Turing M y una palabra w, ¿se detiene M con entrada w?

Como problema de decisión: Consideramos máquinas de Turing con alfabeto $\{0, 1\}$. Entonces,

$$U = \{w' \in \{0,1\}^* \mid \text{existe una MT } M \text{ y } w \in A^* \text{ tal que } w' = C(M)0000w \text{ y } M \text{ se detiene con entrada } w\}$$

Un problema indecidible: Problema de la parada

Problema natural

Dada una máquina de Turing M y una palabra w, ¿se detiene M con entrada w?

Como problema de decisión: Consideramos máquinas de Turing con alfabeto $\{0, 1\}$. Entonces,

$$U = \{w' \in \{0,1\}^* \mid \text{existe una MT } M \text{ y } w \in A^* \text{ tal que } w' = C(M)0000w \text{ y } M \text{ se detiene con entrada } w\}$$

¿Es este problema decidible?

- Vamos a demostrar que no, pero para eso vamos a considerar primero un problema más simple

Un problema de decisión más simple

Otro problema de decisión

Dada una máquina de Turing M, ¿es cierto que M se detiene con entrada $C(M)$?

Le damos como entrada a M su propio código, ¿es válido hacer esto?

- Sí, $C(M)$ es una palabra como cualquier otra

Un problema de decisión más simple

Otro problema de decisión

Dada una máquina de Turing M, ¿es cierto que M se detiene con entrada $C(M)$?

Le damos como entrada a M su propio código, ¿es válido hacer esto?

- Sí, $C(M)$ es una palabra como cualquier otra

Como problema de decisión:

$$H = \{w \in \{0,1\}^* \mid \text{existe una MT } M \text{ tal que } w = C(M) \text{ y } M \text{ se detiene con entrada } C(M)\}$$

Un problema de decisión más simple: Demostración

Teorema

H es indecidible, vale decir, no existe una máquina de Turing M que se detiene en todas las entradas y tal que $H = L(M)$.

Un problema de decisión más simple: Demostración

Teorema

H es indecidible, vale decir, no existe una máquina de Turing M que se detiene en todas las entradas y tal que $H = L(M)$.

Demostración: Por contradicción, asuma que existe una MT $M^\star$ que se detiene en todas las entradas y tal que $H = L(M^\star)$.

Vale decir: para cada MT M, si M se detiene con su propio código, entonces $M^\star$ con entrada $C(M)$ se detiene en un estado final. En caso contrario, $M^\star$ se detiene en un estado que no es final.

Un problema de decisión más simple: Demostración

Sea M_0 una MT que con entrada w funciona de la siguiente forma:

hace funcionar la máquina $M^\star$ con entrada w
si $M^\star$ se detiene en un estado final
    entonces M_0 no se detiene
    en caso contrario M_0 se detiene en un estado final

Un problema de decisión más simple: Demostración

Sea M_0 una MT que con entrada w funciona de la siguiente forma:

hace funcionar la máquina $M^\star$ con entrada w
si $M^\star$ se detiene en un estado final
    entonces M_0 no se detiene
    en caso contrario M_0 se detiene en un estado final

¿Es posible construir M_0? ¿Cómo?

Un problema de decisión más simple: Demostración

La pregunta ahora es si $M^\star$ acepta M_0:

$$
\begin{gathered}
M^\star \text{ acepta } M_0 \\
\text{si y sólo si} \\
M_0 \text{ se detiene con entrada } C(M_0) \\
\text{si y sólo si} \\
M^\star \text{ se detiene en un estado no final con entrada } C(M_0) \\
\text{si y sólo si} \\
M^\star \text{ no acepta } M_0
\end{gathered}
$$

¡Tenemos una contradicción!

Un problema indecidible: Problema de la parada

Podemos concluir que nuestro problema original U es indecidible.

- ¿Por qué?

Un problema indecidible: Problema de la parada

Podemos concluir que nuestro problema original U es indecidible.

- ¿Por qué?

Pero, ¿cómo demostramos formalmente que U es indecidible?

- Tenemos que formalizar el concepto de reducción