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

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 $w \in L$.

Ejemplo: Podemos ver SAT como un problema de decisión.

Supongamos 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$\}$.

Ejercicio: Represente el problema de 3-coloración de grafos como un problema de decisión.

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 $L = \{w \in \{0,1\}^* \mid$ número de 0s y 1s en w 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.

¿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.
- Tesis de Church: **Algoritmo = Máquina de Turing.**

Máquinas de Turing: Componentes

Dado: Alfabeto A que no contiene símbolo especial `B`.

Componentes:

- Memoria: Arreglo infinito (en ambas direcciones) que en cada posición almacena un símbolo de $A \cup \{\texttt{B}\}$.
- 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.

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

Máquinas de Turing: Formalización

Máquina de Turing: (Q, A, q_0, δ, F)

- Q es un conjunto finito de estados.
- A es un alfabeto.
- 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:

$$\delta \quad : \quad Q \times A \rightarrow Q \times A \times \{I, D\}.$$

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

Máquinas de Turing: Ejemplo

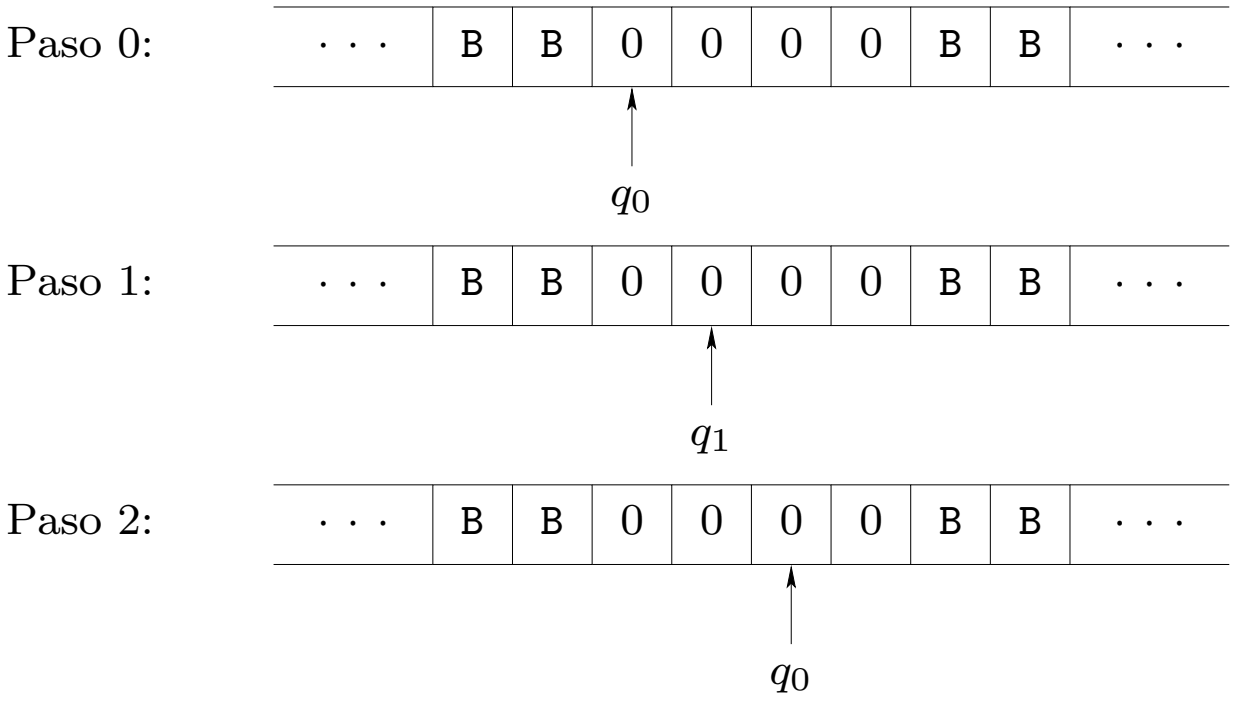
Queremos construir una máquina que verifique si el largo de una palabra de 0s es par: $M = (Q, A, q_0, \delta, F)$

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

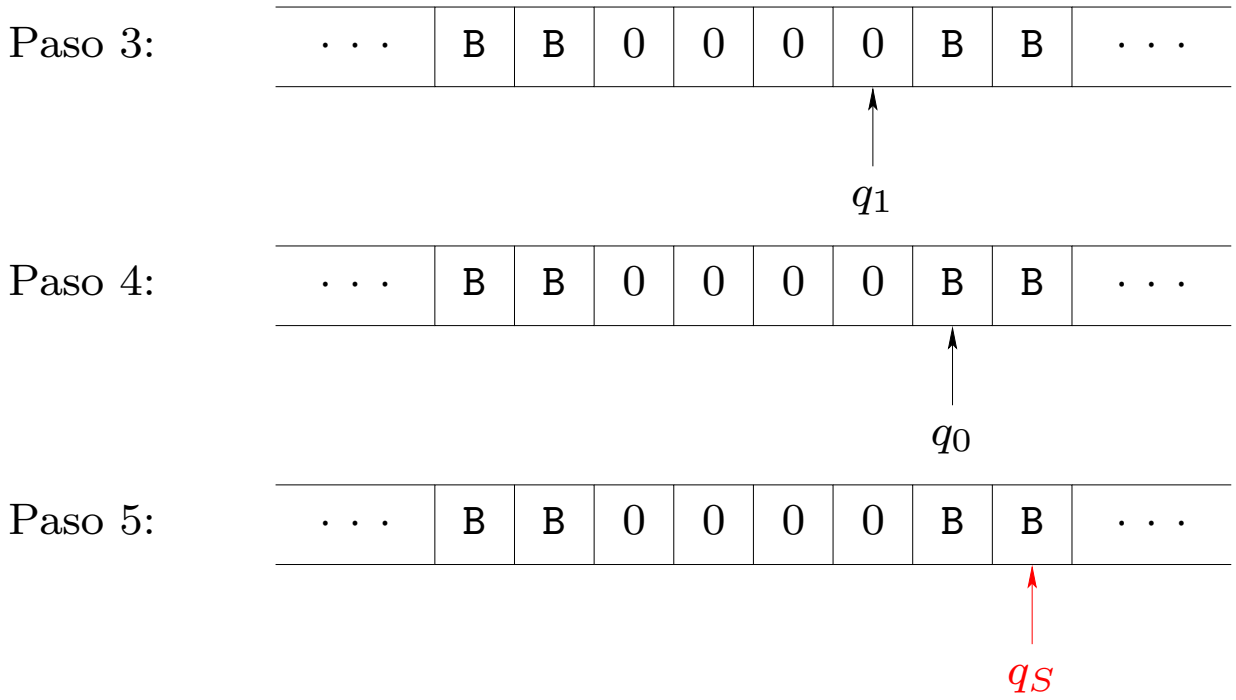
$$\begin{aligned} \delta(q_0, 0) &= (q_1, 0, D) \\ \delta(q_0, \mathtt{B}) &= (q_S, \mathtt{B}, D) \\ \delta(q_1, 0) &= (q_0, 0, D) \\ \delta(q_1, \mathtt{B}) &= (q_N, \mathtt{B}, D) \end{aligned}$$

Máquinas de Turing: Ejecución

Supongamos que $w = 0000$:

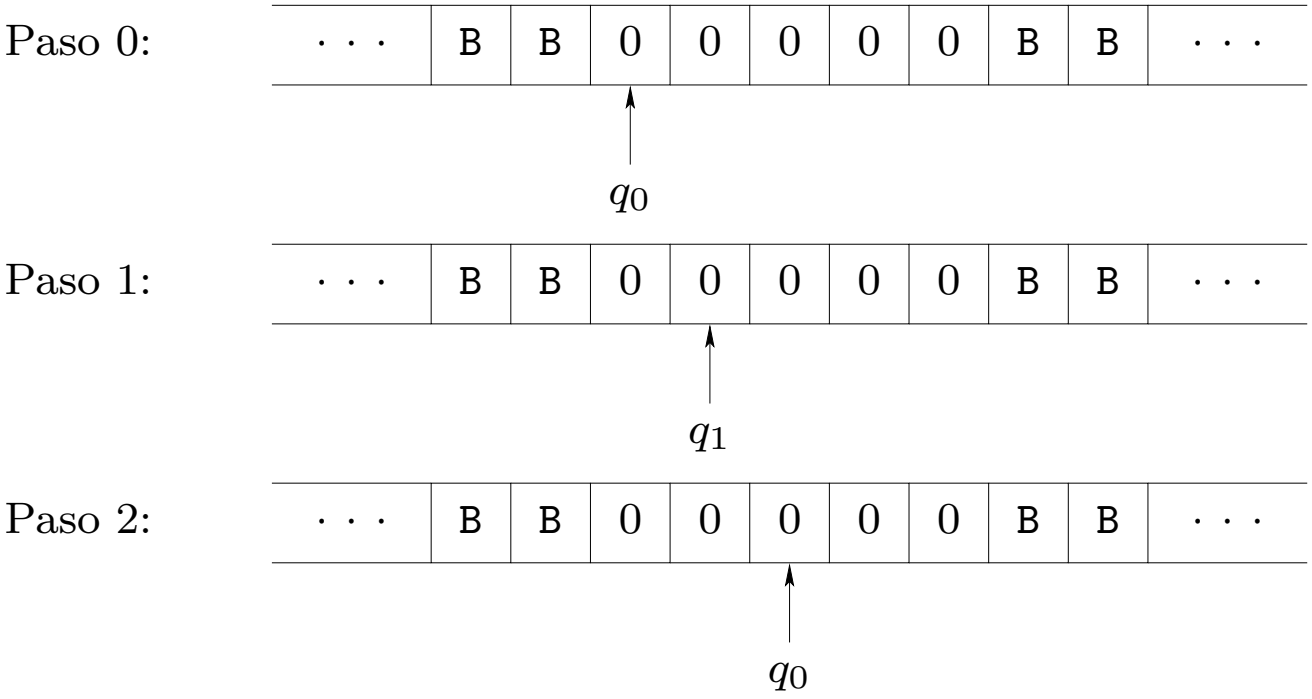


Máquinas de Turing: Ejecución

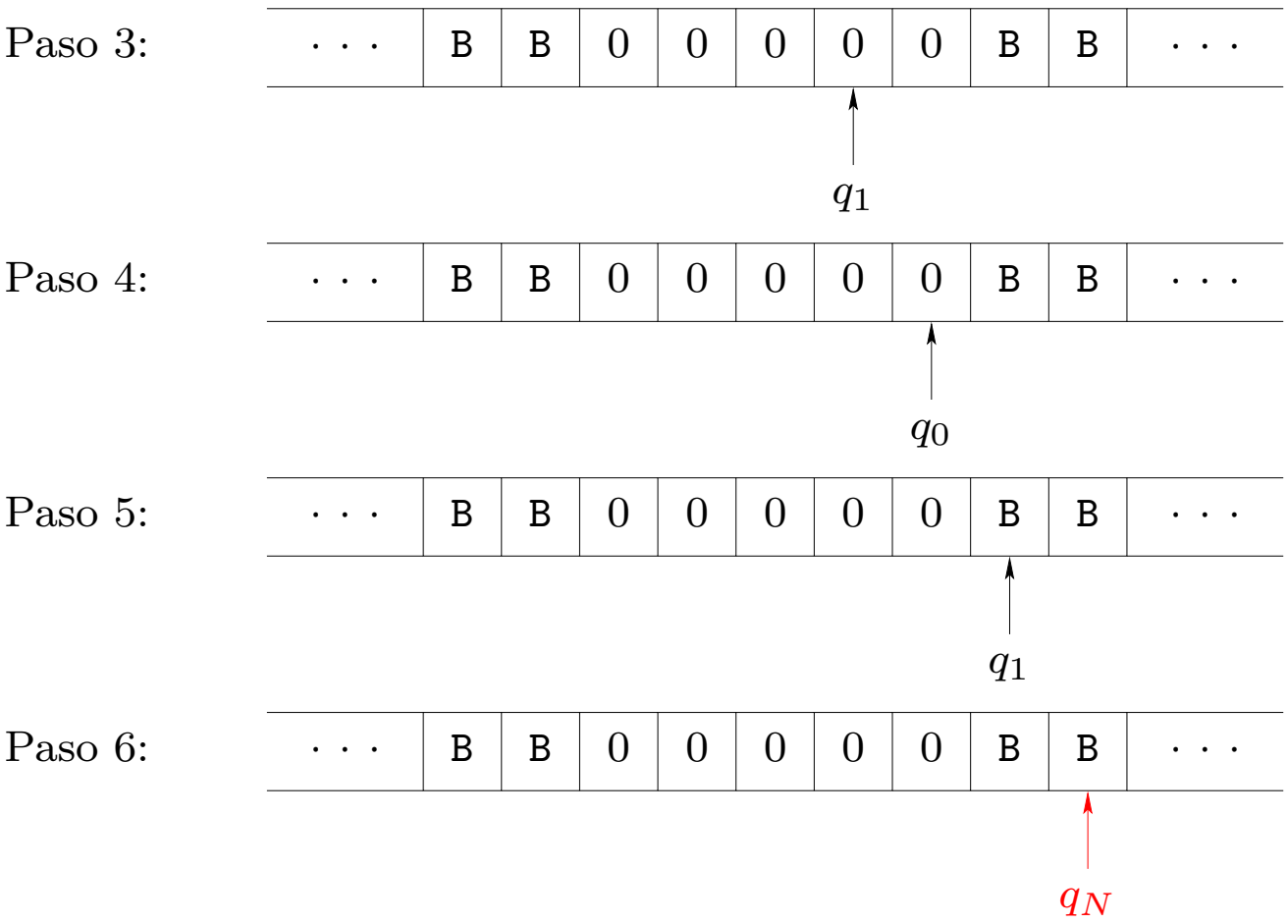


Máquinas de Turing: Otra ejecución

Ahora supongamos que $w = 00000$:

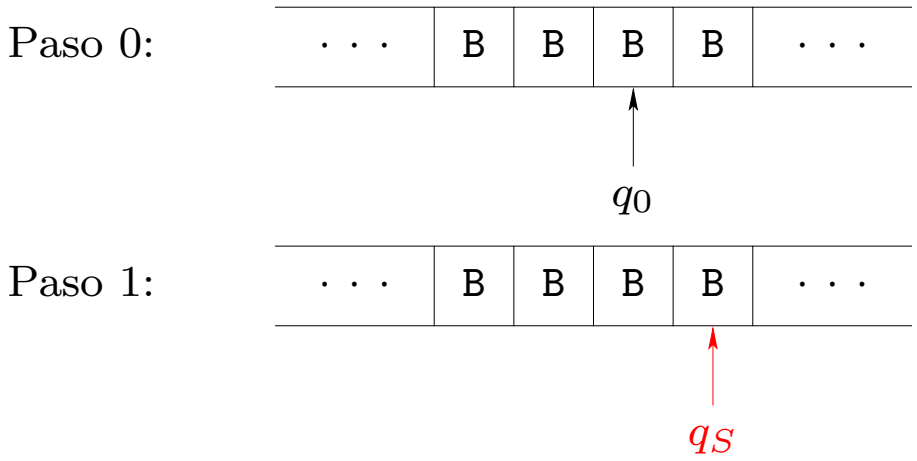


Máquinas de Turing: Otra ejecución



Máquinas de Turing: Palabra vacía

Finalmente supongamos que $w = \varepsilon$:



Inicialmente: La cabeza lectora se encuentra en una posición cualquiera.

El lenguaje aceptado por una máquina de Turing

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

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

Ejercicio: 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.

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:

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

¿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 no se debería quedar en un loop infinito.

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

- Existe un algoritmo que verifica si una palabra w está en L.

Existen algoritmos para todos los problemas que hemos visto hasta ahora.

- ¿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?