Introducción

IIC2283

El objetivo de este curso

Introducir técnicas tanto para el diseño como para el análisis de la complejidad computacional de un algoritmo

El objetivo de este curso

Introducir técnicas tanto para el diseño como para el análisis de la complejidad computacional de un algoritmo

- Técnicas básicas y avanzadas

El objetivo de este curso

Introducir técnicas tanto para el diseño como para el análisis de la complejidad computacional de un algoritmo

- Técnicas básicas y avanzadas

Se dará énfasis a:

- La compresión del modelo computacional sobre el cual se diseña y analiza un algoritmo
- El uso de estructuras de datos adecuadas para la implementación de un algoritmo
- El uso de ejemplos de distintas áreas para mostrar las potencialidades de las técnicas estudiadas

¿Qué es un algoritmo?

¿Podemos formalizar este concepto?

¿Qué es un algoritmo?

¿Podemos formalizar este concepto?

- Máquinas de Turing: Intento por formalizar este concepto

¿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?

¿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?

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

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

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

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: Formalización

Definición

Máquina de Turing: $(Q, \Sigma, \Gamma, q_0, \delta, F)$

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

$$\delta \quad : \quad Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\}$$

δ *es llamada función de transición*

Máquinas de Turing: Funcionamiento

La cinta de la máquina de Turing es infinita en ambas direcciones.

Σ es el alfabeto de entrada y Γ es el alfabeto de la cinta.

- Una palabra de entrada $w \in \Sigma^*$ es colocada en posiciones consecutivas de la cinta
- Todas las otras posiciones contienen el símbolo $\mathtt{B}$

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición inicial de la palabra de entrada

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición inicial de la palabra de entrada

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición inicial de la palabra de entrada

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición inicial de la palabra de entrada

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'
  - Mueve la cabeza lectora a la posición $p - 1$ si X es $\leftarrow$, y a la posición $p + 1$ si X es $\rightarrow$. Si X es $\square$, entonces la cabeza lectora permanece en la posición p

Máquinas de Turing: Aceptación

Los estados de F son utilizados como estados de aceptación.

- Una palabra w es aceptada por una máquina M si y sólo si la ejecución de M con entrada w se detiene en un estado de F

Definición

Lenguaje aceptado por una máquina de Turing M:

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

Máquinas de Turing: Ejemplo

Queremos construir una máquina de Turing M que acepte el lenguaje $L = \{w \in \{0,1\}^* \mid w \text{ es un palíndromo}\}$

Máquinas de Turing: Ejemplo

Queremos construir una máquina de Turing M que acepte el lenguaje $L = \{w \in \{0,1\}^* \mid w \text{ es un palíndromo}\}$

Definimos $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ de la siguiente forma:

- $Q = \{q_0, q_f, q^0_{\rightarrow}, q^1_{\rightarrow}, q^0_{v}, q^1_{v}, q_{\leftarrow}\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, \mathtt{B}\}$
- $F = \{q_f\}$

Máquinas de Turing: Ejemplo

- δ es definida como:

$$
\begin{array}{lcl}
\delta(q_0, \mathtt{B}) & = & (q_f, \mathtt{B}, \square) \\
\delta(q_0, 0) & = & (q^0_{\rightarrow}, \mathtt{B}, \rightarrow) \\
\delta(q_0, 1) & = & (q^1_{\rightarrow}, \mathtt{B}, \rightarrow)
\end{array}
\qquad
\begin{array}{lcl}
\delta(q_{\leftarrow}, \mathtt{B}) & = & (q_0, \mathtt{B}, \rightarrow) \\
\delta(q_{\leftarrow}, 0) & = & (q_{\leftarrow}, 0, \leftarrow) \\
\delta(q_{\leftarrow}, 1) & = & (q_{\leftarrow}, 1, \leftarrow)
\end{array}
$$

$$
\begin{array}{lcl}
\delta(q^0_{\rightarrow}, \mathtt{B}) & = & (q^0_v, \mathtt{B}, \leftarrow) \\
\delta(q^0_{\rightarrow}, 0) & = & (q^0_{\rightarrow}, 0, \rightarrow) \\
\delta(q^0_{\rightarrow}, 1) & = & (q^0_{\rightarrow}, 1, \rightarrow)
\end{array}
\qquad
\begin{array}{lcl}
\delta(q^0_v, \mathtt{B}) & = & (q_f, \mathtt{B}, \square) \\
\delta(q^0_v, 0) & = & (q_{\leftarrow}, \mathtt{B}, \leftarrow)
\end{array}
$$

$$
\begin{array}{lcl}
\delta(q^1_{\rightarrow}, \mathtt{B}) & = & (q^1_v, \mathtt{B}, \leftarrow) \\
\delta(q^1_{\rightarrow}, 0) & = & (q^1_{\rightarrow}, 0, \rightarrow) \\
\delta(q^1_{\rightarrow}, 1) & = & (q^1_{\rightarrow}, 1, \rightarrow)
\end{array}
\qquad
\begin{array}{lcl}
\delta(q^1_v, \mathtt{B}) & = & (q_f, \mathtt{B}, \square) \\
\delta(q^1_v, 1) & = & (q_{\leftarrow}, \mathtt{B}, \leftarrow)
\end{array}
$$

Complejidad de un algoritmo

¿Cuál es la complejidad del algoritmo anterior?

- ¿Qué operaciones contamos para realizar este análisis?

Complejidad de un algoritmo

¿Cuál es la complejidad del algoritmo anterior?

- ¿Qué operaciones contamos para realizar este análisis?

Para una MT M con alfabeto de entrada Σ:

- Paso de M: Ejecutar una instrucción de la función de transición
- $tiempo_M(w)$: Número de pasos ejecutados por M con entrada $w \in \Sigma^*$

Complejidad de un algoritmo

Supuesto

Consideramos máquinas de Turing que se detienen en todas sus entradas.

- En general consideramos algoritmos que se detienen en todas sus entradas

Complejidad de un algoritmo

Definición

El tiempo de funcionamiento de una MT M en el peor caso es definido por la función t_M:

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

Complejidad de un algoritmo

Definición

El tiempo de funcionamiento de una MT M en el peor caso es definido por la función t_M:

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

Ejercicio

Para la MT M del ejemplo sobre palíndromos demuestre que $t_M(n)$ es $O(n^2)$

Otro modelo de computación

Definición

Máquina de Turing con k cintas: $(Q, \Sigma, \Gamma, q_0, \delta, F)$

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

$$\delta \quad : \quad Q \times \Gamma^k \to Q \times \Gamma^k \times \{\leftarrow, \square, \rightarrow\}^k$$

δ *es llamada función de transición.*

MT con k cintas: Funcionamiento

La máquina tiene k cintas infinitas en ambas direcciones.

Σ es el alfabeto de entrada y Γ es el alfabeto de las cintas.

- Una palabra de entrada $w \in \Sigma^*$ es colocada en posiciones consecutivas de la primera cinta
- Las otras posiciones de la primera cinta contienen el símbolo `B`
- Las restantes cintas contienen el símbolo `B` en todas las posiciones

MT con k cintas: Funcionamiento

La máquina tiene una cabeza lectora por cinta.

Al comenzar la máquina se encuentra en el estado q_0, la cabeza lectora de la primera cinta está en la primera posición de la palabra de entrada, el resto de las cabezas están en posiciones arbitrarias.

MT con k cintas: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, \ldots, a_k) = (q', b_1, \ldots, b_k, X_1, \ldots, X_k)$, entonces:

MT con k cintas: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, \ldots, a_k) = (q', b_1, \ldots, b_k, X_1, \ldots, X_k)$, entonces:
  - La máquina escribe el símbolo b_i en la posición p_i de la i-ésima cinta

MT con k cintas: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, \ldots, a_k) = (q', b_1, \ldots, b_k, X_1, \ldots, X_k)$, entonces:
  - La máquina escribe el símbolo b_i en la posición p_i de la i-ésima cinta
  - Cambia de estado desde q a q'

MT con k cintas: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, \ldots, a_k) = (q', b_1, \ldots, b_k, X_1, \ldots, X_k)$, entonces:
  - La máquina escribe el símbolo b_i en la posición p_i de la i-ésima cinta
  - Cambia de estado desde q a q'
  - Mueve la cabeza lectora de la i-ésima cinta a la posición $p_i - 1$ si X_i es $\leftarrow$, y a la posición $p_i + 1$ si X_i es $\rightarrow$. Si X_i es $\square$, entonces la máquina no mueve la cabeza lectora de la i-ésima cinta

MT con k cintas: Aceptación y complejidad

Una palabra w es aceptada por una MT M con k cintas si y sólo si la ejecución de M con entrada w se detiene en un estado final. Tenemos entonces que:

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

MT con k cintas: Aceptación y complejidad

Una palabra w es aceptada por una MT M con k cintas si y sólo si la ejecución de M con entrada w se detiene en un estado final. Tenemos entonces que:

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

Para una MT con k cintas y alfabeto Σ:

- Paso de M: Ejecutar una instrucción de la función de transición
- $tiempo_M(w)$: Número de pasos ejecutados por M con entrada $w \in \Sigma^*$
- Tiempo de funcionamiento M en el peor caso:

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

MT con k cintas: Ejemplo

Construya una MT M con dos cintas que funcione en tiempo $O(n)$ y acepte el lenguaje $L = \{w \in \{0,1\}^* \mid w \text{ es un palíndromo}\}$.

MT con k cintas: Ejemplo

Construya una MT M con dos cintas que funcione en tiempo $O(n)$ y acepte el lenguaje $L = \{w \in \{0,1\}^* \mid w \text{ es un palíndromo}\}$.

Solución: Definimos $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ de la siguiente forma:

- $Q = \{q_0, q_c, q_r, q_v, q_f\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, \mathtt{B}\}$
- $F = \{q_f\}$

MT con k cintas: Ejemplo

- Función δ es definida de la siguiente forma:

$$
\begin{array}{lcl}
\delta(q_0, \mathtt{B}, \mathtt{B}) & = & (q_f, \mathtt{B}, \mathtt{B}, \square, \square) \\
\delta(q_0, 0, \mathtt{B}) & = & (q_c, 0, 0, \rightarrow, \rightarrow) \\
\delta(q_0, 1, \mathtt{B}) & = & (q_c, 1, 1, \rightarrow, \rightarrow) \\
\delta(q_c, 0, \mathtt{B}) & = & (q_c, 0, 0, \rightarrow, \rightarrow) \\
\delta(q_c, 1, \mathtt{B}) & = & (q_c, 1, 1, \rightarrow, \rightarrow) \\
\delta(q_c, \mathtt{B}, \mathtt{B}) & = & (q_r, \mathtt{B}, \mathtt{B}, \leftarrow, \leftarrow) \\
\delta(q_r, 0, 0) & = & (q_r, 0, 0, \leftarrow, \square) \\
\delta(q_r, 0, 1) & = & (q_r, 0, 1, \leftarrow, \square)
\end{array}
\qquad
\begin{array}{lcl}
\delta(q_r, 1, 0) & = & (q_r, 1, 0, \leftarrow, \square) \\
\delta(q_r, 1, 1) & = & (q_r, 1, 1, \leftarrow, \square) \\
\delta(q_r, \mathtt{B}, 0) & = & (q_v, \mathtt{B}, 0, \rightarrow, \square) \\
\delta(q_r, \mathtt{B}, 1) & = & (q_v, \mathtt{B}, 1, \rightarrow, \square) \\
\delta(q_v, 0, 0) & = & (q_v, 0, 0, \rightarrow, \leftarrow) \\
\delta(q_v, 1, 1) & = & (q_v, 1, 1, \rightarrow, \leftarrow) \\
\delta(q_v, \mathtt{B}, \mathtt{B}) & = & (q_f, \mathtt{B}, \mathtt{B}, \square, \square)
\end{array}
$$

Complejidad en distintos modelos

¿Es posible aceptar más rápido si se usa cintas adicionales?

- ¿Tenemos entonces que especificar bajo que modelo estamos resolviendo un problema?

Complejidad en distintos modelos

¿Es posible aceptar más rápido si se usa cintas adicionales?

- ¿Tenemos entonces que especificar bajo que modelo estamos resolviendo un problema?

Sea $L = \{w \in \{0, 1, \#\}^* \mid w \text{ es un palíndromo}\}$

- L es aceptado por una MT con dos cintas en tiempo $O(n)$
- ¿Puede ser L aceptado en tiempo lineal por una MT con una cinta?

Una cota inferior en un modelo de computación

Proposición

Sea $L = \{w \in \{0,1,\#\}^* \mid w \text{ es un palíndromo}\}$ *y* M *una MT con una cinta. Si* $L = L(M)$*, entonces* M *funciona en tiempo* $\Omega(n^2)$*.*

Una cota inferior en un modelo de computación

Proposición

Sea $L = \{w \in \{0,1,\#\}^* \mid w$ *es un palíndromo*$\}$ *y* M *una MT con una cinta. Si* $L = L(M)$*, entonces* M *funciona en tiempo* $\Omega(n^2)$*.*

Demostración: Suponga que $L = L(M)$, donde M es una MT con una cinta.

- Sea Q el conjunto de estados de M

Una cota inferior en un modelo de computación

Sin perdida de generalidad, suponemos que M siempre recorre toda la palabra de entrada.

- ¿Por qué podemos suponer esto?

Una cota inferior en un modelo de computación

Sin perdida de generalidad, suponemos que M siempre recorre toda la palabra de entrada.

- ¿Por qué podemos suponer esto?

Para $w \in \{0, 1, \#\}^*$, sea w^r la palabra obtenida al escribir w en el sentido inverso.

Una cota inferior en un modelo de computación

Sin perdida de generalidad, suponemos que M siempre recorre toda la palabra de entrada.

- ¿Por qué podemos suponer esto?

Para $w \in \{0, 1, \#\}^*$, sea w^r la palabra obtenida al escribir w en el sentido inverso.

Defina L_n como el siguiente lenguaje (para $n > 0$ divisible por 4):

$$L_n \quad = \quad \{w\#^{\frac{n}{2}} w^r \mid w \in \{0,1\}^{\frac{n}{4}}\}$$

Nótese que $L_n \subseteq L$

Complejidad en distintos modelos

Sea $w \in L_n$ y $\frac{n}{4} \leq i \leq \frac{3n}{4}$. Además, sea $C_i(w)$ la secuencia de estados $[q_1, \ldots, q_k]$ en que se encuentra M después de moverse entre las posiciones i e $i+1$ (en cualquiera de las dos direcciones) en la ejecución que tiene a w como entrada.

Complejidad en distintos modelos

Sea $w \in L_n$ y $\frac{n}{4} \leq i \leq \frac{3n}{4}$. Además, sea $C_i(w)$ la secuencia de estados $[q_1, \ldots, q_k]$ en que se encuentra M después de moverse entre las posiciones i e $i+1$ (en cualquiera de las dos direcciones) en la ejecución que tiene a w como entrada.

Definimos $C(w) = \{C_i(w) \mid \frac{n}{4} \leq i \leq \frac{3n}{4}\}$

Complejidad en distintos modelos

Lema

Si $w_1, w_2 \in L_n$ *y* $w_1 \neq w_2$, *entonces* $C(w_1) \cap C(w_2) = \emptyset$

Complejidad en distintos modelos

Lema

Si $w_1, w_2 \in L_n$ *y* $w_1 \neq w_2$, *entonces* $C(w_1) \cap C(w_2) = \emptyset$

Demostración: Suponga que el lema es falso. Entonces existen $i, j \in \{\frac{n}{4}, \ldots, \frac{3n}{4}\}$ tales que $C_i(w_1) = C_j(w_2)$.

Sean u_1 y u_2 las palabra formadas por los primeros i símbolos de w_1 y los últimos $n - j$ símbolos de w_2, respectivamente.

Dado que $C_i(w_1) = C_j(w_2)$, se tiene que $u_1 u_2$ es aceptado por M.

- ¿Cómo se demuestra esto?

Pero $u_1 u_2$ no es un palíndromo, por lo que tenemos una contradicción. □

Complejidad en distintos modelos

Para $w \in L_n$, sea s_w la secuencia más corta en $C(w)$

- $S_n = \{s_w \mid w \in L_n\}$

Complejidad en distintos modelos

Para $w \in L_n$, sea s_w la secuencia más corta en $C(w)$

- $S_n = \{s_w \mid w \in L_n\}$

Por el lema sabemos que $s_{w_1} \neq s_{w_2}$ si $w_1 \neq w_2$

- Por lo tanto: $|S_n| = |L_n| = 2^{\frac{n}{4}}$

Complejidad en distintos modelos

Para $w \in L_n$, sea s_w la secuencia más corta en $C(w)$

- $S_n = \{s_w \mid w \in L_n\}$

Por el lema sabemos que $s_{w_1} \neq s_{w_2}$ si $w_1 \neq w_2$

- Por lo tanto: $|S_n| = |L_n| = 2^{\frac{n}{4}}$

Sea m el largo de la secuencia mas larga en S_n

- Cantidad de posibles secuencias de largo a lo más m:

$$\sum_{i=0}^{m} |Q|^i = \frac{|Q|^{m+1} - 1}{|Q| - 1}$$

Complejidad en distintos modelos

De lo anterior concluimos que: $\frac{|Q|^{m+1}-1}{|Q|-1} \geq 2^{\frac{n}{4}}$

- ¿Por qué?

Complejidad en distintos modelos

De lo anterior concluimos que: $\frac{|Q|^{m+1}-1}{|Q|-1} \geq 2^{\frac{n}{4}}$

- ¿Por qué?

Se tiene entonces que m es $\Omega(n)$

- Por lo tanto existe $w_0 \in L_n$ para el cual $|s_{w_0}|$ es $\Omega(n)$

Complejidad en distintos modelos

De lo anterior concluimos que: $\frac{|Q|^{m+1}-1}{|Q|-1} \geq 2^{\frac{n}{4}}$

- ¿Por qué?

Se tiene entonces que m es $\Omega(n)$

- Por lo tanto existe $w_0 \in L_n$ para el cual $|s_{w_0}|$ es $\Omega(n)$

Entonces: Todas las secuencias en $C(w_0)$ son de largo $\Omega(n)$

Complejidad en distintos modelos

De lo anterior concluimos que: $\frac{|Q|^{m+1}-1}{|Q|-1} \geq 2^{\frac{n}{4}}$

- ¿Por qué?

Se tiene entonces que m es $\Omega(n)$

- Por lo tanto existe $w_0 \in L_n$ para el cual $|s_{w_0}|$ es $\Omega(n)$

Entonces: Todas las secuencias en $C(w_0)$ son de largo $\Omega(n)$

Conclusión: Con entrada w_0, la máquina M toma tiempo $\Omega(n^2)$

- Puesto que M tiene que generar $\frac{n}{2}$ secuencias de estados de largo $\Omega(n)$

¿Qué hemos aprendido?

Al diseñar un algoritmo debemos considerar el modelo de computación sobre el cual será implementado.

- ¿Qué operaciones podemos realizar en el modelo?

¿Qué hemos aprendido?

Al diseñar un algoritmo debemos considerar el modelo de computación sobre el cual será implementado.

- ¿Qué operaciones podemos realizar en el modelo?

Al analizar la complejidad computacional de un algoritmo también debemos considerar el modelo de computación.

- ¿Qué operaciones consideramos al analizar la complejidad de un algoritmo?

¿Qué hemos aprendido?

En general, el análisis de la complejidad de un algoritmo se realiza considerando un tipo particular de entradas.

- El peor caso es muy utilizado, pero también podemos considerar el caso promedio

¿Qué hemos aprendido?

En general, el análisis de la complejidad de un algoritmo se realiza considerando un tipo particular de entradas.

- El peor caso es muy utilizado, pero también podemos considerar el caso promedio

Al estudiar un problema debemos tener en cuenta que cotas inferiores se puede demostrar para su complejidad

- Estas cotas inferiores dependen del modelo de computación considerado

¿Qué consideraciones adicionales debemos tener?

Debemos considerar modelos de computación que representen el funcionamiento de una arquitectura de computadores.

- Por ejemplo, debemos consider acceso directo a los datos y la diferencia de costo entre el uso de memoria principal y secundaria
- Las máquinas de Turing pueden no ser apropiadas en este sentido. ¿Por qué las usamos entonces?

¿Qué consideraciones adicionales debemos tener?

Debemos considerar modelos de computación que representen el funcionamiento de una arquitectura de computadores.

- Por ejemplo, debemos consider acceso directo a los datos y la diferencia de costo entre el uso de memoria principal y secundaria
- Las máquinas de Turing pueden no ser apropiadas en este sentido. ¿Por qué las usamos entonces?

Vamos a considerar dos ejemplos que nos servirán para ilustrar los puntos anteriores: ordenación y la multiplicación de números enteros