Erick Bernal Márquez 317042522
José Alejandro Pérez Márquez 310109800
Tania Michelle Rubí Rojas 315121719 **Tarea 2**

Ejercicio 1

Sea M una Máquina de Turing cuyo tiempo de ejecución está dado por:

$$f_m(n) = 2n^3(n+3)(n-4)$$

¿Cuáles de las siguientes afirmaciones son verdaderas? Justifica tu repuesta.

a. $f_M \in O(n)$

SOLUCIÓN: Sean $f(n) = 2n^3(n+3)(n-4)$ y $g(n) = n$. Para que la afirmación fuera cierta necesitaríamos que existieran constantes positivas c y n_0 tal que $f(n) \leq cg(n)$, para toda $n \geq n_0$. Así,

$$\begin{aligned}
f(n) &\leq cg(n) & \\
2n^3(n+3)(n-4) &\leq cn & \text{definición de } f \text{ y } g \\
\frac{2n^3(n+3)(n-4)}{n} &\leq \frac{cn}{n} & \text{dividimos entre } n \text{ en ambos lados} \\
\color{red}{2n^2(n+3)(n-4)} &\color{red}{\leq c} & \text{álgebra}
\end{aligned}$$

Pero es claro que no existen constantes c y n_0 tal que hagan que la desigualdad que está en color rojo sea cierta. Por lo tanto, la afirmación es **falsa**.

b. $f_M \in O(n^6)$

SOLUCIÓN: Sean $f(n) = 2n^3(n+3)(n-4)$ y $g(n) = n^6$. Esto significa que existen constantes positivas c y n_0 tal que $f(n) \leq cg(n)$, para toda $n \geq n_0$. Así,

$$\begin{aligned}
f(n) &\leq cg(n) & \\
2n^3(n+3)(n-4) &\leq cn^6 & \text{definición de } f \text{ y } g \\
\frac{2n^3(n+3)(n-4)}{n^3} &\leq \frac{cn^6}{n^3} & \text{dividimos entre } n^3 \text{ en ambos lados} \\
2(n+3)(n-4) &\leq cn^3 & \text{álgebra} \\
2n^2 - 2n - 24 &\leq cn^3 & \text{álgebra}
\end{aligned}$$

Sabemos que $n^2 \leq n^3$, por lo que $2n^2 \leq 2n^3$. Además, sabemos que $n \leq n^3$, por lo que $2n \leq 2n^3$. Así,

$$2n^3 + 2n^3 = 4n^3 \qquad \forall n \geq 1$$

Por lo tanto,

$$2n^2 - 2n - 24 \leq 4n^3 \qquad \forall n \geq 1$$

Lo que significa que $c = 4$ y $n_0 = 1$. Por consiguiente, la afirmación es **verdadera**.

c. $f_M \in O(\frac{n^5}{20})$

Solución: Sean $f(n) = 2n^3(n+3)(n-4)$ y $g(n) = \frac{n^5}{20}$. Esto significa que existen constantes positivas c y n_0 tal que $f(n) \leq cg(n)$, para toda $n \geq n_0$. Así,

$$
\begin{aligned}
f(n) &\leq cg(n) && \\
2n^3(n+3)(n-4) &\leq c\left(\frac{n^5}{20}\right) && \text{definición de } f \text{ y } g \\
2n^3(n+3)(n-4) &\leq \frac{cn^5}{20} && \text{multiplicación de fracciones} \\
20(2n^3(n+3)(n-4)) &\leq 20\left(\frac{cn^5}{20}\right) && \text{multiplicamos por 20 en ambos lados} \\
40n^3(n+3)(n-4) &\leq cn^5 && \text{álgebra} \\
\frac{40n^3(n+3)(n-4)}{n^3} &\leq \frac{cn^5}{n^3} && \text{dividimos entre } n^3 \text{ en ambos lados} \\
40(n+3)(n-4) &\leq cn^2 && \text{álgebra} \\
40n^2 - 40n - 480 &\leq cn^2 && \text{álgebra}
\end{aligned}
$$

Sabemos que $n \leq n^2$, por lo que $40n \leq 40n^2$ para toda $n \geq 1$. Así,

$$40n^2 - 40n - 480 \leq 40n^2 \qquad \forall n \geq 1$$

Lo que significa que $c = 40$ y $n_0 = 1$. Por lo tanto, la afirmación es **verdadera**.

d. $f_M \in \Theta(n^6)$

Solución: Sabemos que para cualesquiera dos funciones $f(n)$ y $g(n)$ tenemos que $f(n) = \Theta(g(n))$ si y sólo si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$. Por el inciso *b* tenemos que $2n^3(n+3)(n-4) = O(n^6)$, así que para determinar si la afirmación es falsa o no, nos falta verificar si $2n^3(n+3)(n-4) = \Omega(n^6)$.

Sean $f(n) = 2n^3(n+3)(n-4)$ y $g(n) = n^6$. Para que $2n^3(n+3)(n-4) = \Omega(n^6)$ fuera cierto necesitaríamos que existieran constantes positivas c y n_0 tal que $cg(n) \leq f(n)$, para toda $n \geq n_0$. Así,

$$
\begin{aligned}
cg(n) &\leq f(n) && \\
cn^6 &\leq 2n^3(n+3)(n-4) && \text{definición de } f \text{ y } g \\
\frac{cn^6}{n^3} &\leq \frac{2n^3(n+3)(n-4)}{n^3} && \text{dividimos entre } n^3 \text{ en ambas lados} \\
cn^3 &\leq 2(n+3)(n-4) && \text{álgebra} \\
cn^3 &\leq 2n^2 - 2n - 24 && \text{álgebra}
\end{aligned}
$$

Pero sabemos que $n^3 \geq n^2$, por lo que $kn^3 \geq kn^2$ para cualquier $k \geq 1$. Lo que significa que no existen constantes c y n_0 tal que hagan que la desigualdad que está en color rojo sea cierta. Por lo tanto, $2n^3(n+3)(n-4) \neq \Omega(n^6)$.

Por consiguiente, $f_M \notin \Theta(n^6)$.

Ejercicio 2

Da una descripción general de una máquina de Turing para cada uno de los siguientes lenguajes. Determina la función de complejidad de tiempo en cada caso.

a. $\{w \in \{0,1,\#\}^* \mid w$ contiene el triple de 0's que 1's $\}$

SOLUCIÓN:

- Descripción general

  Sea w una cadena de entrada para la Máquina M.

  1. Si w no contiene ceros ni unos, entonces **acepta** (pues por vacuidad se cumple que contiene el triple de ceros que de unos).
  2. Recorremos a w hasta encontrar un cero. Si encontramos un ␣ (espacio en blanco) antes de encontrar el cero entonces **rechaza** (pues habremos llegado al final de la cadena). En caso contrario:
     2.1. Reemplazamos el cero por una X y buscamos otro cero a partir de donde nos quedamos. Si encontramos ␣ (espacio en blanco) antes de encontrar el cero, entonces **rechaza** (pues habremos llegado al final de la cadena). En caso contrario:
       2.1.1. Reemplazamos el cero por una X y buscamos otro cero a partir de donde nos quedamos. Si encontramos ␣ (espacio en blanco) antes de encontrar el cero, entonces **rechaza** (pues habremos llegado al final de la cadena). En caso contrario:
         2.1.1.1 Reemplazamos el cero por una X y regresamos al inicio de la cadena w.
  3. Recorremos w hasta encontrar un uno. Si encontramos un ␣ (espacio en blanco) antes de encontrar el uno, entonces **rechaza**. En caso contrario, reemplazamos el uno por una Y y regresamos al inicio de la cadena w.
  4. Repetimos 2 y 3 hasta terminar con los ceros de w.
  5. Recorremos a la cadena w. Si encontramos un uno, entonces **rechaza**. En caso contrario, **acepta**.

Mostraremos un breve ejemplo del algoritmo anterior con una cadena $w = 01000010$. Veamos que w tiene el triple de ceros que de unos.

$$\begin{array}{r} 01000010 \\ X1000010 \\ X1X00010 \\ X1XX0010 \\ XYXX0010 \\ XYXXX010 \\ XYXXXX10 \\ XYXXXX1X \\ XYXXXXYX \\ \text{ACEPTA} \end{array}$$

- Complejidad

  La complejidad de esta máquina es de $O(n^2)$, donde n es la longitud de la cadena de entrada w. En el proceso para marcar y buscar un cero o un uno en w, en el peor caso, tenemos que recorrer toda la cadena w. Así que si $|w| = n$, entonces este proceso nos toma n pasos, lo que implica que nos toma $O(n)$. Esto lo debemos de multiplicar por el número de veces que debemos realizar dicho proceso, esto es, una vez por cada elemento de w. Así que como estamos marcando y buscando en tiempo $O(n)$ una cantidad de n veces, entonces tenemos que

  $$O(n) \cdot O(n) = O(n^2)$$

  es el tiempo total de ejecución.

b. $\{w \in \{a, b, c\}^* \mid w$ contiene el número de a's en $w >$ número de b's en $w \geq$ número de c's en w $\}$

SOLUCIÓN:

- Descripción general

  Sea w una cadena de entrada para la Máquina M.

  1. Si w es la cadena vacía, entonces **acepta** (pues por vacuidad se cumple que el número de a's es mayor que el número de b's, y que este último número es mayor o igual al número de c's).
  2. Si w no contiene b's o c's (es decir, es una cadena que contiene puras a's), entonces **acepta** (pues el número de a's es mayor al número de b's, que es cero, y particularmente también se cumple que el número de b's es igual al número de c's).

3. Recorremos a w hasta encontrar una b y la reemplazamos por una X. Regresamos al inicio de w, buscamos una c y la reemplazamos por una Y.
   - Si después de reemplazar una b por una X ya no encuentro una c en toda la cadena, entonces nos movemos al inicio de w y vamos al paso 5.
   - Si no encuentro ninguna b para reemplazarla por una X, pero sí encuentro una c para reemplazar por una Y, entonces **rechaza**.
   - Si ya no encuentro ninguna b para reemplazar ni ninguna c para reemplazar, entonces nos movemos al inicio de w y vamos al paso 5.
4. Vamos al inicio de la cadena w y repetimos 3 hasta terminar con todas las b's y todas las c's.
5. Recorremos a w hasta encontrar una a y la reemplazamos por una Z. Regresamos al inicio de w, buscamos una b o una X y la reemplazamos por una W.
   - Si después de reemplazar una a ya no encuentro una b o una X en toda la cadena, entonces **acepta**. En otro caso, **rechaza**.
6. Vamos al inicio de la cadena w y repetimos 5 hasta terminar con todas las a's y todas las b's.

Mostraremos un breve ejemplo del algoritmo anterior con una cadena $w = cbaaba$. Veamos que w cumple la propiedad.

$$\begin{aligned} cbaaba \\ YXaaba \\ YXaaXa \\ YXZaXa \\ YWZaXa \\ YWZZXa \\ YWZZWa \\ YWZZWZ \\ \text{ACEPTA} \end{aligned}$$

- Complejidad

  La complejidad de esta máquina de es de $O(n^3)$, donde n es la longitud de la cadena de entrada w. En el proceso de reemplazar y buscar una a, b o c, en el peor de los casos, tenemos que recorrer toda la cadena w una cantidad de n veces. En ese caso, si $|w| = n$, entonces este proceso nos toma $n \cdot n = n^2$ pasos, lo que implica que nos toma $O(n^2)$. Esto lo debemos de multiplicar por el número de veces que debemos realizar dicho proceso, esto es, una vez por cada elemento de w. Así que como estamos reemplazando y buscando en tiempo $O(n^2)$ una cantidad de n veces, entonces tenemos que

$$O(n^2) \cdot O(n) = O(n^3)$$

es el tiempo total de ejecución.

Ejercicio 3

Considera la Máquina de Turing definida por:

$$M = (\{q_s, q_1, q_2, q_a, q_r\}, \{0.1\}, \{0, 1, \_\}, \delta, q_s, q_a, q_r)$$

Describe el lenguaje L_M para cada una de las siguientes definiciones de δ. Para cada inciso:

- Justifica tu respuesta agregando un par de ejecuciones (al menos una de aceptación y una de rechazo) para cada inciso.
- Da la función $f_M(n)$ del tiempo de ejecución de la máquina

a. $\delta(q_0, 0) = (q_1, 1, \rightarrow); \delta(q_1, 1) = (q_2, 0, \leftarrow)$
   $\delta(q_2, 1) = (q_0, 1, \rightarrow); \delta(q_1, \_) = (q_a, \_, \rightarrow)$

b. $\delta(q_0, 0) = (q_1, 1, \rightarrow); \delta(q_1, 1) = (q_0, 0, \rightarrow)$
   $\delta(q_1, \_) = (q_a, \_, \leftarrow)$

c. $\delta(q_0, 0) = (q_0, \_, \rightarrow); \delta(q_0, 1) = (q_1, \_, \rightarrow)$
   $\delta(q_1, 1) = (q_1, \_, \rightarrow); \delta(q_1, \_) = (q_a, \_, \leftarrow)$

* Considera que las reglas (transiciones) que no se indican de manera explícita llevan al estado de rechazo.

a. Pondremos de manera gráfica la MT

   A primera vista notemos que no acepta la cadena vacia ϵ, acepta el 0 por sí solo y en cuanto leemos más símbolos tenemos que pasar por más estados. Haciendo múltiples ejecuciones nos daremos cuentas que estos estados sirven para verificar que la cadena continue con puros 1's, es decir que no haya 0's después.

   Concluimos que acepta cadenas de la forma 01^*

   En el peor caso tendremos que recorrer toda la cadena, pero notemos que lo hace de un forma peciluar. Lee el primer simbolo y se regresa, luego lee dos simbolos y se regresa uno, lee dos y se regresa 1, así hasta aceptar la cadena. Siguiendo esta noción si aceptamos una cadena de 1 símbolo la MT hace 2 paso, si aceptamos una cadena de 2 simbolos hacemos 5 pasos, de 3 símbolos 8 pasos, de 4 11 pasos.

a.jpg

$1 \rightarrow 1$ pasos

$2 \rightarrow 5$ pasos

$3 \rightarrow 8$ pasos

$4 \rightarrow 11$ pasos

$5 \rightarrow 14$ pasos

$6 \rightarrow 17$ pasos

Esta tendencia sigue la ecuación $y = 3n - 1$. Así deducimos que la función en tiempo de ejecución de la MT $f_M(n)$ es $3n - 1$ donde n es el tamaño de la entrada. Además está en el orden de $O(n)$.

b. Pondremos de manera gráfica la MT

Notemos que no acepta la cadena vacia, pero sí un 0 sólo, en caso de que el siguiente simbolo sea un 1 el siguiente simbolo necesariamente debe de ser 0 para poder aceptar la cadena, si no es 1 lo rechaza; es decir, acepta las cadenas de la forma (01)*0

Siempre nos movemos a la derecha salvo el último paso para verificar que acabamos de recorrer la cadena, por lo cual $f_M(n) = n + 1$ donde n es el tamaño de la entrada, la cual se encuentra en el orden $O(n)$.

c. Pondremos de manera gráfica la MT

De igual manera no acepta la cadena vacia, acepta cualquier cantidad de 0's pues se mantiene en un *ciclo* con el estado q_0, luego es necasario aceptar un 1 para avanzar al ares.jpeg

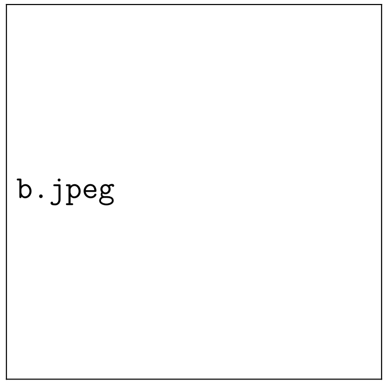
siguiente estado, de la misma forma se *cicla* en el estado q_1 para cualquier cantidad de 1. Por lo cual acepta cadenas de la forma 0^*11^*

De igual manera que el anterior en el peor caso tiene que recorrer toda la cadena, por lo cual $f_M(n) = n + 1$ donde n es el tamaño de la entrada. Orden de $O(n)$

Ejercicio 4

Sea $L = \{a\#b|a, b$ codifican números enteros positivos en binario y máx(a,b) = $a\}$. Da la descripción completa de una máquina de Turing (incluyendo estados, alfabeto y función de transición) que acepte L. Ejemplifica la ejecución de la máquina propuesta mostrando las configuraciones que se producen para las siguientes entradas:

- $\lfloor 7 \rfloor \# \lfloor 5 \rfloor \longrightarrow 111\#101$
- $\lfloor 2 \rfloor \# \lfloor 4 \rfloor \longrightarrow 10\#100$



- $\lfloor 1 \rfloor \# \lfloor 1 \rfloor \longrightarrow 1\#1$

Calcula la complejidad de tiempo, en el peor caso, en función del tamaño de la entrada.

Mostraremos la lógica detrás de la construcción de la Maquina de Turing para hacer más fácil su entendiemiento y analisis.

Primero debemos verificar el tamaño de la entrada, nos interesa que las cadenas a la derecha e izquierda de # tengan el mismo tamaño o bien el primer número sea mayor que el otro en cuanto a longitud; si el segundo número es mayor entonces ya podemos asegurar que la cadena no pertenece al lenguaje. Esto es lo que hace la primera parte de la MT.

La maquina recorrer toda la cadena y después se regresa, vuelve a recorrerla pero está vez con un símbolo menos, se regresa otra vez y de nuevo la recorre con otro símbolo menos, es decir sigue el clásico patrón $n + (n-1) + (n-2) + (n-3) + ...$ (no estamos diciendo que esta sea su fn de complejidad), por lo que su complejidad está en el orden de $O(n^2)$.

Notemos que utilizamos dos 4 símbolos estras: A, B, P, F esto son solamente auxiliares, pues tuvimos que modificar la cadena para llevar el conteo de la longitud, lo que hacen estas letras especiales es "recordar" el símbolo que tenía la cadena y volverlos a poner.

Es de lo que se encarga el estdo q_{16}, vemos que lo hace en un estado y siempre a la izquierda hasta llegar al principio de la cadena ya "recordada". Esto lo hace en un solo recorrido por lo cual tiene complejidad $O(n)$

Una vez que aseguramos ambas cadenas al lado de # tienen la misma longitud o bien el primer numero es mayor solo queda comparar en sí mismo los números.

Observemos que de q_0 a q_1 y q_2 hacemos una distinción dependiendo si leemos un 0 o 1, esto es basicamente para hacer la distinción del número mayor.
Si leimos primero un 1 y luego, después de pasar el #, lo comparamos con otro 1 entonces la cadena toda´via tiene probabilidades de que sea aceptada, de la misma si después leimos un 0. Pero si leímos primero un 0 y luego, después de pasar el #, leemos un 1 entonces en automatico podemos descartar la cadena, pues significa que tenemos un cadena del estilo X01#Y11 y sabemos que 01 es menor que 11. Esta lógica es basicamente lo que hacen los estados $q_0, q_1, q_2, q_3, q_4, q_6$.

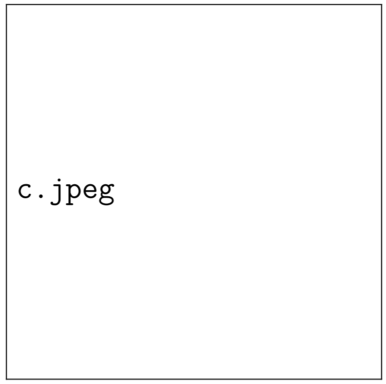
bres.jpeg

Lo que hace el estado q_7 es verificar que el último símbolo de la cadena haya cambiado a Y, así aseguramos que se terminó de leer la cadena.

Al igual que la primera parte de la MT este procedimiento recorre toda la cadena, luego toda la cadena menos un símbolo, luego toda la cadena menos dos simbolos, es decir tiene complejidad $O(n^2)$. La suma de las complejidades es $O(n^2) + O(n) + O(n^2)$ las cuales por propiedades de los ordenes es complejidad $O(n^2)$.

Así definimos la MT en su totalidad como $MT = (Q, \Sigma, \Gamma, \delta, q_5, q_9, q_r)$, donde
$Q = \{q_0, q_1, ..., q_16\}$
$\Sigma = \{0, 1, \#\}$
$\Gamma = \{0, 1, \#, \_, A, B, F, P, X, Y\}$
q_5 es el estado incial
q_9 es el estado de aceptación
q_r son todos los demás estados que no son estados de aceptación. A δ lo definimos con la represetnación gráfica de la MT, ya que una tabla sería más fácil de leer. De igual manera consideramos que las reglas que no se indican de manera explicita se consideran como rechazo.
La Maquina de Turing completa es:

Además ponemos algunas entradas.



Ejercicio 5

Investiga en qué son las funciones de tiempo "construibles" (Time-constructible functions).

- Elige dos de las siguientes funciones y demuestra que son de tiempo construibles:
    a. $2nlog(n)$
    b. $3n^2$
    c. 3^n
- Da otros ejemplos (diferentes a los del inciso anterior) de funciones de tiempo construible
- Da un par de ejemplos de funciones que no son de tiempo construible. Justifica por qué no lo son.

Ejercicio 6

En los programas de las máquinas de Turing hay operaciones qué son frecuentes de utilizar como pasos intermedios, una de estas operaciones es llamada "desplazamiento".

Un desplazamiento consiste en mover todo el contenido de la cinta una posición a la derecha o la izquierda, a partir de una posición específica, y luego regresar la cabeza de la máquina a dicha posición.

a. Indica con todo detalle cómo realizar esta operación.

`cres.jpeg`

b. Considera la siguiente configuración: $110q101$ donde q es el estado inicial para ejecutar la operación de desplazamiento. De acuerdo a la descripción del inciso anterior, muestra todas las configuraciones hasta completar la operación.

c. ¿Cuál es la complejidad de esta operación?

d. Explica o justifica por qué esta operación puede ser útil.

Referencias

Notación asintótica: `https://cs.famaf.unc.edu.ar/~hoffmann/md18/04.html`
Software utilizado para la creación y ejecución de las MT: `https://www.jflap.org/`

41.jpeg

42.jpeg

43.jpeg

4todo.jpeg

`4todores.jpeg`