Facultad de Ciencias
UNAM

Complejidad computacional

Tarea 2 "Máquinas de Turing"

Victor Hugo Gallegos Mota - 316160456
Jose Demian Jimenez Salgado - 314291707
Luis Alberto Hernández Aguilar - 314208682

September 22, 2022

Table of Contents

1 Ejercicio 1

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

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

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

Inciso A

$$f_M(n) \in O(n)$$

RESPUESTA Esto es falso

Primero lo que haremos para mas comodidad sera desarollar el polinomio

$$f_M(n) = 2n^3(n+3)(n-4)$$
$$f_M(n) = 2n^3(n^2 - n - 12)$$
$$f_M(n) = 2n^5 - 2n^4 + 24n$$

Ahora tomaremos cada termino del polinomio y los acotaremos de acuerdo a la definicion de O(n) con g(n)=n
a) $n^5 \leq$ n $\forall$n 1
$\Rightarrow 2n^5 \leq$ 2n $\forall$n 1
En este caso hay contradiccion ya que $2n^5$ no es menor a 2n por lo tanto no es posible acotar con O(n)
$\Rightarrow 2n^5 \leq$ 2n $\forall$n 1
por lo tanto f$_m(n) \notin$ O(n)

Inciso B

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

RESPUESTA Esta es verdadera

Ahora tomaremos cada termino del polinomio y los acotaremos de acuerdo a la definicion de O(n) con g(n)=n^6
a) n$^5 \leq$ n$^6\forall$n 1
$\Rightarrow 2n^5 \leq 2n^6$ $\forall$n 1
b) n$^4 \leq$ n$^6\forall$n 1
$\Rightarrow -2n^4 \leq -2n^6$ $\forall$n 1

c)n$^3 \leq$ n$^6\forall$n 1
$\Rightarrow 24n^3 \leq 24n^6$ $\forall$n 1

$\Rightarrow 2n^6$-$2n^6$+$24n^6$= $24n^6$ $\forall$n 1
Por lo tanto c=24 y n$_0 = 1$
0 <=f(n)<=cn^6

Inciso C

$$f_M(n) \in O(n^5/20)$$

RESPUESTA Este inciso es falso

Ahora tomaremos cada termino del polinomio y los acotaremos de acuerdo a la definicion de O(n) con g(n)=$\frac{n^5}{20}$
a) n^5 $\leq \frac{n^5}{20}$ $\forall$n <= 1
En este caso hay contradiccion ya que n^5 no es menor a $\frac{n^5}{20}$ por lo tanto no es posible acotar con O($\frac{n^5}{20}$)

Inciso D

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

RESPUESTA Tomando en cuenta que $f_M(n) = 2n^3(n+3)(n-4)$, podemos ver que $f_M(n)$ es una función polinomial de grado 6, por lo tanto, es una función polinomial de grado n^6. Ahora tomaremos cada termino del polinomio y los acotaremos de acuerdo a la definicion de Θ: n^6.

- $2n^3(n+3)(n-4) \leq 2n^6$
  Es decir 0 < c1 * n^6 $\leq 2n^3(n+3)(n-4) \leq$ c2 * n^6

$\Longrightarrow 0 < c1 \leq$ -24/n$\leq$c2
$\Longrightarrow c1 = 24/23$ y cuando n $\longrightarrow \infty$
$\Longrightarrow c2 = 0$

Por lo tanto cumple para, $f_M(n) \in \Theta(n^6)$

Complejidad en tiempo es $\Theta(n^6)$

2 Ejercicio 2

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

Inciso A

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

RESPUESTA

1. Escanear la cinta y marcar el primer 1 que no haya sido marcado. Si no se encuentran 1 sin marcar, ir al ultimo paso
2. Explorar la cinta hasta que encuentre un 0 sin marcar y marcarlo. Si no se encuentran 0, entonces rechazar.
3. Escanear la cinta una vez más hasta que encuentre un 0 sin marcar y marcarlo. Si no se encuentran 0, entonces rechacer.
4. Mover la cabeza hacia atrás al comienzo de la cinta y volver al primer paso.
5. Mover la cabeza de regreso al comienzo de la cinta. Escanea la cinta en busca de ceros sin marcar. Si no encuentra ninguno, aceptarlo.De lo contrario rechazarlo.

Visto de otra forma las cadenas aceptadas se verian de la siguiente forma

- XXX#YYYYYYYYY
- X#YYY

Complejidad
La Complejidad en este caso es de O(n^2)
En este caso n es la longitud de la cadena. Ir marcando
cada 0,1 en el peor de los casos seria recorrer en n pasos
$\Rightarrow$ O(n)*O(n)=O(n^2)

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

RESPUESTA

1. Sea w una entrada para la maquina M
2. Si w contiene al menos una a pero no tiene b´s ni c´s aceptara ya que en este caso w tiene mas a's que b's y c's.
3. Si encontramos una a , la marcamos con una X.
· Si encontramos a esta segunda a la reemplazamos con una X y regresamos al inicio de w.
4. Buscamos una b en w, si encontramos antes $\sqcup$ aceptamos siempre y cuando halla a's antes.
· Si encontramos esa b la reemplazamoscon una Y y regresamos al inicio de w.
5. Buscamos una b en w, si encontramos antes $\sqcup$ aceptamos siempre y cuando halla a's antes, alguna b o ninguna.
· Si encontramos esa c la reemplazamoscon una Z y regresamos al inicio de w.
6. Se repiten 3,4 y 5 respectivamente
7.Final: ya hay mas a's estrictamente que b's y c's entonces aceptamos, caso contrario rechazamos

Visto de otra forma las cadenas aceptadas se verian de la siguiente forma

- XXXBBBBZZZZa
- XYZa

Complejidad
La Complejidad en este caso es de O(n^3)
En este caso n es la longitud de la cadena. Ir marcando
cada a,b,c en el peor de los casos seria recorrer en n pasos
a w. Suponiendo que tenemos que llegar al final de w.
Y como nuestro alfabeto tiene tres elementos entonces realizar
dicho proceso nos costaria multiplicar 3 veces a n ya que estamos
buscando los elementos n veces.
$\Rightarrow$ O(n)*O(n)*O(n)=O(n^3)

3 Ejercicio 3

Considera la Máquina de Turing definida por:
M={{q$_s, q_1, q_2, q_a, q_r\}, \{0,1\}, \{0,1,\sqcup\}, \delta, q_0, q_a, q_r\}$
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 aceptacion y una de rechazo)
para cada inciso.
a)$\delta(q_0, 0) = (q_1, 1, \rightarrow); \delta(q_1, 1) = (q_1, 1, \rightarrow)$
$\delta(q_2, 1) = (q_0, 1, \rightarrow); \delta(q_1, \sqcup) = (q_a, \sqcup, \rightarrow)$
Con esta definicion la maquina solo acepta cadenas
de la forma 0(111)*
aqui la ejecución de 0111
$\delta(q_0, 0) = (q_1, 1, \rightarrow) 1111 \sqcup$

$\delta(q_1, 1) = (q_2, 0, \leftarrow)1011\sqcup$
$\delta(q_2, 1) = (q_0, 1, \rightarrow)1011\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow)1111\sqcup$
$\delta(q_1, 1) = (q_2, 0, \leftarrow)1101\sqcup$
$\delta(q_2, 1) = (q_0, 1, \rightarrow)1101\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow)1111\sqcup$
$\delta(q_1, 1) = (q_2, 0, \leftarrow)1110\sqcup$
$\delta(q_2, 1) = (q_0, 1, \rightarrow)1110\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow)1111\sqcup$
$\delta(q_1, \sqcup) = (q_a, \sqcup, \rightarrow)$
por lo tanto 0111 es aceptada
Ahora probemos con 0101
$\delta(q_0, 0) = (q_1, 1, \rightarrow)1101\sqcup$
$\delta(q_1, 1) = (q_2, 0, \leftarrow)1001\sqcup$
$\delta(q_2, 1) = (q_0, 1,)1001\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow)1101\sqcup$
$\delta(q_0, 0) = rechazo$
Como llegamos a la transición trivial entonces
hay rechazo.
Por lo tanto 0101 es rechazada por la maquina bajo esta
definicion.

b)$\delta(q_0, 0) = (q_1, 1, \rightarrow); \delta(q_1, 1) = (q_0, 0, \rightarrow)$
$\delta(q_1, \sqcup) = (q_a, 1, \leftarrow);$
Con esta definicion la maquina solo aceptaria cadenas
en donde cada cero va seguido de un 1 y solo pueden terminar
en cero y empezar con 0
aqui algunas ejecuciones de la maquina:
Primero usaremos la cadena 01010
01010⊔
$\delta(q_0, 0) = (q_1, 1, \rightarrow); 11010\sqcup$
$\delta(q_1, 1) = (q_0, 0, \rightarrow); 10010\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow); 10110\sqcup$
$\delta(q_1, 1) = (q_0, 1, \rightarrow); 10110\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow); 10111\sqcup$
$\delta(q_1, \sqcup) = (q_a, \sqcup, \leftarrow); 11010\sqcup$
Por lo tanto aceptamos 01010
Ahora probemos con 0100
$\delta(q_0, 0) = (q_1, 1, \rightarrow); 1100\sqcup$
$\delta(q_1, 1) = (q_0, 0, \rightarrow); 1000\sqcup$
$\delta(q_0, 0) = (q_1, 1, \rightarrow); 1010\sqcup$
$\delta(q_0, 0) = rechazo$
ya que se llega a la transición
trivial.
Por lo que 0100 no es aceptada con esta
definicion.

c) $\delta(q_0, 0) = (q_0, \sqcup, \rightarrow); \delta(q_0, 1) = (q_1, \sqcup, \rightarrow)$
$\delta(q_1, 1) = (q_a, \sqcup, \leftarrow); \delta(q_1, \sqcup) = (q_a, \sqcup, \leftarrow)$
En esta definicion se aceptan las cadenas que al menos deben
de tener un cero para tener unos en la cadena, los ceros
deben de ir antes de los unos
probaremos con 0011
$\delta(q_0, 0) = (q_0, \sqcup, \rightarrow) \sqcup 011\sqcup$
$\delta(q_0, 0) = (q_0, \sqcup, \rightarrow) \sqcup\sqcup 11\sqcup$
$\delta(q_0, 1) = (q_1, \sqcup, \rightarrow) \sqcup\sqcup\sqcup 1\sqcup$

$\delta(q_1, 1) = (q_1, \sqcup, \rightarrow) \sqcup \sqcup \sqcup \sqcup \sqcup$
$\delta(q_1, \sqcup) = (q_a, \sqcup, \leftarrow)$
por lo tanto aceptamos 0011
Ahora probaremos con 1010
$\delta(q_0, 1) = (q_1, \sqcup, \rightarrow) \sqcup 010$
$\delta(q_1, 0) = rechazo$
Aqui como se llego a la transición trivial la cadena es
rechazada por la maquina.
complejidad:
Aqui nuestra maquina recibe cadenas de tamano n asi que pues
haremos n recorridos para el analisis de la cadena, en este
caso nuestra maquina puede recibir 0's,1's o $\sqcup$
asi que en el peor de los casos la maquina un total de n
lugares de la cadena hasta llegar al final y encontrar una
$\sqcup$
Esto nos toma tiempo O(n) y como tenemos tres elementos que
puede leer nuestra cadena entonces tendriamos que multiplicar
O(n) tres veces: O(n)*O(n)*O(n)=O(n^3).
Por lo que la complejidad de la maquina de Turing es O(n^3).

4 Ejercicio 4

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

- $\lfloor 7 \rfloor \# \lfloor 5 \rfloor \rightarrow 111\#101$
- $\lfloor 2 \rfloor \# \lfloor 4 \rfloor \rightarrow 110\#100$
- $\lfloor 1 \rfloor \# \lfloor 1 \rfloor \rightarrow 1\#1$

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

  RESPUESTA M = ({$q_0 ... q_16, q_a, q_r\}, \{0,1\}, \{0, 1\#, A, B, X, Y, \bigsqcup\}) \delta, q_0, q_a, q_r)$

  - $\lfloor 7 \rfloor \# \lfloor 5 \rfloor \rightarrow 111\#101$
    - $q_0 111\#101$
    - $q_1 11\#101$
    - $X1q_1 1\#101$
    - $X11q_1\#101$
    - $X11\#q_2 101$
    - $X11q_3\#A01$
    - $X1q_3 1\#A01$
    - $Xq_3 11\#A01$
    - $q_3 x11\#A01$
    - $Xq_5 11\#A01$
    - $XAq_6 1\#A01$
    - $XA1q_6\#A01$
    - $XA1\#q_7 A01$
    - $XA1\#Aq_7 01$
    - $XA1q_4\#AB1$
    - $Xq_3 A1\#AB1$
    - $XAAq_6\#AB1$
    - $XAA\#Aq_7 B1$
    - $XAA\#ABq_7 1$

        - $XAA\#q_4ABA$
        - $XAAq_4\#ABA$
        - $XAAq_5\#ABA$
        - $XAA\#Aq_8BA$
        - $XAA\#ABq_8A$
        - $XAA\#ABA_q8$
        - $XAA\#ABq_9A$
        - $XAA\#q_9ABA$
        - $XAAq_9\#ABA$
        - $XAq_9A\#ABA$
        - $q_9XAA\#ABA$
        - $Xq_11AA\#ABA$
        - $XAA_q11\#ABA$
        - $XAA\#q_12ABA$
        - $XAq_16A\#XBA$
        - $q_16XAA\#XBA$
        - $XXq_11A\#XBA$
        - $XXA\#Xq_12BA$
        - $ACEPTACION$
    - $\lfloor 2 \rfloor \# \lfloor 4 \rfloor \rightarrow 110\#100$
        - $q_010\#100$
        - $Xq_10\#100$
        - $X0\#q_2100$
        - $X0\#Xq_300$
        - $Xq_40\#X00$
        - $XBq_5\#X00$
        - $XB\#q_6X00$
        - $XB\#XBq_70$
        - $XBq_8\#XB0$
        - $XB\#q_9XB0$
        - $XB\#XBq_{10}0$
        - $XB\#XB0q_{11}$
        - $RECHAZO$
    - $\lfloor 1 \rfloor \# \lfloor 1 \rfloor \rightarrow 1\#1$
        - $q_01\#1$
        - $Xq_1\#1$
        - $X\#q_21$
        - $Xq_3\#A$
        - $q_3X\#A$
        - $Xq_5\#A$
        - $X\#q_8A$
        - $X\#Aq_8$
        - $X\#q_9A$
        - $Xq_9\#A$
        - $q_9X\#A$
        - $Xq_{11}\#A$
        - $X\#q_{12}A$
        - $Xq_{15}\#X$
        - $q_{16}X\#X$
        - $ACEPTACION$

    Para calcular la complejidad de tiempo, debemos tener en cuenta dos factores para calcular la complejidad de tiempo, el tamaño de la entrada y el tamaño de la máquina de Turing, tomando en cuenta el numero de comparaciones en este caso el tamaño de la máquina de Turing es constante, por lo que la complejidad de tiempo es en función del tamaño de la entrada . Entonces al comparar digito a digito y se concluye que si la cadena izquierda es superior se asigna el estado de aceptacion pero se rechaza cuando la derecha es mayor entonces se comienza a comparar por la izquierda y se repite, otro caso de aceptacion es cuando tiene digitos iguales, por lo tanto el peor de los casos seria comparar numeros iguales. Visto de otro modo al traarse de dos extremos iguales se tiene la primer parte es $O(k^2) = ((2n-1/2))^2 = O(1/4(n^2-2n+1))$ Al finalizar se consulta el extremo derecho de la cinta no contenga digitos sin visitar recorriendo hasta el final de la misma y repite la comparacion de digito a digito lo que se resume en una suma de ambos pasos o sea O(k)+O(n)=O(1/2n+n) Volviendo a recorrer la cinta recorriendo k digitos por cada k digitos de la otra cinta, Asi que si se sumaran todas las secciones de la Maquina de Touring tenemos que Decimos que $T(n)$ es $O(f(n))$ si hay constantes c y n_0 tales que $T(n)$ <= $cf(n)$ para todo $n \geq n_0$. Un programa cuyo tiempo de ejecucion es $O(f(n))$ se dice que tiene una tasa de crecimiento de $f(n)$. En esta notacion $f(n)$ es una cota superior de la tasa de crecimiento de $T(n)$. entonce la funcion $O(1/2(n^2-2n+1)+1/2n+n)$ es O(n^2)

5 Ejercicio 5

Investiga en que son las funciones de tiempo “construibles” (Time-constructible functions).

— Elige dos de las siguiente funciones y demuestra que son de tiempo construibles:

a. $2n\ logn$
b. $3n^2$
c. 3^n

De acuerdo a lo visto en clase, que una función $f(n)$ sea tiempo-construible quiere decir que existe una máquina de Turing M que logra computar a $f(n)$, y para demostrarlo debemos construir una máquina M que tome como entrada a n, una cadena de n unos, y que escriba en la cinta de entrada el valor obtenido por $f(n)$.

Descripción general de una máquina de Turing M que computa a $3n^2$:

1. Sea w una entrada (cadena de n unos) para la máquina M.
2. Triplicamos w, es decir, escribimos w dos veces más.
3. Recorremos la cinta hasta encontrar un espacio en blanco (ya no hay más unos en la cinta) y escribimos un separador (cualquier símbolo) en ese espacio.
4. Regresamos (nos movemos a la izquierda) hasta encontrar un 1 antes del separador:
   - Si encontramos un 1, entonces lo cambiamos por una X y regresamos (nos movemos a la derecha) para escribir en la cinta tres veces w (la cadena inicial de n unos).
   - Si encontramos un espacio en blanco, terminamos la ejecución.
5. Repetimos el paso 4.

Descripción general de una máquina de Turing M que computa a 3^n:

1. Sea w una entrada (cadena de n unos) para la máquina M.
2. Recorremos la cinta hasta encontrar un espacio en blanco y escribimos un #.
3. Después del símbolo # escribimos una Y y tres unos.
4. Regresamos (nos movemos a la izquierda) hasta encontrar un 1 antes del símobolo #.
5. Al encontrar un 1 nos movemos una vez hacia la izquierda:
   - Si en la posición anterior encontramos un espacio en blanco, terminamos la ejecución.
   - Si no encontramos un espacio en blanco, entonces nos movemos una vez a la derecha y cambiamos el 1 que encontramos por una X.
6. Seguimos recorriendo hacia la derecha hasta el siguiente espacio en blanco para escribir una Y.
7. Después de la Y escribimos tres veces la cantidad de unos que se encuentre entre las dos últimas Y.
8. Repetimos los pasos del 4 al 7.

Como existen máquinas de Turing que computen $3n^2$ y 3^n, entonces podemos afirmar que ambas funciones son de tiempo construible.

— Da otros ejemplos (diferentes a los del inciso anterior) de funciones de tiempo construible.

Por lo visto en clase, todas las cotas son funciones de tiempo construible, entonces algunos otros ejemplos serían:

- n
- n^2
- $n\ logn$
- $n!$

— Da un par de ejemplos de funciones que no son de tiempo construible. Justifica por qué no lo son.

Sea $HALT(n)$ una función que regresa 1 si todos los problemas de longitud n se detienen con cualquier entrada, o 0 en otro caso.
$HALT(n)$ se reduce al conocido *Problema del Paro*, pues es como si quisieramos determinar si un programa termina su ejecución o no, por lo tanto la función $HALT(n)$ no es de tiempo construible y, por ende, cualquier función que dependa de $HALT(n)$, como $n + HALT(n),\ nHALT(n),\ etc.$, tampoco es de tiempo construible.

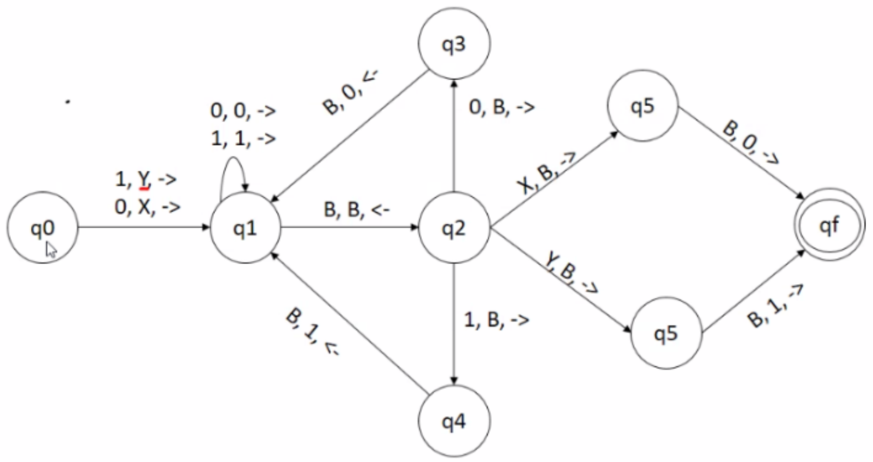
6 Ejercicio 6

En los programas de las máquinas de Turing hay operaciones que 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 a 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.

Para realizar esta operación definimos la siguiente máquina de Turing:
$M = \{Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ q_f,\ q_r\}$, donde:

- $Q = \{q_0,\ q_1,\ q_2,\ q_3,\ q_4,\ q_5,\ q_6,\ q_r\}$
- $\Sigma = \{0,\ 1\}$
- $\Gamma = \{0,\ 1,\ B,\ X,\ Y\}$
- La función de transición δ está definida de acuerdo al siguiente diagrama:



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.

Las configuraciones son las siguientes:

- $110q_0101 \implies C_1 = (q_0, 110, 101)$
- $110Yq_101 \implies C_2 = (q_1, 110Y, 01)$
- $110Y0q_11 \implies C_3 = (q_1, 110Y0, 1)$

- $110Y01q_1B \implies C_4 = (q_1, 110Y01, B)$
- $110Y0q_21 \implies C_5 = (q_2, 110Y0, 1)$
- $110Y0Bq_4B \implies C_6 = (q_4, 110Y0B, B)$
- $110Y0q_1B1 \implies C_7 = (q_1, 110Y0, B1)$
- $110Yq_20B1 \implies C_8 = (q_2, 110Y, 0B1)$
- $110YBq_3B1 \implies C_9 = (q_3, 110YB, B1)$
- $110Yq_1B01 \implies C_{10} = (q_1, 110Y, B01)$
- $110q_2YB01 \implies C_{11} = (q_2, 110, YB01)$
- $110Bq_5B01 \implies C_{12} = (q_5, 110B, B01)$
- $110B1q_f01 \implies C_{13} = (q_f, 110B1, 01)$

c. ¿Cuál es la complejidad de esta operación?

La complejidad de esta operación es $f(n) = 4n$, pues, para una entrada de tamaño n, se realizan a lo más $4n$ pasos.

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

Esta operación de desplazamiento de bits es útil para realizar operaciones matemáticas de manera eficiente, pues requiere menos cálculos que hacer para el CPU que las matemáticas convencionales ya que basta con mover unos bits, insertar o cambiar algún bit, dependiendo de la operación, y listo.

7 Referencias

- Oaks, S., & Wong, H. (2004). Java Threads: Understanding and Mastering Concurrent Programming (3.a ed.). O'Reilly Media.
- Raynal, M. (2015). Concurrent Programming: Algorithms, Principles, and Foundations (2013 ed.). Springer.
- Wellings, A. J. (2004). Concurrent and Real-Time Programming in Java (1.a ed.). John Wiley & Sons Inc.
- Méndez, P. J. T. (2006). PROGRAMACIÓN CONCURRENTE. Paraninfo.
- (S/f). Computerhope.com. Recuperado el 22 de septiembre de 2022, de https://www.computerhope.com/jargon/b/bit-shift.htm