Práctico 1 Complejidad Computacional

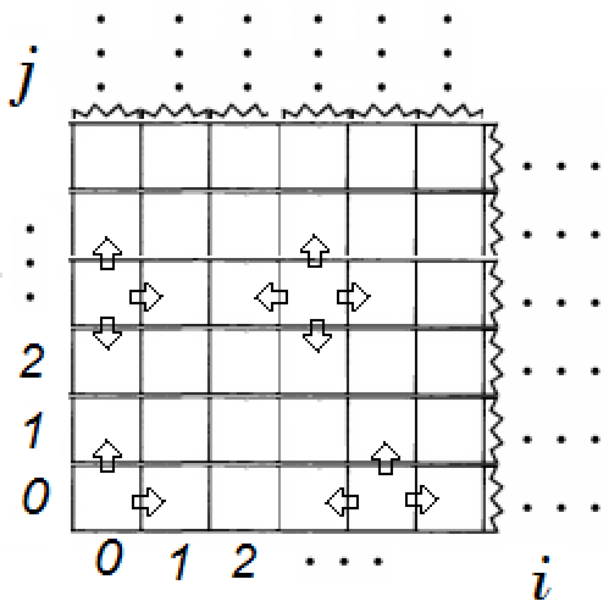
Julián Viera

jviera@fing.edu.uy
julviera44@gmail.com

19 de agosto de 2021

Agenda

Se considera una variante bidimensional de la MT denominada *TDTM* (Two Dimensional Turing Machine).
La maquina consiste en una malla semi-infinita de celdas como se muestra en la figura, y además de los movimientos a la DERECHA y a la IZQUIERDA se puede mover hacia arriba (UP) o hacia abajo (DOWN). Si la consideramos como una matriz de celdas $M(i, j)$ con $i, j \geq 0$, la TDMT no se mueve LEFT si $i = 0$ y no se mueve DOWN si $j = 0$, quedando en la misma posición.



Ejercicio 8- Planteo del problema (1.7 del Arora)

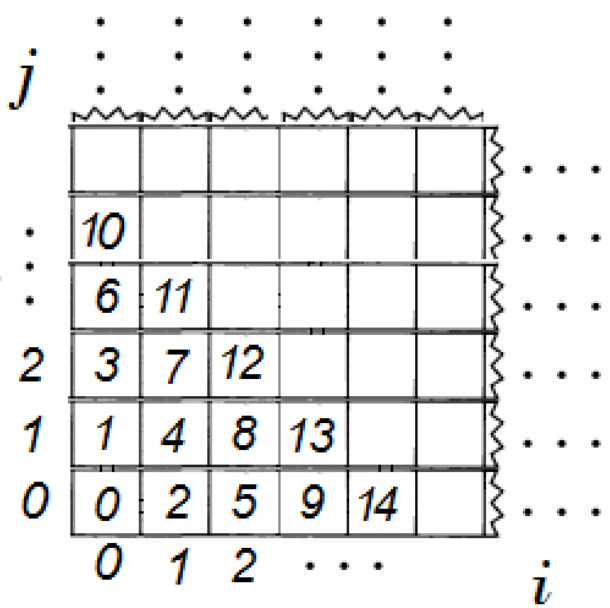
Se asume que la entrada de tamaño n se almacena en el rango de celdas que va desde $(0, 0)$ a $(n - 1, 0)$.
La TDTM comienza su ejecución con el cabezal apuntando a la celda $(0, 0)$.

El problema consiste en mostrar que si una función booleana f es computable en tiempo $T(n)$ por una *TDTM* M, entonces $f \in DTIME\,(T(n)^2)$.

Esto implica mostrar que existe una MT convencional M^* que computa f en tiempo $\mathcal{O}(T(n)^2)$.

Ejercicio 8 - una posible solución

Sea M la TDTM y M^* la MT convencional de k cintas que va a simular a M.
La idea básica es hacer un mapeo de las celdas de la malla bidimensional de M en una cinta de trabajo unidimensional de M^*. Este mapeo se hace por diagonales de la malla, como se muestra en la figura. A cada celda (i, j) de la malla de M le va a corresponder una celda k en la cinta de M^*, con $M(i, j) = M^*(k)$.



i	j	k
0	0	0
0	1	1
1	0	2
0	2	3
1	1	4
2	0	5
0	3	6
1	2	7

Ejercicio 8- Simulación de movimientos de la TDTM

Una propiedad importante que verifica la matriz de la TDTM M es que en cada diagonal a $-45°$, cada celda (i, j) de esa diagonal verifica $i + j + 1$ =número de elementos en esa diagonal=cte para esa diagonal. Se utilizará esto en lo que sigue.

Movimiento UP

En este caso el cabezal de M pasa de la celda (i, j) a la celda $(i, j + 1)$.

La MT M^* que simula a M pasa de la posicion k a una posición k'. Para determinar el valor de k' observamos que de acuerdo al mapeo propuesto, M debe mover su cabezal a la DERECHA (R)una cantidad de posiciones exactamente igual al número de celdas que tiene la diagonal a $-45°$ que pasa por (i, j), que por la observación anterior vale $i + j + 1 \Longrightarrow k' = k + i + j + 1$.

Ejercicio 8- Simulación de movimientos de la TDTM

Movimiento RIGHT

En este caso en M se pasa de la celda (i, j) a la celda $(i + 1, j)$. La MT M^* que simula a M pasa de la posicion k a una posición k'. Para determinar el valor de k' observamos que de acuerdo al mapeo propuesto, M^* debe moverse igual que en el caso UP (i+j+1 veces R) y luego dar un paso adicional a la derecha, para un total de $i + j + 2$ pasos. Tenemos así que $k' = k + i + j + 2$.

Movimiento DOWN

En este caso en M se pasa de la celda (i, j) a la celda $(i, j - 1)$ si $j > 0$, o se queda en la misma celda si $j = 0$.
De acuerdo al mapeo, puede verse que en el caso $j > 0$ M^* debe moverse a la IZQUIERDA (L),un número de celdas inferior en una unidad al número de celdas en la diagonal a $-45°$ que pasa por (i, j), lo que da un total de $i + j$ pasos. Tenemos así que $k' = k - i - j$ si $j > 0$ y $k' = k$ si $j = 0$.

Ejercicio 8- Simulación de movimientos de la TDTM

Movimiento LEFT

En este caso M se mueve de la celda (i, j) a la celda $(i-1, j)$ si $i > 0$, o se queda en la misma celda si $i = 0$.

De acuerdo al mapeo, constatamos que en el caso $i > 0$ M^* debe moverse a la IZQUIERDA (L),un número de celdas exactamente igual a la cantidad de celdas en la diagonal a $-45°$ que pasa por la celda (i, j), lo que da un total de $i + j + 1$ pasos. Tenemos así que la nueva posición del cabezal de M es $k' = k - i - j - 1$ si $i > 0$ y $k' = k$ si $i = 0$.

Ejercicio 8 - Detalles de la estructura de la MT M^*

La M^* usa 3 cintas de trabajo para calcular la cantidad de movimientos a realizar en cada paso:

1. Una cinta con un contador para la coordenada i de la celda actual de M, que verifica $|\llcorner i \lrcorner| \leq \log T(n)$.
2. Una cinta con un contador para la coordenada j de la celda actual de M, que verifica $|\llcorner j \lrcorner| \leq \log T(n)$.
3. Una cinta para calcular el desplazamiento $\Delta k = k' - k$ a partir de i y j, de acuerdo al tipo de movimiento efectuado. Se cumple que Δk puede calcularse en tiempo $\mathcal{O}(\log T(n))$.

Ejercicio 8-Tiempo de ejecución de la MT M^*

La observación crucial es que, como la *TDTM* M por hipótesis computa f en $T(n)$ pasos $\Longrightarrow$ la cantidad de celdas de la malla visitadas durante la ejecución de M es a lo sumo $T(n)$.
¿Qué implica esto? Que para cualquier celda (i, j) visitada durante la ejecución de M, se cumple $i + j + 1 \leq T(n)$. Observar que el mayor valor posible de $i + j + 1$ se obtiene cuando todos los movimientos a partir de la celda $(0, 0)$ inicial son una combinación de movimientos hacia la derecha (RIGHT) o hacia arriba (UP), que incrementan a i o a j respectivamente.
¿Cuál es el mayor número de pasos que tiene que ejecutar M^* para simular un único paso de M?. De acuerdo a los resultados anteriores, corresponde a un movimiento RIGHT de M, que requiere $i + j + 2$ movimientos a la derecha de M^*.
Como $i + j + 1 \leq T(n) \Rightarrow i + j + 2 \leq T(n) + 1 = T_{max1pasodeM}$, que es una cota para el número máximo de pasos que debe ejecutar M^* para simular un único paso de ejecución de M.

Ejercicio 8 - Tiempo de ejecución de la MT M^*

El peor caso entonces para el número TOTAL de pasos de ejecución de M^* es que todos los movimientos de M sean RIGHT. En dicho caso y como por hipótesis M se ejecuta en $T(n)$ pasos, el total de pasos T_{M^*} de M^* necesarios para simular M resulta ser
$T_{M^*} \leq T_{max1pasodeM} x T(n) = (T(n)+1)T(n) = T(n)^2 + T(n)$
$\Longrightarrow T_{M^*} = \mathcal{O}(T(n)^2)$ ■

Ejercicio 8 - Funcion de mapeo de $(i, j) \longrightarrow k$

Podemos obtener una expresión exacta para la función de mapeo de la celda de M de coordenadas (i, j) en la celda k de M^*.
Para esto observamos que el valor de k se obtiene sumando la cantidad de celdas de la malla en las diagonales a $-45°$ que están por debajo de la celda (i, j), excepto el $(0, 0)$, y agregando las celdas de la diagonal por (i, j) de izquierda a derecha hasta llegar a la celda. Teniendo en cuenta que la diagonal n-ésima tiene n elementos y que la diagonal inmediatamente por debajo de la celda (i, j) tiene $i + j$ elementos, resulta

$$k(i,j) = \sum_{c=2}^{i+j} c + i + 1 = \frac{i^2 + 2ij + j^2 + 3i + j}{2}$$

Aplicación al caso del movimiento RIGHT:
M se mueve desde (i, j) a $(i + 1, j)$.

$$\Longrightarrow k' = k(i+1, j) = \frac{(i+1)^2+2(i+1)j+j^2+3(i+1)+j}{2}$$

$$\Longrightarrow \Delta k = k' - k = i + j + 2$$

La clase de lenguajes P (Polynomial Time)

$$P = \bigcup_{c \geq 1} DTIME(n^c).$$

Es la clase de los algoritmos que pueden resolverse en forma **eficiente** en cuanto a tiempo de ejecución, teniendo una cota superior polinomial de tiempo de ejecucion.

Equivale a decir que un lenguaje $L \in P$ si existe una MT M y dos constantes $c_1 > 0$ y $c_2 \geq 1$ tales que M decide L en un tiempo $T(n) \leq c_1 n^{c_2}$.

Ejercicio 12 - Planteo del problema (1.8 del Arora-Barak)

Se considera la función *LOOKUP* : con entrada $\langle x, i \rangle$ (donde x es un string binario e i es un número natural), *LOOKUP* devuelve el i-ésimo bit de x o 0 si $|x| < i$.

Probar que $LOOKUP \in P$.

Ejercicio 12 - codificación de la entrada $\langle x, i \rangle$ en la MT

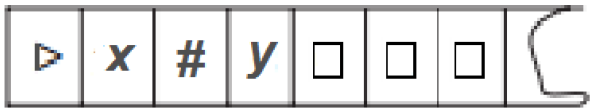
En la MT M se tiene como entrada a la pareja $\langle x, y \rangle$ donde y es la codificación en binario del natural i.
Una forma de poder distinguir en la cinta de entrada entre x e y es usar como separador el simbolo $\#$, de forma que el string de entrada a M sea $x\#y$.
Una manera de codificar esto en bits sin ambiguedades es la siguiente (ver capítulo 0 del Arora-Barak):
$0 \mapsto 00$
$1 \mapsto 11$
$\# \mapsto 01$
El largo de la entrada de la MT vale entonces $n = |x| + |y| + |\#|$.

Ejercicio 12- una posible solución

La idea es diseñar un MT M que busque la posición i-ésima de x recorriendo el string de entrada x, llevando la cuenta del largo del substring de x y comparándolo en cada paso con el valor de i. Si se encuentra la posición i-ésima, copia ese bit de x en la cinta de salida. Si el último bit de x está en una posición menor que i, M escribe 0 en su cinta de salida.

Para implementar esto M emplea 4 cintas:

cinta ① : es la cinta de entrada, de solo lectura



cinta ② : se usa para copiar en ella el valor de y.

cinta ③ : es un contador que se incrementa en cada paso y se compara con el valor de y de la cinta ②.

cinta ④ : cinta de salida donde se escribe el resultado. Para no tener ambiguedades, si x tiene un bit e-ésimo se escribe primero un 1 y luego el valor de dicho bit de x.

Ejercicio 12- descripción de M a nivel de implementación

1. Copiar y en la cinta 2.
2. Mover el cabezal de la cinta 1 a la celda de la derecha. Si el símbolo en dicha celda es #, escribir 0 en la cinta 4 de salida y *PARAR*.
3. Incrementar el contador de la cinta 3.
4. Comparar los valores de las cintas 2 y 3. Si son iguales, escribir 1 en la cinta 4 de salida y a su lado el bit actual de la cinta 1. Si son distintos, volver al paso 2.

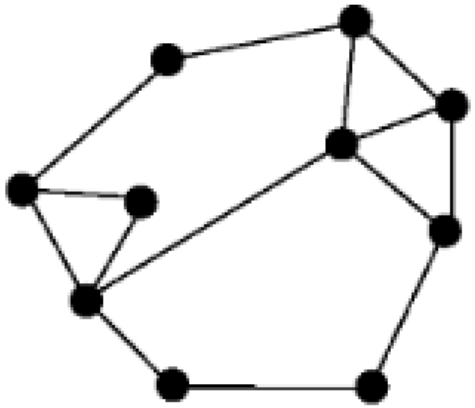
Ejercicio 12-Tiempo de ejecución de la MT M

Los costos en tiempo del algoritmo anterior son los siguientes:
① lleva $\mathcal{O}(|x| + |\#| + |y|) = \mathcal{O}(n)$ pasos.
El bucle ② − ④ se repite en el peor caso $|x|$ veces $(i > |x|)$.
② es $\mathcal{O}(1)$.
③ es $\mathcal{O}(|y|)$.
④ es $\mathcal{O}(|y|)$.
$\Longrightarrow$ M computa $LOOKUP$ en
$\mathcal{O}(|x| * |y|) = \mathcal{O}((|x| + |y|)^2) = \mathcal{O}((|x| + |\#| + |y|)^2) = \mathcal{O}(n^2)$.
$\Longrightarrow$ $LOOKUP \in P$ ■

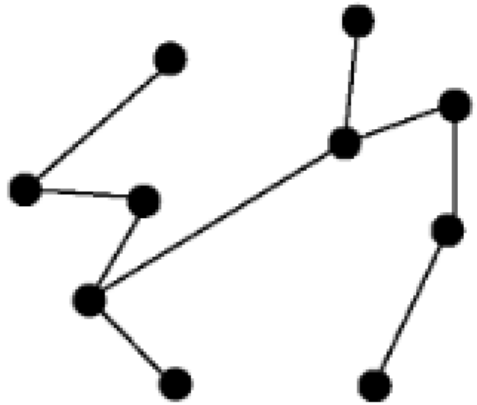
Ejercicio 13 b - Planteo del problema (1.14 del Arora)

Lenguaje/problema de decisión *TRIANGLEFREE* :
El conjunto de todos los grafos que no contienen un *TRIANGULO*.
Se define como triángulo de un grafo G a toda terna de vértices distintos (u, v, w) de G tal que hay arista en G entre todo par de vértices de la terna.

Probar que *TRIANGLEFREE* está en la clase P



No pertenece a TRIANGLEFREE



Pertenece a TRIANGLEFREE

Ejercicio 13 b - una posible solución

Se puede usar la representación del grafo G con la matriz de adyacencia o con listas de adyacencia. Usaremos esta última.
La idea del algoritmo para decidir si $G \in TRIANGLEFREE$ es la siguiente:

1. Para cada nodo u de G, se considera su lista de adyacencia LA_1.
2. Para cada nodo v adyacente a u en G, se considera su lista de adyacencia LA_2.
3. Recorrer la lista LA_2, probando para cada vértice de la misma si también pertenece a LA_1. Si se encuentra un vértice w que verifique ésto, entonces la terna de vértices (u, v, w) es un triángulo de G. El algoritmo para y devuelve FALSE.
4. Si al terminar todas las recorridas no se encuentra ningún triángulo, el algoritmo devuelve TRUE.

Pseudocodigo del algoritmo

```
trianglefree = true
Para u de 1 a n hacer
    LA1=ListaAdyacentes(G,u)
    Mientras NoVacia(LA1) hacer
        v = Primero(LA1)
        LA1 = Resto(LA1)
        LA2=ListaAdyacentes(G,v)
         Mientras NoVacia(LA2) hacer
             w = Primero(LA2)
             LA2 = Resto(LA2)
             Si Pertenece(w,LA1)
                 trianglefree = false
             Fin Si
         Fin Mientras
     Fin Mientras
Fin Para
Devolver trianglefree
```

Medida del tamaño de la entrada en problemas de grafos

Para estimar la complejidad en el tiempo de una MT que ejecuta un algoritmo sobre un grafo G, en un análisis a nivel de pseudocódigo o algoritmo de alto nivel se trabaja usualmente con una medida del tamaño de la entrada dada por $n = N$, número de vértices del grafo. La justificación es la siguiente:

1. Un grafo de N vértices se puede representar en la cinta de entrada de la MT como una matriz de adyacencia o como listas de adyacencia utilizando N^2 bits.
2. Si el algoritmo resulta polinómico en N, es decir corre en $\mathcal{O}(P_1(N))$ en el análisis de alto nivel, también será polinomial para una entrada N^2 en la MT, con un tiempo de ejecución $\mathcal{O}(P_2(N^2))$. Por ejemplo recorrer la lista de adyacencia de un vértice requiere tiempos $\mathcal{O}(1)$para el acceso y $\mathcal{O}(N)$ para el recorrido en el algoritmo de alto nivel, mientras que en la MT se requieren $\mathcal{O}(N^2) = \mathcal{O}(n)$ pasos, es decir es una operación lineal en el tamaño de la entrada en ambos casos.

Complejidad en el tiempo del algoritmo propuesto

Los costos en tiempo del algoritmo anterior son los siguientes:
El bucle ① se repite n veces, una por cada vértice.
El bucle ② interno a ① se ejecuta en el peor caso $n-1$ veces.
El bucle ③ interno a ② requiere en el peor caso $(n-1)^2$ pasos.
$\Longrightarrow$ el algoritmo es $\mathcal{O}(n^4)$.
$\Longrightarrow$ $\mathit{TRIANGLEFREE} \in P$ ■

Ejercicio 7- Planteo del problema (1.5 del Arora)

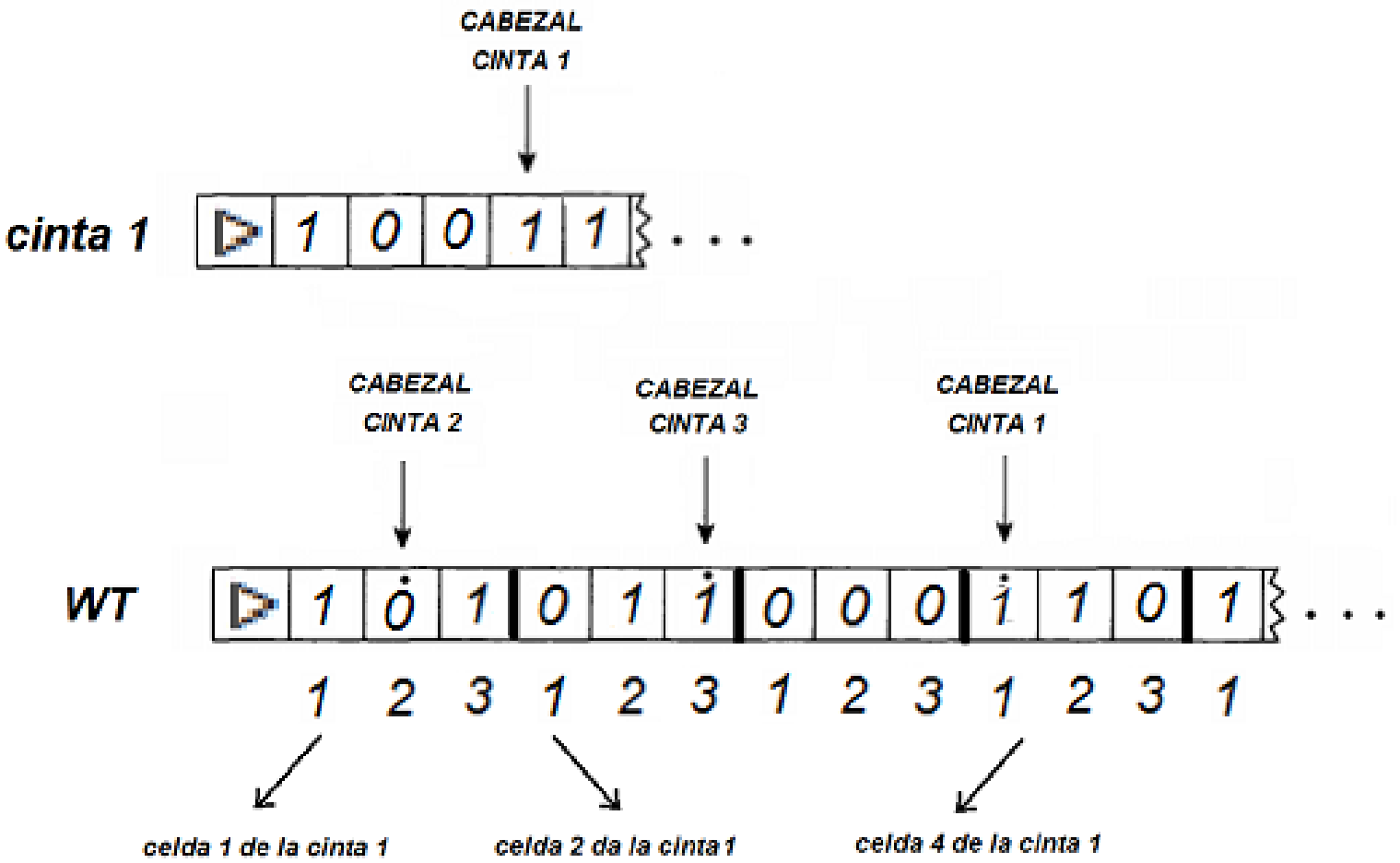
Se define una MT M como *OBLIVIOUS* (denotada como OTM) si los movimientos de sus cabezales no dependen de la entrada sino solo del TAMAÑO de la entrada. Es decir que M es oblivious si para cada entrada $x \in \{0,1\}^*$ y $i \in \mathbb{N}$, la ubicación de cada uno de los cabezales de M en el paso i-ésimo de ejecución con entrada x depende únicamente de $|x|$ y de i.
Mostrar que para toda función time-constructible $T : \mathbb{N} \to \mathbb{N}$, si $L \in DTIME\,(T(n))$ entonces existe una OTM que decide L en tiempo $DTIME\,(T(n)^2)$.

Ejercicio 7 - una posible solución

Sea M la MT de k cintas y M^* la OTM que va a simular el comportamiento de M.
Las características de diseño de M^* son las siguientes:

1. M^* va a emplear 2 cintas: una cinta de entrada y una cinta de trabajo o working tape WT.
2. En la cinta WT de M^* se codifican $TODAS$ las k cintas de trabajo de M, al igual que en la proposición 1.6 de Arora-Barak. En las posiciones $1, k+1, 2k+1...$ se codifica la primera cinta de M, en las posiciones $2, k+2, 2k+2...$ la segunda, etc.
3. El alfabeto de cinta de M^* va a contener todos los símbolos del alfabeto de M, pero además por cada símbolo σ del alfabeto de M en el de M^* se tiene también el símbolo $\dot{\sigma}$. El símbolo con punto se usa para indicar la posición actual del cabezal de cada cinta (en todo momento hay un único caracter con punto por cada cinta de M).

Ejemplo de cinta de trabajo para $k = 3$



Ejercicio 8- Simulación de la MT M con la OTM M^*

Inicialmente, M^* COPIA su cinta de entrada en el WT, en las posiciones $1, k+1, 2k+1, ...n*k+1$ para una entrada de largo n. Para simular cada paso de M, la idea clave es que M^* recorra su WT de PRINCIPIO a FIN y luego vuelva a la posición inicial de la cinta.

- En la pasada hacia la derecha, M^* registra los k simbolos con punto que encuentra asociados a las k cintas de M (simbolos de las celdas apuntadas por los cabezales de M).
- Luego M^* usa la función de transición δ de M para determinar el nuevo estado, los $k-1$ simbolos a escribir en las celdas bajo los cabezales de cada cinta y los movimientos de los k cabezales.
- En la pasada hacia la izquierda se escriben los símbolos y se actualizan las posiciones de los cabezales.

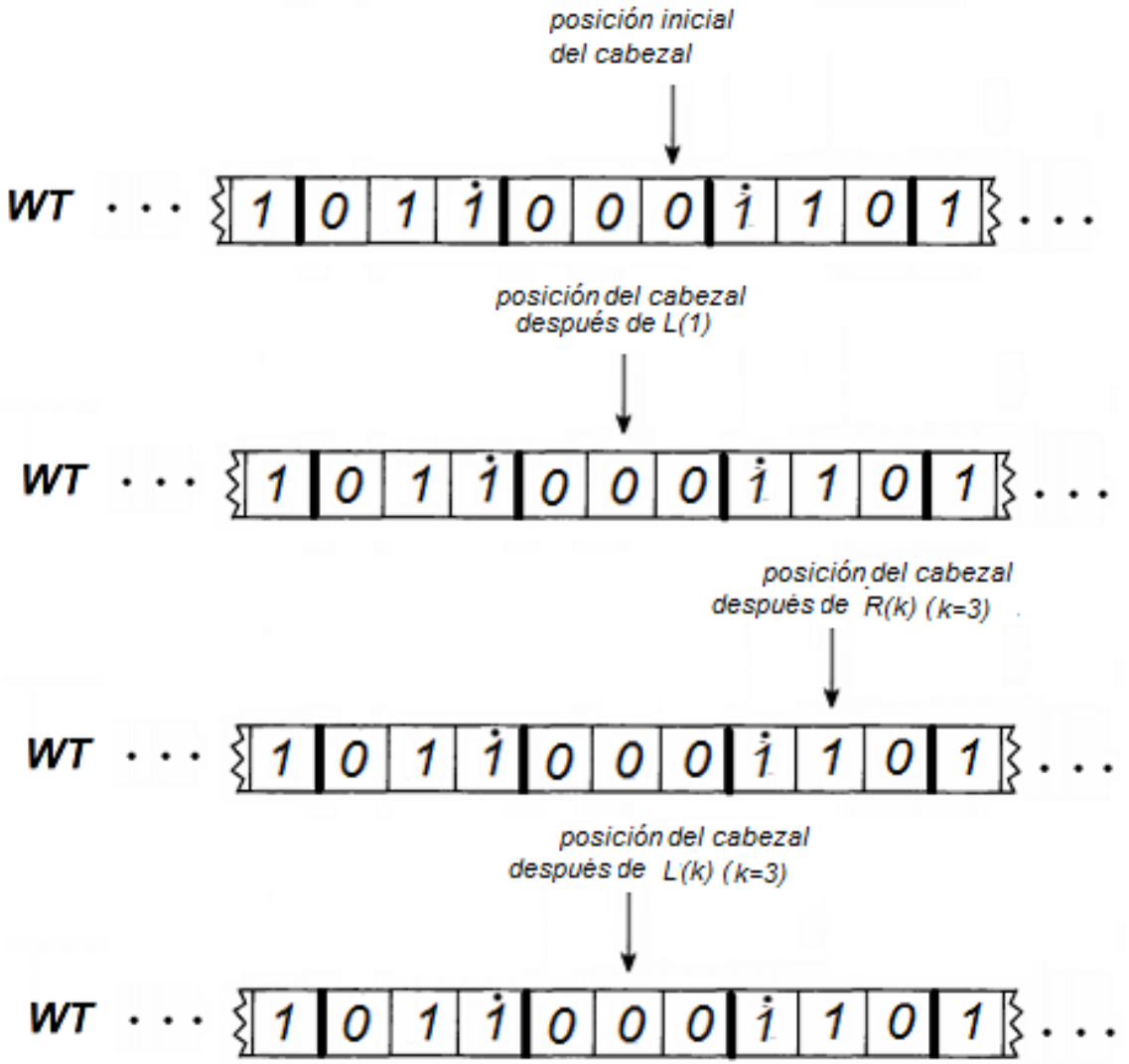
Detalle de la mecánica de actualización de M^*

¿En la pasada hacia la izquierda, cómo hace M^* para escribir los datos y mover los cabezales en forma independiente de la entrada? Cada movimiento en la pasada de derecha a izquierda de M^* se compone de la siguiente secuencia de movimientos:

1. Moverse una celda a la izquierda (denotamos este movimiento como $L(1)$).
2. Moverse k celdas hacia la derecha (denotado como $R(k)$).
3. Moverse k celdas hacia la izquierda (denotado como $L(k)$).

Esta secuencia $L(1)R(k)L(k)L(1)R(k)L(k)...$ permite actualizar las k cintas respetando la definición de OTM, como veremos a continuación.

Secuencia de movimientos $L(1)R(k)L(k)$ en la pasada a la izquierda

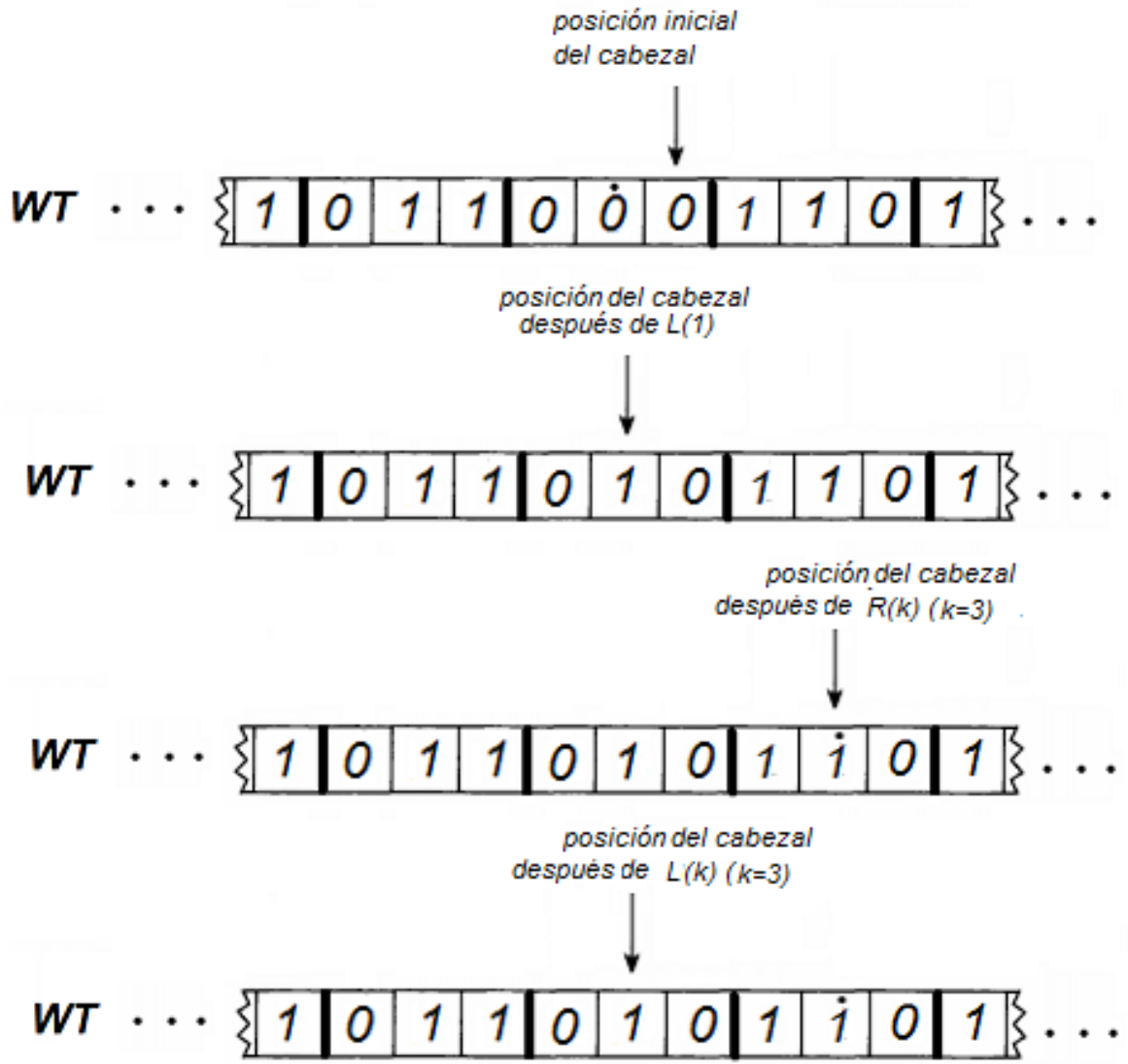


Actualización de una cinta con movimiento R

Supongamos que luego de moverse una celda a la izquierda con $L(1)$, se encuentra un símbolo con punto, correspondiente a la posición del header de una cierta cinta j.
En la pasada hacia la derecha de M^* se determinó que para dicha cinta j corresponde escribir el símbolo σ_1 y mover el cabezal a la derecha (R).
Lo que hace M^* luego de $L(1)$ es lo siguiente:

1. Sustituye el símbolo actual de la celda por el nuevo símbolo calculado σ_1 para la cinta j (sin punto).
2. Hace el movimiento $R(k)$ de k celdas a la derecha
3. Le pone un punto al símbolo encontrado luego de $R(k)$, indicando que esa es la nueva posición del header de la cinta j.
4. Hace el movimiento $L(k)$ de k celdas a la izquierda y guarda información de que la cinta j ya está actualizada (por ejemplo codificando el status de las actualizaciones en el propio estado de la OTM M^*).

Actualización de una cinta con movimiento R

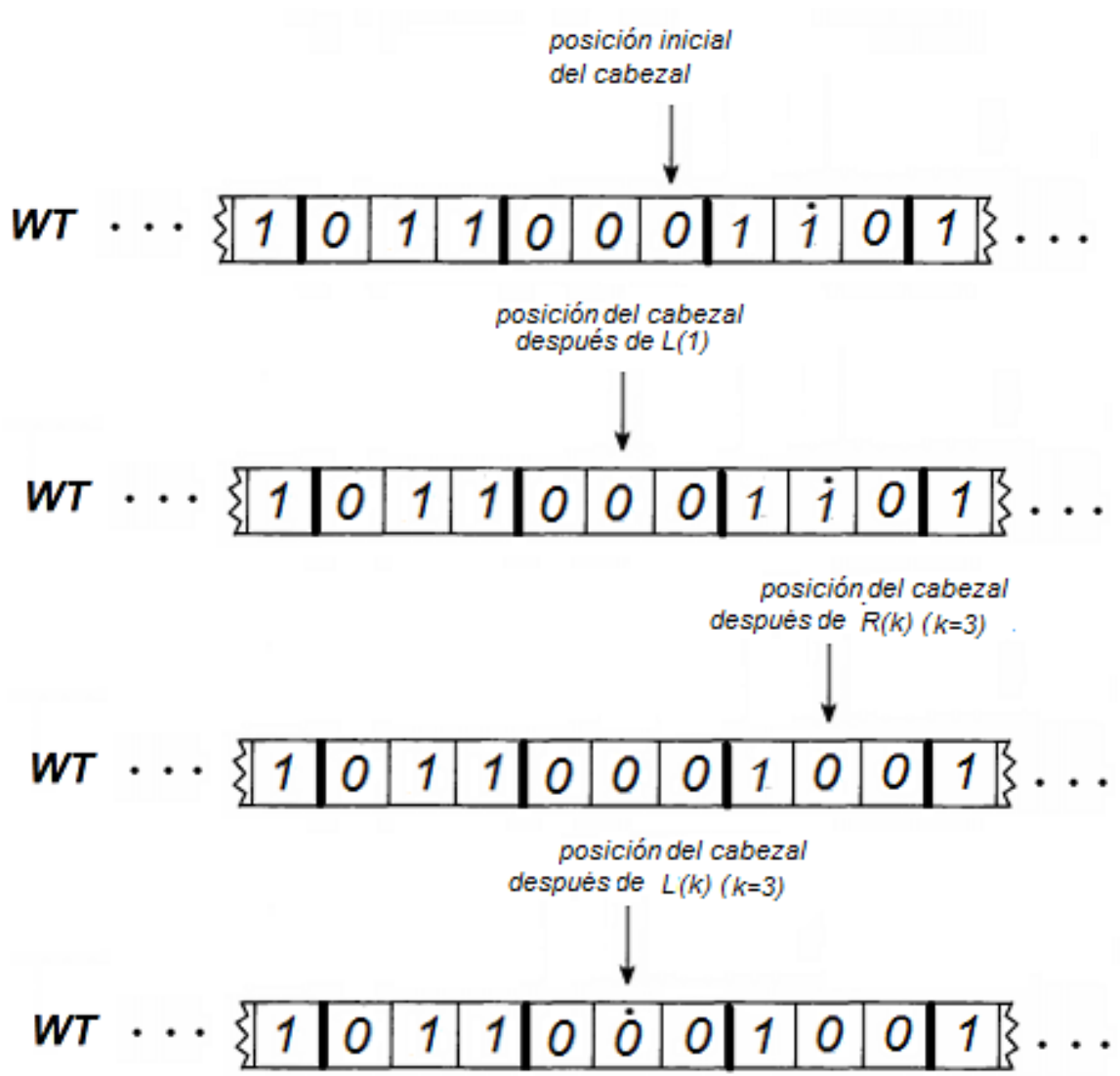


Actualización de una cinta con movimiento L

Supongamos ahora que luego de moverse una celda a la izquierda con $L(1)$ y k celdas a la derecha con $R(k)$ se encuentra un símbolo con punto, correspondiente a la posición del header de una cierta cinta j con status de NO ACTUALIZADA. Eso indica que en esa cinta el cabezal se tiene que mover a la izquierda, y que por eso cuando se pasó anteriormente por dicha celda con punto no se hizo la actualización. En este caso M^* realiza lo siguiente luego de llegar a ella mediante $L(1)R(k)$:

1. Sustituye el símbolo actual de la celda por el nuevo símbolo calculado σ_1 para la cinta j (sin punto).
2. Hace el movimiento $L(k)$ de k celdas a la izquierda
3. Le pone un punto al símbolo encontrado luego de $L(k)$, indicando que esa es la nueva posición del header de la cinta j.
4. Cambia el status dela cinta j a ACTUALIZADA.

Actualización de una cinta con movimiento L



Cantidad de movimientos en la pasada hacia la derecha de M^*

¿Cuantas celdas debe recorrer la OTM M^* en cada pasada hacia la derecha?

La idea es utilizar el hecho de que en cada movimiento de M a lo sumo se visita una celda nueva más de cada cinta.

Lo que hace entonces M^* es al inicio poner un marcador de fin de recorrida en la posición $k+1$. Cada vez que se llega a dicho marcador de fin de recorrida en la pasada hacia la derecha, se mueve dicho marcador k celdas hacia la derecha.

Tiempo de ejecución de la OTM M^*

La observación clave es que, como la MT M por hipótesis decide L en $T(n)$ pasos $\Longrightarrow$ la cantidad de celdas recorridas en cada pasada a la derecha de la OTM M^* es a lo sumo $k * T(n)$, es decir nunca van a haber más de $k * T(n)$ celdas visitadas en el WT de M^*. Por cada paso de ejecución de la MT M, la OTM M^* realiza la siguiente cantidad de pasos:

- En la pasada hacia la derecha, $\mathcal{O}(k * T(n))$ pasos $= \mathcal{O}(T(n))$.
- En la pasada hacia la izquierda, $\mathcal{O}(k * T(n) * 2k)$ pasos $= \mathcal{O}(T(n))$.

$\Longrightarrow$ el tiempo de ejecución de un único paso de M simulado por M^* es $\mathcal{O}(T(n))$. Como M^* debe simular a lo sumo $T(n)$ pasos de ejecución de $M \Longrightarrow$ el tiempo total de ejecución de M^* es $\mathcal{O}(T(n)^2)$.

Fin de ejecución de la OTM M^*

Una observación importante es que la MT M con entrada de tamaño n que se está simulando, puede parar en cualquier momento dependiendo de su entrada.
El tiempo de fin de ejecución de la OTM M^*, por definición de máquina oblivious, no debe depender de la entrada.
Para tener en cuenta esto, inicialmente M^* calcula el valor $T(n)$ en tiempo $T(n)$ (recordar que por hipótesis $T(n)$ es time-constructible).
Luego M^* emplea un contador inicializado en 0, que se incrementa en uno por cada paso simulado de la ejecución de M.
Cuando dicho contador llega al valor $T(n)$, se tiene la seguridad de que la MT M ya paró para cualquier entrada de tamaño n, y por lo tanto en ese momento M^* detiene su ejecución.