Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2213 — Lógica para ciencias de la Computación 2021-1

NOMBRE: Matías Duhalde

# Tarea 3 - Parte A

Sea $M$ una Máquina de Turing que resuelve el problema propuesto en el enunciado.

En primer lugar, $M$ debe verificar que la palabra $w$ que reciba como entrada, tenga el formato correcto, que debe ser de la forma $A\#B\#w_1\#w_2\#...\#w_n$, donde $A$ es el string que representa el posible conjunto oscuro, y $A \in \{0,1\}^n$, $B$ es el string que representa la cantidad de nodos de la forma $1^n$, y los strings $w_i$, con $1 \leq i \leq n$ corresponden al vector de adyacencia del nodo $i$, es decir, representa los nodos que están conectados al nodo $i$ mediante una arista. Estos strings pertenecen a $\{0,1\}^n$.

Este primer paso funcionaría como un parser, de manera análoga a como se comprueba que un string corresponda a la codificación de una Máquina de Turing. Por lo tanto, es posible de computar mediante una MT. En caso de que $w$ no sea válido, $M$ inmediatamente rechaza. Este paso se puede realizar en "una sola pasada". Solamente basta comprobar que todos los substrings (delimitados por el símbolo #) tengan el mismo largo $n$ (igual a la cantidad de nodos), que cumplan el formato correcto, y que haya una cantidad $n+2$ de substrings. Sumado a lo anterior, se debe volver al inicio de $w$ para continuar el algoritmo. De esta manera, la cantidad de **cambios** es constante ($\Theta(1)$).

Posteriormente, se debe comenzar el análisis del conjunto oscuro $S$, codificado en el input $w$ dentro del substring $A = a_1a_2...a_n$. Para cada $a_i$, con $1 \leq i \leq n$, si su valor es 0, entonces el nodo no pertenece al conjunto oscuro $S$ que queremos comprobar. Dado que asumimos que el grafo es no dirigido, no nos interesan sus conexiones, por lo que podemos ignorar todo el substring $w_i$. Para hacer esto, $M$ sustituye todos caracteres del substring $w_i$ a 0, quedando de la forma $0^n$. En el "mejor" caso, ningún carácter en $A$ es 0, o en otras palabras, nuestro conjunto oscuro $S$ es igual a $V$. Aquí, solo habría que hacer dos **cambios** para realizar esta comprobación. Para el peor caso, en el cual todos los caracteres en $A$ son ceros ($S = \emptyset$), habría que hacer $n$ iteraciones, en las cuales se lee la posición del carácter $a_i$, se reemplaza por un carácter auxiliar $x$ para denotar que ya iniciamos la iteración, se va a su substring $w_i$ respectivo, y este se llena de ceros, haciendo dos **cambios** en cada ciclo (ida + vuelta). Además, se realiza un ciclo extra para asegurarse que no hay más ceros en $A$, totalizando una cantidad de **cambios** igual a $2n + 2$, lo que equivale a $\Theta(n)$. *Nota: si bien en el peor caso $M$ puede aceptar directamente debido a que $S$ es vacío, y cumple la condición de conjunto oscuro, decidí hacerlo así para mostrar la cantidad de **cambios**. Además, el input será aceptado de igual manera según lo explicado en el paso siguiente.*

Para el último paso, $A = a_1a_2...a_n$ queda de la forma $\{x,1\}^n$. Para cada carácter de $A$ tal que $a_i = 1$, debemos revisar si alguno de los otros nodos del conjunto oscuro está conectado por una arista a este. Dado que en el paso anterior eliminamos todas las conexiones a los nodos que no son del conjunto oscuro, y sólo permanecen intactas las conexiones de los nodos en $S$, basta con revisar el $i$-ésimo carácter de cada $w_j$, con $1 \leq j \leq n$. Si encontramos un 1, significa que hay una conexión entre un nodo del conjunto oscuro y el nodo que estamos revisando actualmente (que también es del conjunto oscuro), por lo que $M$ rechaza. Si revisamos todos los 1s en $A$, y no nos encontramos con caracteres 1 en los $i$-ésimos caracteres de $w_j$, $M$ acepta. De esta manera, ningún par de nodos $\{s_1, s_2\}$ en $S$ se encuentran conectados, y $S$ efectivamente corresponde a un conjunto oscuro de $G$. Tenemos que en el mejor caso, ningún carácter de $A$ es 1, por lo que se acepta inmediatamente, luego de dos **cambios** que toma en realizar esta verificación. En el peor caso, todos los caracteres $a_i$ de $A$, tienen el valor 1. De esta manera, para cada carácter $a_i$, se reemplaza por un carácter auxiliar $x$ para denotar que ya iniciamos la iteración, y se revisa el $i$-ésimo carácter de los substrings $w_j$. Para el carácter $a_1 = 1$, se revisa el primer carácter de $w_1$, de $w_2$, hasta $w_n$. Para el $a_2 = 1$, se revisa el segundo carácter de $w_1$, $w_2$, hasta $w_n$, y así, hasta el carácter $a_n = 1$, para el cual se revisa el $n$-ésimo (último) carácter de $w_1$, $w_2$, hasta $w_n$. Cada iteración toma dos **cambios** (ida + vuelta), más 1 **cambio**

para darse cuenta que no hay más 1s, por lo que en el peor caso, se producen un total de $2n + 1$ **cambios**, lo cual corresponde a una cantidad lineal de **cambios** $\Theta(n)$.

Podemos comprobar que el lenguaje de $M$ corresponde a aquel pedido en el enunciado. Si $w$ no cumple el formato, $M$ rechaza a $w$ y $w \notin L(M)$. En caso contrario, se aplica el algoritmo descrito anteriormente. Si para algún par de nodos $\{s_i, s_j\} \subseteq S$, con $1 \leq i \leq n$ y $1 \leq j \leq n$ se cumple que existe una arista conectando $s_i$ y $s_j$, entonces, el $j$-ésimo carácter de $w_i$ y el $i$-ésimo carácter de $w_j$ serán 1, lo cual será detectado por la última parte del programa y se rechazará el $w$, y $w \notin L(M)$. Si no se detecta ningún par de este tipo, $M$ acepta a $w$ y $w \in L(M)$. Por lo tanto, $L(M)$ corresponde a las palabras que de la forma $a_1...a_n\#C(G)$, tal que el conjunto representado por los nodos donde $a_i = 1$ corresponde a un conjunto oscuro de $G$. Además, según lo descrito en los párrafos anteriores, $M$ tiene una complejidad de **cambios** lineal acotada por $\Theta(n)$.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2213 — Lógica para ciencias de la Computación 2021-1

NOMBRE: Matías Duhalde

# Tarea 3 - Parte B

Sea $\hat{M}$ una Máquina de Turing no-determinista que resuelve el problema propuesto en el enunciado.

$\hat{M}$ recibe un string de la forma $D\#C(G)$, donde $D$ representa el largo del conjunto oscuro que queremos encontrar en $G$, y es de la forma $1^k$, con $0 \leq k \leq n$, y $n$ la cantidad de nodos del grafo $G$. En primer lugar, $\hat{M}$ comprueba que el string $D$ tenga el formato correcto, lo cual se puede realizar comparándolo con el primer substring de $C(G)$, que es de la forma $1^n$ y representa la cantidad de nodos del grafo. La comparación puede realizarse bit por bit, lo que tomaría en el peor caso (cuando $k = n$) una cantidad de cambios lineal en base a $n$, acotada por $\Theta(n)$. En caso de no cumplir con el formato, se rechaza $w$.

El segundo paso, es añadir un string $A$ de la forma $0^n$ entre $D$ y $C(G)$, delimitado por #s, quedando la entrada $w$ de la forma $D\#A\#C(G)$. Para hacer esto, basta con extender el string $D$ hacia la izquierda con ceros hasta llegar a un largo $n$ (hacer *padding*), colocar un #, y posteriormente colocar 1s a la izquierda del # hasta generar un string de largo $k$, que correspondería al nuevo substring $D$. Finalmente, se deben sustituir todos los caracteres del string $A$ a ceros, llegando hasta el carácter # a la derecha de $A$. Hacer el *padding* usaría una cantidad $2n$ de cambios, alternando bit a bit entre $D$ y el primer substring de $C(G)$. Luego, generar el string $1^k$ a la izquierda tomaría $2k$ cambios. En el peor caso, $k = n$, por lo que estaría acotado por $2n$. Finalmente, reemplazar los 1s del string $A$ por ceros tomaría un solo cambio. De esta manera, en este paso los cambios están acotados por $\Theta(n)$.

El tercer paso, consiste en configurar el string $A$ para representar un posible conjunto oscuro. Sea $q_A$ el estado actual de $\hat{M}$ y $\delta$ su función de transición, entonces $\delta(q_A, 0) = \{(q_A, 0, \leftarrow), (q_A, 1, \leftarrow)\}$. Al llegar al carácter # a la izquierda de $A$, $\hat{M}$ cambia de estado. Aquí comienza el no-determinismo de $\hat{M}$. Se generan todos los strings posibles $A$, que representan a todos los subconjuntos de $V$, que posteriormente serán evaluados para ver si corresponden a un conjunto oscuro. Un ejemplo de los conjuntos generados para $n = 3$, y los pasos que se siguen para cada resultado, se puede ver en la figura 1. Este paso no requiere de ningún cambio ($\Theta(1)$).

Una vez generado el string $A$, se debe verificar que este cumpla la condición de que el tamaño del subconjunto que representa sea $k$, o en otras palabras, que $A$ tenga una cantidad $k$ de caracteres 1. Para

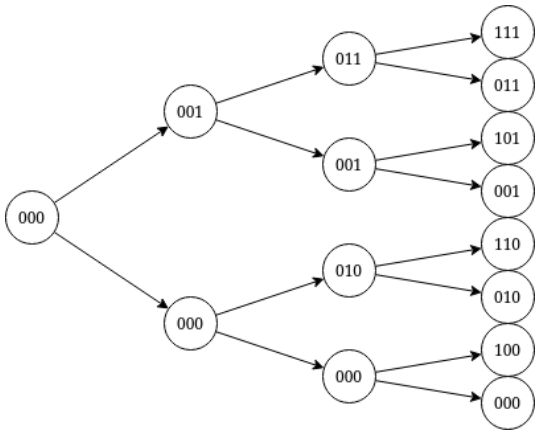


Figure 1: Ejemplo de strings $A$ generados para $n = 3$

esto, se compara $D$ con $A$ uno a uno, tal que haya misma cantidad de caracteres 1 en ambos substrings. Con $D = d_1 d_2 ... d_k = 1^k$ y $A = a_1 a_2 ... a_n$, se aplicaría la siguiente heurística. Se borra $d_1$ (se substituye por $\epsilon$), y se busca el primer $a_i = 1$ en $A$, con $1 \leq i \leq n$. Si se encuentra, se reemplaza temporalmente por $x$, y se vuelve al comienzo de $D$ (es decir $d_2$), y se repite el procedimiento. Si en algún momento no se encuentra ningún carácter en $A$ tal que $a_i = 1$, significa que la cantidad de nodos en $A$ es menor que $k$, por lo que termina la ejecución de esta rama. Si se eliminan todos los caracteres de $D$, se elimina el carácter $\#$, y se revisa si queda algún $a_i$ en $A$ tal que $a_i = 1$. Si se encuentra este caso, también se termina la ejecución, ya que la cantidad de nodos en $A$ es mayor a $k$. Si esto no ocurre, se vuelve al inicio de $w$, y en el camino se reemplazan los caracteres auxiliares $x$ por 1. En el peor caso, el largo $k$ de $D$ es $n$, por lo que se deben realizar $2n$ cambios. Entonces, la cantidad de cambios en este paso está acotada por $\Theta(n)$.

Al llegar a este punto, sólo bastaría en realizar la comprobación si $A$ representa a un conjunto oscuro de $G$. Para hacer esto, sólo basta verificar si el string $w$ resultante pertenece al lenguaje de la Máquina de Turing $M$ definida en el ítem anterior, por lo que podemos ocuparla como subrutina. Además, sabemos que la cantidad de cambios de esta máquina está acotada por $\Theta(n)$. Sumando lo descrito en los párrafos anteriores, podemos concluir que la cantidad de cambios de una ejecución de $\hat{M}$ también está acotada por $\Theta(n)$.

El lenguaje de $\hat{M}$ corresponde a aquel pedido en el enunciado. Si $w$ no cumple el formato, se rechaza y $w \notin L(\hat{M})$. En caso contrario, de manera no-determinística se generan todos los subconjuntos posibles de $V$. $\hat{M}$ se detiene para aquellas máquinas donde el largo del subconjunto es distinto a $k$. Para los casos donde el largo del subconjunto es $k$, se pasa $w$ modificada a $M$. Si $M$ acepta al input, entonces $\hat{M}$ también acepta, y $w \in L(\hat{M})$. Si por otro lado en ninguna ejecución de $\hat{M}$, $M$ acepta, entonces $w \notin L(\hat{M})$. Por lo tanto, el lenguaje $L(\hat{M})$ de $\hat{M}$ corresponde a las palabras $w$ de la forma $1^k \# C(G)$ tal que $C(G)$ es una codificación de un grafo, y existe un conjunto oscuro de tamaño $k$ en $G$.