Tarea 3

Profesor: Pablo Barceló - **Auxiliares:** Bernardo Subercaseaux, Javier Oliva
Ayudantes: Joaquin Cruz, Heinich Porro, Lucas Torrealba, Florencia Yañez
Alumno: Sebastián Sepúlveda A.

Soluciones

P1.1 Demuestre: Si $G = (V, E)$ es un grafo dirigido y fuertemente conexo tal que $\forall v \in V$ se cumple que $in(v) = out(v)$, entonces G tiene un circuito euleriano.

Para comenzar, notemos que como G es un grafo dirigido y conexo, y además se cumple que para todo $v \in V$ la cantidad de arcos incidentes a v, es decir los arcos que "entran y salen", es la misma:

$$in(v) = out(v)$$
$$|\{(v, u) \in E : u \in V\}| = |\{(u, v) \in E : u \in V\}| = s$$

Dado a que estamos trabajando con grafos dirigidos, ocuparemos la definición de $grado(v) = grado^+(v) + grado^-(v)$ donde $grado^+(v)$ el número de arcos en el conjunto $in(v)$ y $grado^-(v)$ el número de arcos en el conjunto $out(v)$. Por lo dicho anteriormente $grado(v) = 2s$. Lo que queremos demostrar es que existe ciclo dirigido en G tal que se pasa por cada arco solo 1 vez.
Primero asumimos que $|V| \geq 2$, con esto el $grado(v) \geq 2k$ con $k \in \mathcal{N}$. Dado un vértice $v_0 \in V$ comenzaremos a construir el circuito euleriano. Lo primero que hacemos es construir un camino simple arbitrario:

$$v_0 e_1 v_1 \dots v_{k-1} e_k v_k$$

Como G es finito y fuertemente conexo la frecuencia será finita y existe. Además, va a tener un limite donde v_k sea el máximo largo que tendrá el camino. Por tanto, podemos señalar que este camino máximo será cuando:

$$v_0 = v_k$$

En efecto, por contradicción, asumamos que $v_0 \neq v_k$. Esta conexión aporta en valor impar al grado de v_k. Luego, como v_k tiene grado par, dado a lo dicho en el comienzo, entonces existe un camino desde v_k por el cual podemos extender el camino desde v_0. Esto es una contradicción, pues ya habiamos señalado que v_k era máximo. Finalmente, tenemos que existe un camino euleriano en G. Lo que no podemos señalar es que este camino sea único, pues el *grado* de cada vértice no necesariamente es el mismo, y pueden existir más de 1 sólo camino euleriano.

P1.2 Dado k, modele el problema utilizando grafos. Hint: Utilice $V = \Sigma^k$ como conjunto de vértices.

Primero definimos el grafo que ocuparemos como $G_k(V, E)$, con $|V| = n$ y $|E| = m$, y el subindice k como la palabra $k-$completa que queremos formar. Luego definimos nuestros casos bases. Para $k = 0$ definimos nuestro grafo como vacio. Mientras que para $k = 1$ sabemos que debemos generar vértices que serán los distintos valores que pertenecen a Σ^k, es decir $\{1, 0\}$.

En general, cada arco e_i que una a los vértices del grafo será una concatenación entre los vértices v_{i-1} y v_i, y tendrá un peso w asociado, que significará el largo de la palabra que se forma de la unión de ambos vértices.

Como varias palabras no son palindromos, debemos considerar un grafo G_k dirigido en ambos sentidos, pues nos interesa el orden que tendrá la unión de las palabras y queremos encontrar todas las palabras posibles (**Ver Figura 1**). Sin embargo, como queremos encontrar la palabra con menor largo, debemos considerar aquellos caminos del grafo dirigido que pasen por todos los arcos una vez, y que sean de menor peso **Ver Figura 2** .

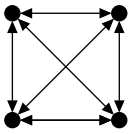

Figura 1: Grafo dirigido general

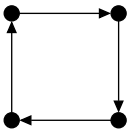

Figura 2: Circuito euleriano conveniente, sin pesos

Entonces, volviendo al caso $k = 1$, primero formamos nuestro grafo G_1 que será dirigido en ambos sentidos, conexo y con pesos, ver imagen de la izquierda en **Figura 3** :

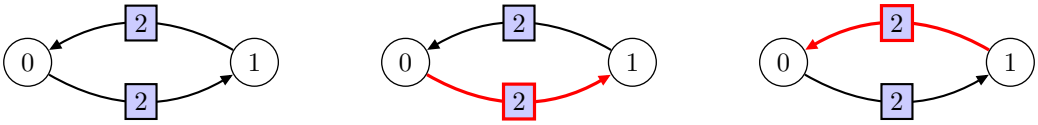

Figura 3: De izquierda a derecha: grafo conexo, dirigido que se debe crear; camino que genera la palabra 10, camino que genera la palabra 01

Luego vemos los caminos que hay para formar una palabra. Es importante notar que debemos detenernos cuando volvemos a un vértice ya recorrido, pues en ese caso estamos considerando la palabra que representa el vértice 2 veces. En la **Figura 3**, la imagen del centro y de la derecha, el arco pintado en rojo no se considera, pues volvemos al vértice inicial.

Para $k \geq 1$ vamos a tener una cantidad de 2^k vértices siguiendo el procedimiento anterior. Dado a que en un inicio queremos ver todas las palabras posibles que podemos formar con el conjunto que nos entrega Σ^k entonces G será 2^k-completo, es decir, que cada arco tendrá $|in(v)| = |out(v)|$, pues G es dirigido en ambos sentidos. Por esto y la parte anterior, podemos concluir que G tiene circuito euriliano. Es decir, que cualquier grafo que creemos vamos a poder construir una palabra que considere todos los elementos de Σ^k.

Ahora debemos resolver que sea del menor largo posible. Para ello ocupamos Dijkstra para encontrar el camino más corto que pase por todos los arcos antes de formar el circuito euleriano. Una vez encontrado el camino que pase por todos los vértices y con peso minimo, esta información lo guardamos y repetimos el procedimiento para un camino

distinto, si es que lo hay, comenzando desde el punto inicial.

Consideremos el ejemplo para $k = 2$ (**ver Figura 4**). Podemos notar que en un sólo circuito euleriano podemos encontrar, en particular 4 palabras distintas con mismo largo minimo.

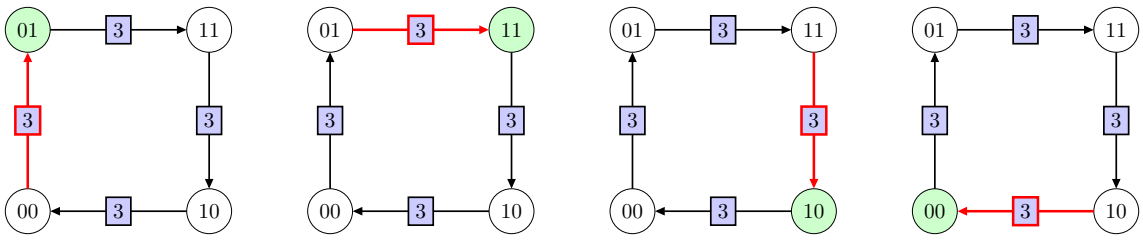

Figura 4: Circuito euleriano para grafo generado por elementos de Σ^2

Destacamos también que es importante, para encontrar este camino con peso minimo, que los vértices que son vecinos el uno del otro tengan un patrón de similitud entre ellos, es decir que entre los vértices que se unen, supongamos v_1 y v_2, la subpalabra final de largo $k-1$ de v_1 sea igual a la subpalabra inicial de v_2 de largo $k-1$ (el caso análogo también es cierto). Por ejemplo, para $k = 3$ la palabra 001 tiene subpalabra de largo 2 a 00 y 01, por lo que todas las palabras que terminen con 00 o comiencen con 01 serán vecinos de 001.
Por último, nuestro grafo no va a tener loops, pues no queremos considerar en nuestro resultado final un elemento de Σ^k que se repita 2 veces en nuestra palabra $k-$completa.

P1.3 Demuestre que existe una palabra $k-$completa cuyo largo es menor a $2^{k+1} + k - 1$.

En efecto, como queremos encontrar la palabra más corta posible que sea $k-$completa, el circuito euleriano con menor peso debe ser nuestro candidato. Notemos que las palabras $k-$completa tienen un largo máximo de $k2^k$ en el caso que no simplifiquemos las subpalabras que estén dentro de la palabra completa. Esta cota superior nos confirma que debe existir un valor menor que ese.

Consideremos el ejemplo de la Figura 4, donde tenemos un circuito euleriano y veamos el procedimiento de unir las subpalabras. Primero tenemos todos los nodos de largo 2 que tienen que ser unidos con un nodo vecino del menor peso. Sea el nodo que estamos analizando v_1 y el nodo vecino candidato a unión v_2. El nodo vecino v_2 va a ser una palabra que tenga una subpalabra de largo 2-1 al final o al inicio igual a la subpalabra de largo 2-1 inicial o final de v_1. Luego, repetimos el procedimiento recursivamente para cada nodo, y después para cada unión de cada nodo con su vecino, hasta que generemos 1 sola palabra. El procedimiento se explica gráficamente en la siguiente imagen:

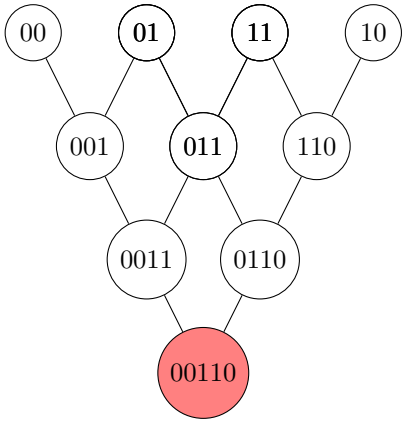

Figura 5: Generación de palabra

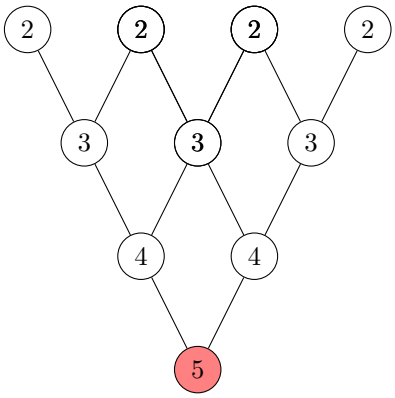

Figura 6: Largo de palabra

El procedimiento de unión en general será juntar vértices v_i y v_{i+1} vecinos que tengan un arco con peso minimo, lo que se traduce a que la subpalabra final de largo $k-1$ de v_i sea igual a la subpalabra inicial de v_{i+1} de largo $k-1$ o que la subpalabra inicial de largo $k-1$ de v_i sea igual a la subpalabra final de v_{i+1} de largo $k-1$

En general, como tendremos grafos con $|V| = 2^k$ donde cada valor de cada vértice tendrá largo k, entonces en cada circuito euleriano de peso minimo tendremos el siguiente resultado:

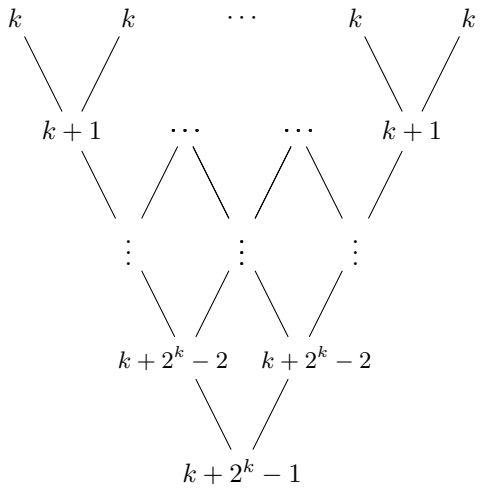

Figura 7: Caso general de largo de palabra

Donde, como mencionamos antes, k es el largo de cada elemento de Σ^k y $2^k - 1$ es la cantidad de arcos que hay desde las hojas hasta la raiz. Por tanto existe una palabra que será $k-$completa de largo $2^k - 1 + k$ menor que $2^{k+1} - 1 + k$.

P1.4 Demuestre que si S es una palabra $k-$completa tal que no existen palabras $k-$completas más cortas entonces su largo es $2^k + k - 1$.

En la pregunta anterior vimos que existía una palabra $k-$completa de largo $2^k + k - 1$, ahora queremos demostrar que es mínima, es decir, que no hay otra palabra de largo menor a esa. Haremos la demostración por contradicción.

Suponemos que hay una palabra de largo menor a 2^k+k-1 por ejemplo 2^k+k-2. Veamos que esto no puede suceder.

Si volvemos al análisis de la parte 2 y 3, vemos que la palabra de largo $2^k + k - 2$ es una palabra generada por la union de $2^k - 1$ nodos de largo k, cuyo camino al nodo raíz es de $2^k - 2$. Por ende, esta palabra puede ser $(k - 1)-$completa, pero no así ser $k-$completa, pues no tendría una subpalabra igual al nodo restante. Esto se explica graficamente en la siguiente figura:

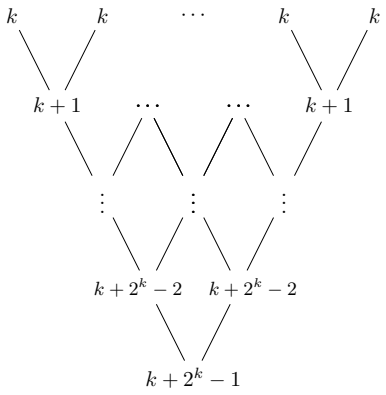

Figura 8: Largo palabra $k-$completa minima

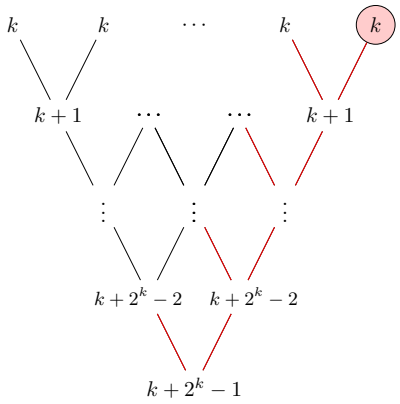

Figura 9: Extracción de un nodo de la Figura 8

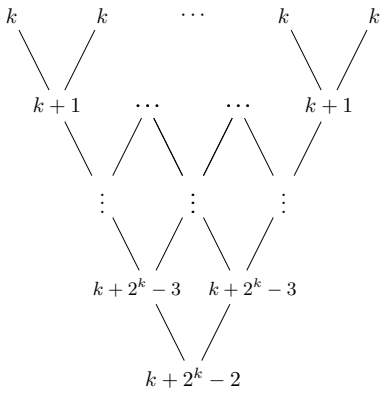

Figura 10: Largo palabra resultante con extracción de nodo

P1.5 Entregue un archivo `MiStringBinario.py,java,cpp` que reciba por entrada estándar un natural k e imprima una palabra $k-$completa de largo mínimo. *[Adjunto al documento]*

P2.1 Entregue una forma cualquiera de hacer llegar a todos los magos al cuarto del señor tenebroso.

Primero, para simplificar términos definimos a cada personaje como elementos v_i, con $i \in [1, \ldots, n]$, con $n = 8$ en este caso particular, y el tiempo que tarda cada uno como t_i. Además, los tiempos los ordenamos en la siguiente tabla:

Tabla 1: Tiempos de cada personaje en el ejemplo del problema

	Hermione	*Harry*	*Ron*	*George*	*Luna*	*Neville*	*Ginny*	*Fred*
	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
T (min)	30	40	40	40	50	50	75	120

Para que todos lleguen al cuarto, los hechiceros deben salir de grupos de a 2, y debe volver 1, para que otro par de hechiceros ocupe la capa. Independiente del tiempo que tarden.

Veamos un ejemplo cualquiera para los 8 magos del problema $(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)$. La secuencia para que lleguen al destino sería:

Tabla 2: Secuencia para que todos los magos lleguen al destino

Secuencia	Magos en el punto inicial	Magos en el punto final
$v_1 v_2 \rightarrow v_1 v_2$	$v_3, v_4, v_5, v_6, v_7, v_8$	v_1, v_2
$v_1 \leftarrow v_1$	$v_1, v_3, v_4, v_5, v_6, v_7, v_8$	v_2
$v_1 v_3 \rightarrow v_1 v_3$	v_4, v_5, v_6, v_7, v_8	v_1, v_2, v_3
$v_3 \leftarrow v_3$	$v_3, v_4, v_5, v_6, v_7, v_8$	v_1, v_2
$v_3 v_4 \rightarrow v_3 v_4$	v_5, v_6, v_7, v_8	v_1, v_2, v_3, v_4
$v_4 \leftarrow v_4$	v_4, v_5, v_6, v_7, v_8	v_1, v_2, v_3
$v_5 v_6 \rightarrow v_5 v_6$	v_4, v_7, v_8	v_1, v_2, v_3, v_5, v_6
$v_5 \leftarrow v_5$	v_4, v_5, v_7, v_8	v_1, v_2, v_3, v_6
$v_4 v_5 \rightarrow v_4 v_5$	v_7, v_8	$v_1, v_2, v_3, v_4, v_5, v_6$
$v_5 \leftarrow v_5$	v_5, v_7, v_8	v_1, v_2, v_3, v_4, v_6
$v_7 v_8 \rightarrow v_7 v_8$	v_5	$v_1, v_2, v_3, v_4, v_6, v_7, v_8$
$v_7 \leftarrow v_7$	v_5, v_7	$v_1, v_2, v_3, v_4, v_6, v_8$
$v_5 v_7 \rightarrow v_5 v_7$		$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$

P2.2 Modele el problema de hacerlo en el menor tiempo posible como uno de distancia mínima en un grafo. Especifique el grafo usado.

Para poder lograr la menor cantidad del tiempo en el traslado de los magos primero hay que trasladar a todos los magos al cuarto, esto es lo que se logró en la parte anterior. Los traslados de a pares son $n-1$ en total. El regreso de cada mago son 1 menos a los viajes de a pares, es decir $n-2$, pues el último viaje de los n magos siempre es par. Esto nos entrega el número total de tiempos que es $2n-3$. Esto es importante para no considerar en la construcción del grafo aquellos caminos que superen esta cota, pues son viajes que hace 1 solo mago con la capa en ida y vuelta, lo cual no tiene sentido respecto a lo que queremos resolver en el problema. Además, de esta manera logramos movimientos recursivos, es decir, cada vez que realizamos un movimiento, estamos obligados a seguir otro movimiento ya conocido.

Luego, como los magos salen de a pares y vuelve 1 siempre, el grafo que se va a construir debe considerar dos estados importantes, estar en un punto inicial A, o estar en un punto final B. Al comienzo del problema todos los magos van a estar en la posición A, luego se van a tomar pares de elementos de este conjunto para trasladarlos al punto B y así recursivamente. En general como el procedimiento es recursivo, dado a lo que se explicó en el párrafo anterior, si B tiene elementos un mago de los que están en B debe ser trasladado al punto A, si y solo sí es que A tiene elementos, para que luego dos elementos de A sean trasladados a B. Esto implica que los estados o nodos de nuestro grafo sería estar en A o B y los arcos que unen a estos estados serian el/los magos que se van a cambiar de estado y el tiempo que cuesta realizar tal cambio.

Dado esto, el grafo que va a modelar el problema será un grafo dirigido, pues el proceso debe entregar un resultado final; conexo, pues un estado implica otro estado dado un movimiento, y la cantidad de conexión entre los nodos es finita.

Para simplificar la cantidad de nodos y para una mejor representación del grafo se van a considerar 3 elementos $\{v_1, v_2, v_3\}$ inicialmente en el punto A, que se denotará a_1, a_2, a_3. En caso de que a_1 y a_2 cambien de estado, este será a b_1 y b_2 respectivamente, donde el arco que conecta a este cambio será $T_1, 2$ que denota el máximo entre los nodos que cambian de estado, en este caso 1 y 2. El grafo para este caso se ve en la siguiente imagen:

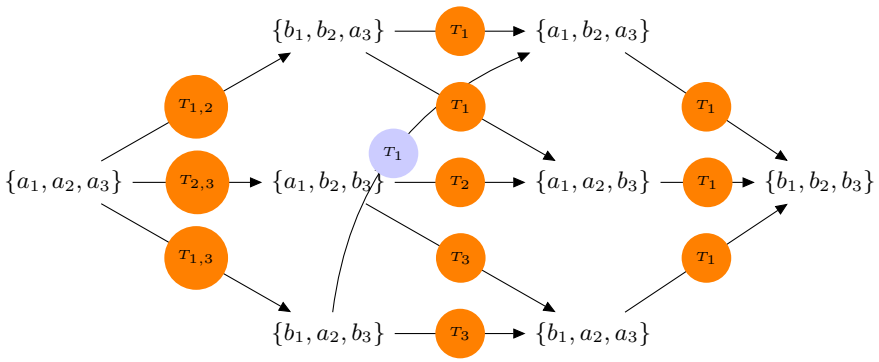

Figura 11: Grafo en caso de tener 3 elementos

Dado a que hay dos estados que son estar en A o en B los elementos de A pueden ser booleanos (`True`, `False`) o $\{1, 0\}$, los elementos de B serian los que no se escoje para A.

P2.3 ¿Cuál es el menor tiempo posible con los tiempos entregados? Describa una secuencia de traslados que permita obtener ese tiempo total.

La secuencia de traslados sería la siguiente:

Tabla 3: Traslados minimos que se deben realizar al grafo

Secuencia	Magos en el punto inicial (A)	Magos en el punto final (B)
$v_1v_2 \rightarrow v_1v_2$	$v_3, v_4, v_5, v_6, v_7, v_8$	v_1, v_2
$v_1 \leftarrow v_1$	$v_1, v_3, v_4, v_5, v_6, v_7, v_8$	v_2
$v_7v_8 \rightarrow v_7v_8$	v_1, v_3, v_4, v_5, v_6	v_2, v_7, v_8
$v_2 \leftarrow v_2$	$v_1, v_2, v_3, v_4, v_5, v_6$	v_7, v_8
$v_1v_2 \rightarrow v_1v_2$	v_3, v_4, v_5, v_6	v_1, v_2, v_7, v_8
$v_1 \leftarrow v_1$	v_1, v_3, v_4, v_5, v_6	v_2, v_7, v_8
$v_5v_6 \rightarrow v_5v_6$	v_1, v_3, v_4	v_2, v_5, v_6, v_7, v_8
$v_2 \leftarrow v_2$	v_1, v_2, v_3, v_4	v_5, v_6, v_7, v_8
$v_1v_2 \rightarrow v_1v_2$	v_3, v_4	$v_1, v_2, v_5, v_6, v_7, v_8$
$v_1 \leftarrow v_1$	v_1, v_3, v_4	v_2, v_5, v_6, v_7, v_8
$v_1v_3 \rightarrow v_1v_3$	v_4	$v_1, v_2, v_3, v_5, v_6, v_7, v_8$
$v_1 \leftarrow v_1$	v_1, v_4	$v_2, v_3, v_5, v_6, v_7, v_8$
$v_1v_4 \rightarrow v_1v_4$		$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$

En general el método para encontrar el menor tiempo para trasladar a los elementos del punto inicial al final es ordenar los tiempos de menor a mayor. Luego se toman los dos menores del conjunto A y se trasladan a B volviendo al menor elemento, ahora en B hacia A. Llegado este punto los elementos de B son impares, mientras que en el paso anterior los elementos de B eran par, por ende se sigue una recursión donde el invariante es la cantidad de elementos de A, y las condiciones para escoger los elementos que se trasladan de A hacia B, y viceversa, se escoge según la paridad en la cantidad de elementos de B y los elementos que van quedando en A. Con esto construimos el algoritmo que permite calcular el menor tiempo.

P2.4 Entregue un archivo `hogwarts.{py,java,cpp}` que al ejecutarlo reciba por entrada estándar 1 entero n la cantidad de magos disponibles (Todos deben llegar al cuarto del señor tenebroso) seguido de n enteros t_i que indica el tiempo que toma el mago número i y entregue en la salida el tiempo mínimo que tomaría a esos magos llegar al cuarto del señor tenebroso.

Adjunto al documento