Tarea 3

Profesor: Alejandro Hevia
Auxiliares: Ivanna Bachmann, Rodrigo Fuentes, Vicente Rojas
Alumno: Sebastián Sepúlveda A.

Soluciones

Problema 1.1 Los strings sobre el alfabeto $\{0, 1, 2\}$ que representan, en base 3, a un número n, tal que n no es divisible por 9

R:

$$[0^+(1|2)] \mid \{0^*[(1|2)(0|1|2)]\}$$

Problema 1.2 Los strings sobre $\{a, b\}$ que contengan ab como substring, pero que no contengan ba.

R:

$$(a^+b^+)^*$$

Problema 1.3 Los strings sobre $\{a, b, c\}$ que contengan abc como substring, pero que no empiecen y terminen con el mismo símbolo (por ejemplo, $abcb$ está en el lenguaje, pero $babcb$ y $abca$ no)

R:

$$b^*(a^+bc^+)^*|(a^+bc^+)^*b^*$$

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.
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.

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.

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:

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:
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

Tabla 4: Add caption

Linea de productos	**Fecha primera operación comercial**	**Planta representativa/características**
BWR/1	1960	Dresden 1 BWR de tamaño comercial inicial
BWR/2	1969	Oyster Creek Plantas compradas únicamente en economía Planta de ciclo directo grande.
BWR/3	1971	Dresden 2 Primera aplicación de bomba Jet Sistema de refrigeración de emergencia mejorado (ECCS)
BWR/4	1972	Vermont Yankee Mayor densidad de potencia (20 %)
BWR/5	1977	Tokai 2 ECCS mejorado Control de flujo de válvula
BWR/6	1978	Clinton (1987) Sala de control compacta Sistema de protección de reactor de estado sólido.
ABWR	1996	Kashiwasaki-Kariwa 6 Reactor interno de bombas. Accionamiento de barras de control de movimiento fino

Tabla 5: Add caption

Linea de productos	Fecha primera operación comercial	Planta representativa/características
BWR/1	1960	Dresden 1
BWR/2	1961	Dresden 1 BWR de tamaño comercial inicial
BWR/3	1962	Oyster Creek Plantas compradas únicamente en economía Planta de ciclo directo grande.
BWR/4	1963	Dresden 1 BWR de tamaño comercial inicial
BWR/5	1964	Dresden 1 BWR de tamaño comercial inicial
BWR/6	1965	Dresden 1 BWR de tamaño comercial inicial
BWR/7	1966	Dresden 1 BWR de tamaño comercial inicial