Examen de Programación 3
12 de julio de 2019

En recuadros con este formato aparecerán aclaraciones que cumplen una función explicativa pero que no eran requeridos como parte de la solución.

Ejercicio 1 (30 puntos)

En este ejercicio aplicamos uno de los argumentos centrales que utilizamos en el curso para el análisis del algoritmo de Dijkstra.

Sea $G = (V, E)$ un grafo dirigido en el que cada arista $e \in E$ tiene asociada una *longitud* positiva ℓ_e. La *longitud de un camino* P en G es la suma de las longitudes de las aristas que lo componen.

Sea $s \in V$ un vértice arbitrario. Para cada vértice $w \in V$ tal que existe un camino de s a w en G definimos la *distancia* de s a w, denotada $d(w)$, como la longitud mínima entre los caminos de s a w.

Consideramos un conjunto $S \subset V$ que contiene a s y definimos E_S como el conjunto de aristas (x, y) que parten de un nodo x de S y terminan en un nodo y de $V \setminus S$. Supongamos que para cada vértice $u \in S$ se conoce un camino P_u de s a u en G de longitud $d(u)$.

Sea $(u, v) \in E_S$ tal que

$$d(u) + \ell_{(u,v)} = \min_{(x,y)\in E_S} \{d(x) + \ell_{(x,y)}\}. \tag{1}$$

(a) Demuestre que $d(v) = d(u) + \ell_{(u,v)}$, mostrando que el camino P de s a v que se obtiene concatenando v a P_u es un camino de longitud mínima de s a v.

(b) Suponga ahora que G puede tener aristas con longitud negativa. ¿Sigue siendo válida la demostración de la parte (a)? Justifique.

Notación: Dado un camino simple $P = v_1, v_2, \ldots, v_k$ denotamos con ℓ_P la longitud de P y con $\ell_P(v_i, v_j)$, $i \leq j$, la longitud del subcamino de P que va desde v_i hasta v_j.

Solución:

(a) Sea $P' = u_1, u_2, \ldots, u_r$ un camino arbitrario de s a v. Como $s \in S$ y $v \in V \setminus S$, P' recorre una arista (u_i, u_{i+1}) con $u_i \in S$ y $u_{i+1} \in V \setminus S$, es decir, una arista que pertenece a E_S. Descomponiendo P' en tres tramos, uno que llega hasta u_i (eventualmente vacío), otro compuesto por una única arista (u_i, u_{i+1}), y un tercero desde u_{i+1} hasta u_r (eventualmente vacío), tenemos

$$\begin{aligned}
\ell_{P'} &= \ell_{P'}(u_1, u_i) + \ell_{(u_i,u_{i+1})} + \ell_{P'}(u_{i+1}, u_r) && (2)\\
&\geq \ell_{P'}(u_1, u_i) + \ell_{(u_i,u_{i+1})} && (3)\\
&\geq d(u_i) + \ell_{(u_i,u_{i+1})} && (4)\\
&\geq d(u) + \ell_{(u,v)}, && (5)
\end{aligned}$$

donde (3) surge de que las longitudes son positivas y por lo tanto $\ell_{P'}(u_{i+1}, u_r) \geq 0$, (4) sigue de la definición de d y (5) es consecuencia de (1).

Como P' es arbitrario, (5) implica que $d(v) \geq d(u) + \ell_{(u,v)}$. Como por otra parte P es un camino de s a v de longitud $\ell_{P_u} + \ell_{(u,v)}$, que es exactamente igual a $d(u) + \ell_{(u,v)}$, concluimos que $d(v) = d(u) + \ell_{(u,v)}$.

(b) No. En (3) se utilizó el hecho de que las longitudes no son negativas. Como contrajemplo sean $S = \{s\}$ y las aristas (s, v) de longitud 1, (s, y) de longitud 2, y (y, v) de longitud -2. El mínimo en (1) es 1, que se alcanza para la arista (s, v), pero el camino (s, y, v) tiene longitud 0.

Ejercicio 2 (40 puntos)

Un autor de libros infantiles ha escrito un libro de la serie *Elige tu propia aventura* que consta de n capítulos. En este tipo de libros, el lector comienza en el capítulo 1 y, a partir de allí, cada vez que llega al final de un capítulo debe elegir entre **2 o 3 opciones**. Cada una de las opciones lo dirige hacia un capítulo distinto desde donde continuar con la historia, hasta que eventualmente llega a alguno de los *capítulos finales* en los cuales la historia se desenlaza y no se presentan opciones para continuar. El autor ha sido cuidadoso y no hay forma de que se repita un capítulo a lo largo del desarrollo de una historia. Además, en una planilla ha anotado, para cada capítulo v, la lista de capítulos que pueden seguirse inmediatamente a partir de v.

El editor, que está evaluando la calidad del libro, nos pide que determinemos el largo mínimo de las historias (medido en cantidad de capítulos), L, y la **cantidad** de historias diferentes de ese largo, N.

(a) Dé un algoritmo para calcular L y N que admita una implementación cuyo tiempo de ejecución sea $O(n)$. Reescriba cualquier algoritmo que utilice del material teórico del curso.

Sugerencia: Modele el problema mediante un grafo dirigido, donde cada vértice es un entero correspondiente al número de un capítulo, y para cada vértice v calcule la cantidad de caminos de largo mínimo, $N(v)$, desde el vértice 1 hasta el vértice v. Para ello exprese $N(v)$, para $v \neq 1$, en función de los valores $N(u)$ para vértices u que pertenecen al nivel inmediatamente anterior al de v en una recorrida BFS desde el capítulo inicial.

(b) Demuestre que el algoritmo propuesto puede implementarse de forma que el tiempo de ejecución sea $O(n)$. Puede utilizar argumentos estudiados durante el curso sin necesidad de reescribirlos.

Sugerencia: Para analizar la complejidad del algoritmo propuesto note que la cantidad de opciones que se presentan al final de cada capítulo está acotada por una constante.

Solución:

(a) El algoritmo para solucionar el problema está basado en BFS y se presenta en la figura 1.

Denotamos $L(w)$ el nivel de un vértice w en una recorrida BFS desde el vértice 1; $L(w)$ se calcula en la variable level$[w]$ en el algoritmo de la figura 1.

Notamos que $L(w)$ es precisamente la distancia mínima desde el vértice 1 a w y observamos que todo camino de largo mínimo que llega a w directamente desde un vértice t debe estar formado por un camino de largo mínimo de u a t más una arista de t a w. Por lo tanto, en tal camino debe cumplirse que $L(w) = L(t) + 1$. Por otra parte, el conjunto de caminos de largo mínimo hasta w puede partirse en subconjuntos disjuntos según el último vértice que visitan justo antes de llegar a w. De lo expuesto concluimos que se cumple la siguiente relación

$$N(1) = 1\,, \qquad N(w) = \sum_{t \in V : (t,w) \in E, L(w) = L(t)+1} N(t)\,, \quad w \neq 1\,. \tag{6}$$

La recurrencia (6) es justamente la que calcula el algoritmo en los pasos 6 y 22; el ciclo del paso 11 se ejecuta una y solo una vez para cada nodo de un cierto nivel i, lo cual garantiza que todos los términos de la sumatoria en (6) son sumados una y solo una vez para los nodos w que pertenecen al nivel $i + 1$.

Cuando se visita un vértice que representa un capítulo final, lo cual se detecta en el paso 12, ese vértice se agrega al conjunto finales. En el momento en que por primera vez se completa un nivel que contiene uno o más capítulos finales, el ciclo del paso 9 se interrumpe inmediatamente después de incrementar el contador de nivel i. En consecuencia, al terminar este ciclo, $i - 1$ es el menor nivel que contiene capítulos finales, y este es por lo tanto el largo de las historias más cortas posibles, como se calcula en el paso 29. La cantidad de historias de ese largo es la suma de $N(u)$ para todos los capítulos finales u de menor nivel en la recorrida BFS, tal como se calcula en el paso 30.

(b) El análisis de complejidad es idéntico al de BFS (ver capítulo 3 de Kleinberg & Tardos), teniendo en cuenta que en el caso de este ejercicio tenemos $|E| = O(n)$. Solo cabe aclarar que, representando el conjunto finales adecuadamente, por ejemplo mediante una lista enlazada, el paso 13 requiere tiempo

$O(1)$ y se ejecuta no más de n veces, mientras que el paso 30 requiere tiempo $O(n)$. De forma similar, el paso 12 no se ejecuta más de n veces y requiere tiempo $O(1)$ cada una si G se presenta mediante listas de adyacencia.

```
1  Hacer descubierto [t] = false para todo t ∈ V
2  Hacer N [t] = 0 para todo t ∈ V
3  Hacer level [t] = 0 para todo t ∈ V
4  Crear finales como un conjunto vacío
5  Hacer descubierto [1] = true
6  Hacer N [1] = 1
7  Hacer i = 0
8  Inicializar L_0 = {1}
9  while L_i no es vacío and finales es vacío do
10     Crear L_{i+1} vacío
11     foreach t ∈ L_i do
12         if t no tiene aristas salientes then
13             Agregar t a finales
14         else
15             foreach arista (t, w) saliente de t do
16                 if not descubierto [w] then
17                     descubierto [w]=true
18                     level [w]=i+1
19                     Agregar w a L_{i+1}
20                 end
21                 if level [w]=i+1 then
22                     N [w] = N [w] + N [t]
23                 end
24             end
25         end
26     end
27     Incrementar i
28 end
29 Hacer L = i − 1
30 Hacer N = Σ_{u∈finales} N[u]
```

Figura 1: Algoritmo para calcular L y N.

Ejercicio 3 (30 puntos)

Consideramos tres pilas de monedas, p_1, p_2 y p_3. Cada pila tiene una cantidad finita de monedas. Las monedas pueden tener diferentes valores. Disponemos de las operaciones $Valor(p)$, que calcula la suma de los valores de todas las monedas de la pila p, y $eliminarTope(p)$, que elimina la moneda en el tope de la pila p (o no hace nada si p está vacía).

Usando estas operaciones queremos maximizar el valor acumulado entre las tres pilas,

$$V = Valor(p_1) + Valor(p_2) + Valor(p_3)\,,$$

sujeto a la condición de que $Valor(p_1) = Valor(p_2) = Valor(p_3)$.

(a) Supongamos que inicialmente se cumple que $Valor(p_1) < Valor(p_2) < Valor(p_3)$. ¿Es posible construir una solución con $V > 3 \times Valor(p_1)$? Justifique su respuesta.

(b) Dé un algoritmo **ávido** que permita resolver el problema.

(c) Demuestre la corrección de su algoritmo.

Sugerencia: Muestre que la solución construida por su algoritmo *se mantiene por delante* de cualquier otra, en el sentido de que conserva en las pilas a todas las monedas que forman parte de una solución óptima. En otras palabras, nunca se elimina una moneda que pertenece a una solución óptima.

Solución:

(a) El valor máximo que se puede obtener es $3 \times Valor(p_1)$. Como solo se puede eliminar monedas, no es posible aumentar el valor de ninguna de las pilas y solo se puede llegar a equiparar sus valores disminuyendo el valor de algunas de ellas. Por lo tanto, la pila de valor mínimo (en este caso p_1) determina una cota superior para V igual al triple de su valor.

(b) El algoritmo de la figura 2 resuelve el problema.

```
1 Algorithm greedyPilas(p1, p2, p3)
      Input: p1, p2 y p3 son tres pilas de enteros
2     while (Valor(p1) ≠ Valor(p2) or Valor(p1) ≠ Valor(p3) or Valor(p2) ≠ Valor(p3)) do
3         Sea i ∈ arg máx { Valor(pi) : i ∈ {1,2,3} }
4         eliminarTope(pi)
5     end
6     return 3 × Valor(p1)
7 end
```

Figura 2: Algoritmo ávido para equilibrar el valor de las pilas maximizando el valor total

(c) En primer lugar observamos que el algoritmo termina. Como hay una cantidad finita de monedas en cada pila y en cada paso se elimina una moneda de la pila con mayor valor, eventualmente (a más tardar cuando todas las pilas están vacías) los valores de las pilas se igualan, por lo cual no se cumple la condición del ciclo del paso 2 y el algoritmo termina.

Sea $\bar{V}$ el valor acumulado óptimo para una determinada instancia del problema. Vamos a mostrar por inducción en la cantidad de iteraciones del ciclo del paso 2 que la solución construida por el algoritmo *está por delante* de cualquier otra, en el sentido de que a lo largo de toda la ejecución del algoritmo se cumple que $3 \times Valor(p_i) \geq \bar{V}$, para todo $i \in \{1,2,3\}$. Observar que, por los argumentos desarrollados en la parte a, esto es equivalente a decir que el conjunto de monedas que componen una solución óptima es un subconjunto de las monedas que se mantienen en las pilas a lo largo de la ejecución del algoritmo. Cuando aún no se ha realizado ninguna iteración del ciclo la tesis se cumple trivialmente. Asumamos ahora que la tesis se cumple inmediatamente después de completar j iteraciones, $j \geq 0$, y sea i la pila seleccionada en el paso 3 durante la iteración $j+1$. Por la condición del paso 2, existe una pila, p_k, con valor diferente al de p_i y, por la definición de i en el paso 3 y la hipótesis de inducción se cumple que

$$3 \times Valor(p_i) > 3 \times Valor(p_k) \geq \bar{V}\,.$$

La desigualdad estricta implica que la moneda en el tope de la pila p_i no pertenece a una solución óptima, de modo que al retirarla en el paso 4 la tesis se mantiene.

Concluimos que, cuando el algoritmo termina, el valor calculado en el paso 6 es mayor o igual a $\bar{V}$ y, como no se cumple la condición del paso 2, todas las pilas tienen el mismo valor implicando que la solución computada es correcta.