MA4701. Optimización Combinatorial. 2013.
Profesor: José Soto
Escriba(s): Roberto Bobadilla y Álvaro Bustos.
Fecha: Jueves 26 de Septiembre, 2013.

Cátedra 10

1. El Algoritmo de Bellman-Ford

Definición 1. Consideremos $G = (V, E)$ grafo dirigido y $\ell : E \to \mathbb{R}$ una función de largo conservativo (no hay ciclos de largo total negativo). Definimos los conjuntos:

$$\mathcal{W}(u, v) := \{\text{paseos de } u \text{ a } v\},$$
$$\mathcal{P}(u, v) := \{\text{caminos de } u \text{ a } v\}.$$

Recordemos que el principio de optimalidad de Bellman nos asegura que si ℓ es conservativo entonces:

$$d(u, v) = \min_{p \in \mathcal{P}(u,v)} \ell(P) = \min_{p \in \mathcal{W}(u,v)} \ell(P).$$

Para comprobar que la segunda igualdad es efectiva, basta ver que $\leq$ es consecuencia de que todo camino es un paseo y que la desigualdad contraria se tiene porque no hay ciclos negativos.

Sin embargo, d no es métrica. Notemos que, en el caso general, $d(u, v) \neq d(v, u)$ pues el grafo es dirigido; por lo tanto, la existencia de un (u, v)-dicamino no implica la existencia de un (v, u)-camino, y si existen ambos caminos, en general no son iguales y ni siquiera de la misma longitud. En la figura (1) se ve que $d(x, y) = 2$ pero $d(y, x) = 3$. Más aún, $d(a, b) = 7$, pero $d(b, a) = \infty$, así que ni siquiera podemos asegurar que la finitud de $d(x, y)$ implique la de $d(y, x)$.

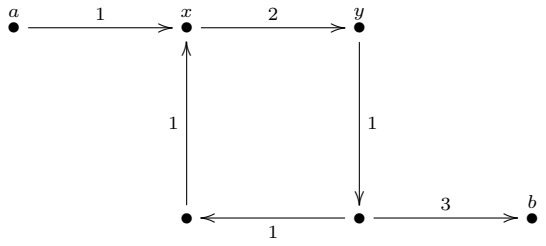

Figura 1: Grafo en que $d(x, y) = 2 \neq 3 = d(y, x)$.

Más aún, como ℓ no es necesariamente positiva, d puede tomar valores negativos. Sin embargo, d posee algunas propiedades similares a las de una función distancia, como vemos a continuación:

Proposición 1. *La función $d : V \times V \to \mathbb{R}$ satisface la desigualdad triangular,*

$$d(u, v) \leq d(u, w) + d(w, v),$$

y, más aún, cumple con la propiedad de optimalidad:

$$d(u, v) = \min_{w \in N^-(v)} d(u, w) + \ell(w, v).$$

Sin embargo, en general no existe una forma «ordenada» de evaluar la última expresión sin usar un algoritmo como Bellman-Ford o similar; puede ser que, para algún trío de vértices u, v, w, dicha expresión para $d(u, v)$ se halle en función de $d(u, w)$ y viceversa.

Recordemos que, para implementar el algoritmo de Bellman-Ford visto en la clase anterior, definimos inductivamente las distancias parciales como sigue:

Definición 2. Para $G = (V, E)$ digrafo con función de largos conservativos $\ell : E \to \mathbb{R}$, definimos las **distancias óptimas con i nodos internos** como sigue:

$$d_i(u, v) = \min_{P \in \mathcal{P}_i(u,v)} \ell(P) = \min_{P \in \mathcal{W}_i(u,v)} \ell(P),$$

en que definimos $\mathcal{W}_i(u, v)$ como el conjunto de todos los (u, v)-paseos con a lo más i nodos internos; análogamente definimos $\mathcal{P}_i$ reemplazando «paseos» por «caminos».

Nótese que, de esta forma, $d(u, v) = d_{n-2}(u, v)$, en que $n = |V|$, y por tanto $d_k = d_{n-2}$ para todo $k \geq n - 2$. Esta igualdad solamente vale cuando ℓ es conservativa; si no lo es, siempre existirá un par $u, v \in E$ para el cual $d_k(u, v) \to -\infty$ cuando $k \to \infty$.

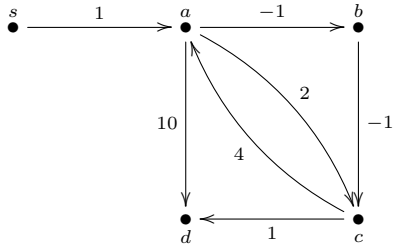

Figura 2: Cálculo sucesivo de $d(x, y)$.

Para poder calcular algorítmicamente d_i, notemos que podemos definir este valor por recursión, ya que evidentemente d_0 coincide con ℓ entre los vecinos de un vértice u dado:

$$d_0(x, y) = \begin{cases} \ell((x, y)) & \text{si } (x, y) \in E, \\ \infty & \text{si no,} \end{cases}$$

e inductivamente:

$$d_{i+1}(u, v) = \min\left(d_i(u, v), \min_{w \in N^-(v)} d_i(u, w) + \ell((w, v))\right).$$

Esta fórmula ahora es «amigable» computacionalmente, siendo la base del algoritmo de Bellman-Ford. La idea detrás de este algoritmo es que, fijando u, llenaremos recursivamente una tabla con los valores de $d_i(u, v)$ para cada v, «corrigiendo» sucesivamente las distancias en cada iteración. Como ejemplo, podemos ver el cálculo de d_i en la figura 1, para $i = 0, 1, 2, 3$:

	s	a	b	c	d
0	0	1	∞	∞	∞
1	0	1	0	3	11
2	0	1	0	-1	4
3	0	1	0	-1	4

1.1. Digrafos acíclicos

Queremos estudiar el caso particular de los digrafos acíclicos (en el sentido de que no poseen ciclos dirigidos; nótese que el grafo no dirigido subyacente puede tener ciclos). En la figura 3 podemos ver un ejemplo de grafo acíclico.

Una propiedad evidente de los digrafos acíclicos es que cualquier función de longitudes $\ell : E \to \mathbb{R}$ es conservativa. Por tanto, la distinción entre «camino minimal» y «paseo minimal» es irrelevante.

Teorema 1. *Si $G = (V, E)$ es un digrafo acíclico, entonces existe un ordenamiento $v_1, ..., v_n$ de V tal que $(v_i, v_j) \in E \implies i < j$. Cualquier orden que satisfaga esta propiedad se llama **orden topológico** de G; se puede obtener un orden topológico en V usando búsqueda en profundidad (por consiguiente, en un tiempo $O(n + m)$).*

Así, esta herramienta nos permite proponer el siguiente algoritmo para encontrar caminos mínimos desde $s \in V$, en un digrafo acíclico G con función de largo ℓ:

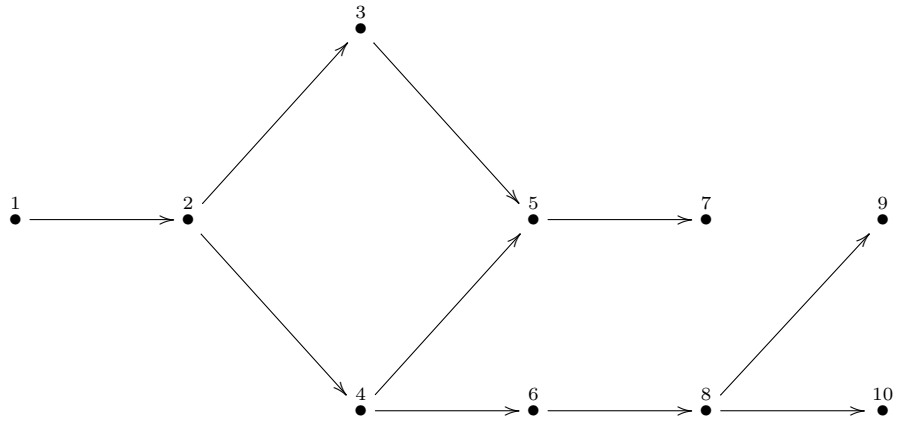

Figura 3: Digrafo sin ciclos dirigidos. Puede observarse el orden topológico en el grafo.

(1) Etiquetamos $v_1, ..., v_n$ en orden topológico.

(2) Si $v_i = s$, entonces $d(s, v_j) = \infty$ para todo $j < i$. Asimismo, fijamos $d(s, v_i) = 0$.
Para los casos restantes, aplicamos la recursión:

$$(\forall j = i+1, ..., n)\, d(s, v_j) = \min_{w \in N^-(v_j)} d(s, w) + \ell(w, v_j).$$

Nótese que $N^-(v_j) \subseteq \{v_1, ..., v_n\}$; así, $d(s, v_j)$ se calcula en función de valores ya calculados (es decir, esto efectivamente constituye un algoritmo recursivo).

(3) Retornamos la matriz $d(s, \cdot)$.

Notemos que, ya que ℓ es conservativo (ya que el grafo es acíclico), y el orden topológico nos asegura que todo (v_i, v_j)-camino (con $i < j$) contiene solo vértices de $\{v_i, v_{i+1}, \ldots, v_j\}$, el siguiente resultado es directo:

Teorema 2. *El algoritmo anterior encuentra un camino óptimo y tiene complejidad $O(m + n)$.*

Para comprobar que el cálculo de la complejidad es correcto, basta notar que el orden de cada paso es:

(1) es $O(m + n)$ pues corresponde al algoritmo de búsqueda en profundidad.

(2) es $O(n)$ (pues se recorrerán todos los vértices) más el tiempo que toma estudiar los valores ya calculados en los vecinos anteriores de cada nodo, que resulta ser $\sum_{j=i+1}^{n} O(d^-(v_j)) = O(m)$.

Es fácil modificar el algoritmo para que también almacene a los predecesores. Esto queda propuesto como ejercicio.

2. Matriz de distancia

Definición 3. Para un digrafo $G = (V, E)$ con función de longitudes ℓ conservativa, definimos su **matriz de distancias** como $d := [d(u, v)]_{(u,v) \in V^2}$. Nótese que d contiene todos los valores de $d(u, v)$ para cualquier par posible de vértices.

Un algoritmo «naïve» para calcular d corresponde a utilizar el algoritmo de Bellman-Ford para calcular $d(u, v)$, para todos los pares de vértices posibles. Notemos que, como Bellman-Ford es $O(nm)$ y lo ejecutamos n veces (ya que evaluamos $d(u, \cdot)$ para todo $u \in V$), este algoritmo es $O(n^2 m)$, que es polinomial pero bastante lento; más aún para grafos «densos» en los que m es $\Theta(n^2)$.

En lo que sigue, usaremos la notación:

$$M_i(u, v) := d_{i+1}(u, v),$$

de modo tal que el término izquierdo corresponda a la distancia de u a v usando i arcos del digrafo G. De este modo, la siguiente identidad corresponde simplemente a reescribir uno de los resultados anteriores:

$$M_i(u,v) = \min_{w \in V} M_{i-1}(u,w) + M_1(w,v).$$

Recordemos que, para dos matrices A, B en $V \times V$ su producto se define de forma análoga a la definición dada en álgebra lineal:

$$A \cdot B(u,v) = \sum_{w \in V} A(u,w)B(w,v).$$

Así también, podemos escoger otro par de operaciones asociativas tales que una distribuya con respecto a la otra, y definir un «producto» generalizado de forma análoga a lo anterior. En particular, la siguiente operación $\wedge$ es asociativa:

$$x \wedge y := \min(x,y),$$

y, más aún, $+$ distribuye sobre $\wedge$, como vemos a continuación:

$$(x \wedge y) + z = \min(x,y) + z = \min(x+z, y+z) = (x+z) \wedge (y+z).$$

De este modo, podemos definir:

Definición 4. Dadas dos matrices A y B en $V \times V$, definimos la operación $\otimes$ mediante la igualdad:

$$(A \otimes B)(u,v) := \min_{w \in V}(A(u,w) + B(w,v)).$$

Podemos ver que esta nueva operación $\otimes$ es asociativa; la demostración es exactamente igual a la que se lleva a cabo en álgebra lineal, reemplazando $\cdot$ por $+$ y cambiando $+$ por $\wedge$. Por tanto, tiene sentido definir «$\otimes$-potencias», mediante la fórmula usual:

$$A^{\otimes 1} := A,\ A^{\otimes(n+1)} := A^{\otimes n} \otimes A,$$

en que usamos la notación $A^{\otimes k}$ para distinguir estas «potencias» de las usuales. De este modo, la igualdad de más arriba para $M_i(u,v)$ puede escribirse de forma breve como:

$$M_i = M_{i-1} \otimes M_1 = M_1^{\otimes i}.$$

Así, para calcular $d = M_j$, con $j \geq n-1$, proponemos el siguiente algoritmo:

(1) Calcular $M = M_1$, de forma análoga a como se hizo previamente para d:

$$M_1(u,v) = \begin{cases} 0 & \text{si } u = v, \\ \ell(u,v) & \text{si } (u,v) \in E, \\ \infty & \text{si no.} \end{cases}$$

(2) Calcular iterativamente los valores de $M^{\otimes 2}, ..., M^{\otimes j}$.

(3) Retornar $M_j = M^{\otimes j}$ al terminar las iteraciones.

Este algoritmo calcula d en tiempo $O(jn^3)$, o bien (recordando que j es $O(n)$), $O(n^4)$. Podemos mejorar el tiempo de ejecución mediante divisiones binarias (de manera similar al algoritmo binario para calcular potencias) utilizando el hecho de que $\otimes$ es un operador asociativo. Para ello, reemplazaremos el paso (2) por:

(2') Calcular iterativamente:

$$M^{\otimes 2}, M^{\otimes 4} = (M^{\otimes 2})^{\otimes 2}, M^{\otimes 8}, \ldots, M^{\otimes 2^k},$$

con $2^k \geq n-1$, de modo que $k \in \Theta(\log(n))$.

De este modo reducimos el tiempo de ejecución del algoritmo de $O(n^4)$ a $O(n^3 \log(n))$. Cabe notar que técnicas similares son aplicables en una variada gama de circunstancias (búsqueda binaria, algoritmos de tipo `QuickSort` o `MergeSort`, etcétera) y muchas veces un proceso secuencial puede reducirse de $O(n)$ a $O(\log(n))$.

3. El método de Floyd-Warshall

Este método es un algoritmo $O(n^3)$ para calcular la matriz de distancia M (para un digrafo G con una función de distancia $\ell : E(G) \to \mathbb{R}$ conservativa). Teniendo la matriz de distancia es fácil encontrar todos los caminos óptimos entre cualquier par de vértices $u, v \in V(G)$. Para ello definiremos la siguiente función auxiliar:

Definición 5. Sea $\{v_1, \ldots, v_n\}$ un ordenamiento cualquiera de los vértices de V. Dado tal ordenamiento, llamaremos $f_i(u, v)$ al largo mínimo de un camino (o paseo) de u a v que use como nodos internos un subconjunto de $\{v_1, \ldots, v_i\}$.

Notemos que f_i, por su misma definición, debe satisfacer la siguiente relación:

$$f_i(u,v) = \text{mín} \left(\begin{array}{l} \text{largo de un } (u,v)\text{-camino} \\ \text{con } v_i \text{ como nodo interno} \end{array} , \begin{array}{l} \text{largo de un } (u,v)\text{-camino} \\ \text{sin } v_i \text{ como nodo interno} \end{array} \right),$$

y que el segundo argumento es $f_{i-1}(u, v)$. Si logramos expresar el primer término en función de f_{i-1} obtendremos una recurrencia que permite calcular f_i.

Para hallar tal expresión, consideremos un (u, v)-camino P óptimo entre todos los que usan nodos internos entre $\{v_1, v_2, \ldots, v_i\}$ y que pasan por v_i. Podemos descomponer P como unión de dos caminos P_1 y P_2, siendo P_1 un camino de u a v_i y P_2 un camino de v_i a v. De este modo, tenemos la relación:

$$\ell(P) = \ell(P_1) + \ell(P_2) = f_{i-1}(u, v_i) + f_{i-1}(v_i, v),$$

en que la relación $\leq$ es directa del hecho de que P es un camino óptimo entre los posibles, y la desigualdad contraria es consecuencia del criterio de optimalidad de Bellman. Así, deducimos el siguiente resultado:

Lema 1. *La función f_i se relaciona con f_{i-1} mediante la relación de recurrencia dada por:*

$$f_i(u, v) = \text{mín}(f_{i-1}(u, v_i) + f_{i-1}(v_i, v), f_{i-1}(u, v)).$$

Usando este lema, podemos finalmente enunciar el algoritmo de Floyd-Warshall:

Algorithm 1 Algoritmo de Floyd-Warshall

Require: $G = (V, E)$ grafo dirigido, $\ell : E \to \mathbb{R}$ función de distancias conservativa.
  $V = \{v_1, \ldots, v_n\}$.
  for $(u, v) \in E \times E$ **do**

$$f_0(u,v) \leftarrow \begin{cases} 0 & \text{si } u = v, \\ \ell(u,v) & \text{si } (u,v) \in E, \\ \infty & \text{en otro caso.} \end{cases}$$

  end for
  for $i \in \{1, \ldots, n\}$ **do**
    $f_i(u, v) \leftarrow \text{mín}(f_{i-1}(u, v_i) + f_{i-1}(v_i, v), f_{i-1}(u, v))$.
  end for
  return $M := f_n$.