Bellman-Ford y Floyd-Warshall

Colectivo Estructuras de Datos y Algoritmos

Noviembre 2021

1. Sean $G = \langle V, E \rangle$ un grafo dirigido y ponderado que no contiene ciclos de costo negativo, $s \in V$ y m el máximo entre todo $v \in V$ del menor número de aristas en un camino de costo mínimo de s a v. Modifique el algoritmo *Bellman-Ford* para que termine correctamente a lo sumo en $m+1$ pasos de recorridos sobre E incluso si m no es conocido de antemano.

Respuesta

Se sabe que en la i-ésima iteración del algoritmo *Bellman-Ford* todo vértice $v \in V$ con un camino de costo mínimo de s a v de longitud a lo sumo i aristas cumple $d[v] = \delta(s, v)$ (aplicación de la propiedad de la convergencia y parte de la demostración del funcionamiento del algoritmo). Por la propiedad anterior y la definición de m se cumple que luego de m pasos todos los vértices están bien calculados, los alcanzables cumplen $d[v] = \delta(s, v)$ y los no alcanzables mantienen el valor $d[v] = \infty$. Entonces es posible terminar el algoritmo *Bellman-Ford* luego de m pasos sin afectar la correctitud. Dado que el valor de m es desconocido no se pueden hacer m pasos directamente. Sin embargo, como en el paso $m+1$ y siguientes del algoritmo los valores de distancia no cambian, al modificar el algoritmo para que ejecute el recorrido sobre E hasta que todos los $d[v]$ mantengan su valor en un paso se garantiza que termina a lo sumo en $m+1$ pasos. La observación de que es a lo sumo $m+1$ y no exactamente $m+1$ es consecuencia de la propiedad del RELAX bajo un orden arbitrario de las aristas. La orden del ejercicio es explícita en cuanto a la inexistencia de ciclos de costo negativo por lo que no es necesario comprobar su existencia.

```
BELLMAN-FORD-(M + 1)(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
cambios ← TRUE
while cambios do
  cambios ← FALSE
  for cada arco (u, v) ∈ E[G] do
    RELAX-M(u, v, w)
  end for
end while

RELAX-M(u, v, w)
if d[v] > d[u] + w(u, v) then
  d[v] ← d[u] + w(u, v)
  π[v] ← u
  cambios ← TRUE
end if
```

2. Modifique el algoritmo *Bellman-Ford* para que asigne -∞ a $d[v]$ en todo vértice v alcanzable desde el origen a través de un ciclo de costo negativo. La modificación no debe alterar la complejidad temporal del algoritmo.

Respuesta

Al realizar un análisis detallado de la segunda mitad de la demostración de la correctitud del algoritmo *Bellman-Ford* se encuentra una propiedad importante:

Lema 1 *Todo ciclo de costo negativo alcanzable desde el origen tiene al menos dos vértices consecutivos* u, v *tal que* $d[v] > d[u] + w(u, v)$.

La demostración del lema es omitida pues está en la conferencia (últimas 2 diapositivas).

Sea S el conjunto de todos los vértices $v \in V$ que en el paso $|V|$ del algoritmo *Bellman-Ford* cumplen la condición $d[v] > d[u] + w(u, v)$. Por el Lema 1 se sabe que S contiene al menos un vértice de cada ciclo de costo negativo alcanzable desde el origen así que todo vértice alcanzable desde el origen a través de un ciclo de costo negativo es alcanzable por los vértices de S.

Una solución consiste en hallar S en $O(E)$ y hacer un *DFS* en $O(V + E)$ partiendo de los vértices de S encontrando así todos los vértices alcanzables desde el origen a través de un ciclo de costo negativo. Queda por demostrar que todos los vértices en S son alcanzables desde el origen a través de un ciclo de costo negativo. Con dicha demostración queda asegurada la correctitud del algoritmo planteado.

Demostración por reducción al absurdo:
Si existe un camino de costo mínimo de s a un vértice $v \in S$ se sabe que existe un camino simple de costo mínimo de s a v. Tal camino tendría una longitud de a lo sumo $|V| - 1$. Luego, por la propiedad de la cota superior sucede que $d[v] = \delta(s, v)$ al finalizar el paso $|V| - 1$ y no varía nunca más entrando en contradicción con $v \in S$. Luego, no existe camino alguno de costo mínimo del origen a los vértices que pertenecen a S.
Además, aquellos vértices v no alcanzables desde el origen mantienen invariante $d[v] = \infty$ por la propiedad de la no existencia de camino así que no serán añadidos a S.
Luego, todos los vértices en S son alcanzables desde el origen a través de un ciclo de costo negativo.

Respecto a la complejidad temporal del pseudocódigo:
El código en las líneas 2 y 8 corresponden a asignar valores iniciales a listas de longitud $|V|$ en $O(|V|)$, de la 3 a la 7 es $O(|V||E|)$ para calcular los caminos de costo mínimo y en de las líneas 9 a la 13 se realiza una iteración sobre el conjunto de aristas $O(|E|)$ para encontrar los vértices pertenecientes a S y hallar los vértices alcanzables desde estos en $O(|V| + |E|)$. Luego, el algoritmo modificado tiene un orden de complejidad temporal total de $O(|V||E|)$, cumpliendo así el requisito del problema relativo a la complejidad temporal.

```
1:  BELLMAN-FORD(G, w, s)
2:  INITIALIZE-SINGLE-SOURCE(G, s)
3:  for i ← 1 hasta |V[G]| − 1 do
4:      for cada arco (u, v) ∈ E[G] do
5:          RELAX(u, v, w)
6:      end for
7:  end for
8:  visited ← [FALSE] × |V|
9:  for cada arco (u, v) ∈ E[G] do
10:     if d[v] > d[u] + w(u, v) y visited[v] = FALSE then
11:         DFS-VISIT(G, v)
12:     end if
13: end for
```

3. Sea $G = \langle V, E \rangle$ un grafo dirigido y ponderado que contiene ciclos de costo negativo. Diseñe un algoritmo que encuentre y liste los vértices de uno de esos ciclos en $O(|V||E|)$.

Respuesta

La idea principal detrás de este problema es que si existe al menos un ciclo de costo negativo entonces es posible encontrar un vértice v alcanzable desde un ciclo de costo negativo (mostrado en el ejercicio 2) y retrocediendo en la lista π desde v siempre se halla un ciclo de costo negativo. Para garantizar encontrar el ciclo, si existe, se crea un vértice s y aristas con costo 0 de s al resto de los vértices. Luego de un análisis detallado de la complejidad se encontrarán con que no es la adecuada para el problema así que para ajustarla a los requerimientos del problema es necesario usar un truco explicado en la respuesta del siguiente ejercicio. Se deja al estudiante la demostración la correctitud de este procedimiento. Puede hacer uso del siguiente lema.

Se define como d_x a la lista d luego de x pasos del recorrido de relajación de las aristas. Por ejemplo, $d_{|V|-1}$ es la lista d justo antes de comprobar la existencia de ciclos de costo negativo.

Lema 2 *Si un vértice v pertenece a un ciclo de costo negativo de longitud k y $d_x[v] \neq \infty$ entonces al realizar k pasos más del recorrido de relajación de las aristas se cumple $d_x[v] > d_{x+k}[v]$.*

Demostración:
Sea $c = \langle v_0, v_1, \ldots, v_k \rangle$, donde $v_0 = v_k$, un ciclo de costo negativo del grafo que es alcanzable desde el origen. Sea x el primer paso donde $d_x[v_0] \neq \infty$, $x < |V|$ porque existe un camino simple del origen a v_0 al ser v_0 alcanzable desde el origen. Aplicando la desigualdad del RELAX en los k pasos siguientes se obtiene:

$$d_{x+i}[v_i] \leq d_{x+i-1}[v_{i-1}] + w(v_{i-1}, v_i) \text{ para todo } i, 1 \leq i \leq k,$$

dado que RELAX(v_{i-1}, v_i, w) en el paso $x+i$ hace que $d_{x+i}[v_i] \leq z + w(v_{i-1}, v_i)$ donde z toma valores en el rango $d_{x+i}[v_{i-1}] \leq z \leq d_{x+i-1}[v_{i-1}]$ dado que la sucesión de valores en la misma posición de la lista d forma una secuencia no creciente por el efecto del RELAX.
Sustituyendo las k desigualdades (la i-ésima dentro de la $(i+1)$-ésima) se tiene:

$$d_{x+k}[v_k] \leq d_x[v_0] + \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Como $v_0 = v_k$,

$$d_{x+k}[v_0] \leq d_x[v_0] + \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

y $\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$ pues c es un ciclo de costo negativo,

$$d_{x+k}[v_0] < d_x[v_0] \qquad \blacksquare$$

4. Sea $G = \langle V, E \rangle$ un grafo dirigido y ponderado. Diseñe un algoritmo que determine para todo $v \in V$ el valor de $\delta^*(v) = \min_{u \in V}\{\delta(u, v)\}$. La complejidad temporal de su algoritmo debe ser $O(|V||E|)$.

Respuesta

La función a optimizar es distinta a la de conocer los valores de los caminos de costo mínimo de un origen al resto de los vértices. En esta ocasión se desea saber para cada vértice v cuál es el costo del camino de menor costo que empieza en un vértice cualquiera y termina en v. La semejanza entre ambos problemas es indiscutible. Nos encontramos frente a un problema de modelación.

Sea $G' = \langle V', E' \rangle$ donde $V' = V + \{s\}$ y $E' = E + \{(s, v_1), (s, v_2), \ldots, (s, v_n)\}$. El vértice s será el origen de los caminos de costo mínimo hacia los vértices y el costo de las aristas que salen de s hacia el resto de los vértices es 0.
Se deja a los estudiantes la demostración de que el camino costo mínimo de s a v en G' es igual $\delta^*(v)$ en G.

Aunque la modelación es correcta la complejidad temporal no lo es:
$O(|V'||E'|) = O((|V|+1)(|E|+|V|)) = O(|V|^2 + |V||E|)$.
Es posible mejorar el orden de complejidad temporal observando que s no pertenece a ciclo alguno por construcción de G' y antes del primer paso $d[s]$ está bien calculado. Luego, toda relajación en las aristas salientes de $d[s]$ luego del primer paso no afecta el valor en d del resto de los vértices por la propiedad de la convergencia.

Otra forma de ver la modelación es inicializar la lista de distancia de todos los vértices en G a 0, en vez de ∞. Tal operación se justifica con que todo vértice tiene un camino de longitud cero que empieza y termina en él. Así eliminamos el vértice y las aristas extras que impiden alcanzar la complejidad temporal deseada. La implementación del algoritmo sería la de conferencia donde el método INITIALIZE-SINGLE-SOURCE(G, s) asigna valor 0 en todas las posiciones de la lista d.

5. Describa cómo detectar la presencia de ciclos de costo negativo utilizando la salida (matriz D) del algoritmo *Floyd-Warshall*.

Respuesta

La solución consiste en buscar valores negativos en la diagonal principal de la matriz D y se apoya en D_x que es la matriz D luego de x iteraciones del recorrido externo.
Existe un ciclo de costo negativo si y solo si $D_n[i,i] < 0$ para algún vértice i:

- $D_n[i,i]$ es el costo de un camino de i a sí mismo así que si $D_n[i,i]$ es negativo entonces existe un camino de i a sí mismo (un ciclo) con costo negativo.
- Sea, si existe, un ciclo de costo negativo con menor cantidad de vértices.
  - Si tiene un solo vértice entonces algún $w(i,i) < 0$ por lo que $D_0[i,i]$ es negativo y como los valores en D nunca se incrementan también es negativo $D_n[i,i]$.
  - Si tiene al menos dos vértices entonces sean k el vértice con mayor índice en el ciclo e i otro vértice del ciclo. $D_{k-1}[i,k]$ y $D_{k-1}[k,i]$ son costos de caminos de costo mínimo correctos porque no están basados en ciclos de costo negativo dado ambos $D_{k-1}[i,k]$ y $D_{k-1}[k,i]$ no incluyen a k como vértice intermedio y k e i están en el ciclo de costo negativo de menor longitud. Como $i \rightsquigarrow k \rightsquigarrow i$ es un ciclo de costo negativo entonces $D_{k-1}[i,k] + D_{k-1}[k,i] < 0$ y $D_k[i,i]$ tiene un valor negativo al ser $D_k[i,i] \leq D_{k-1}[i,k] + D_{k-1}[k,i]$. Luego, dado que los valores en D nunca se incrementan $D_n[i,i] < 0$. ■

6. Sea $\Pi = (\pi_{ij})$ la matriz de antecesores donde π_{ij} es -1 si $i = j$ o no hay camino de i a j, en otro caso π_{ij} es el antecesor de j en algún camino de costo mínimo que empiece en i. Diseñe un algoritmo que calcule Π en $O(|V|^3)$ usando la salida (matriz D) del algoritmo *Floyd-Warshall*.

Respuesta

La solución se apoya en la matriz de adyacencia A del grafo $G = \langle V, E \rangle$. Supongamos que no existen ciclos de costo negativo. Por la desigualdad triangular se sabe que para todo arco $(u,v) \in E$ se cumple $\delta(s,v) \leq \delta(s,u) + w(u,v)$. Sustituyendo la desigualdad con la notación D en tres vértices $i, j, k \in V$: $D[i,j] \leq D[i,k] + w(k,j)$. Si se cumple la igualdad entonces el vértice k es el antecesor de j en un camino de costo mínimo de i a j. Tal valor existe porque si $D[i,j] \neq \infty$ entonces hay un camino de i a j y, por tanto, un último vértice que hizo RELAX en la pareja (i,j).

La complejidad de la encontrar el vértice k de la igualdad es lineal respecto a la cantidad de vértices y la cantidad de parejas de vértices donde se busca el valor de π es $|V|^2$. Luego la complejidad temporal total es $O(|V|^3)$.

```
1: for i ← 1 hasta |V[G]| do
2:    for j ← 1 hasta |V[G]| do
3:       for k ← 1 hasta |V[G]| do
4:          if D[i, j] = D[i, k] + w(k, j) then
5:             π_ij ← k
6:          end if
7:       end for
8:    end for
9: end for
```

7. Sea $G = \langle V, E \rangle$ un grafo dirigido y ponderado. Para todo vértice $i \in V$, se define como subgrafo de antecesores de i en G a $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$ donde $V_{\pi,i} = \{j \in V : \pi_{ij} \neq -1\} \cup \{i\}$ y $E_{\pi,i} = \{(\pi_{ij}, j) : j \in V_{\pi,i} - \{i\}\}$. Demuestra que para todo $i \in V$ el subgrafo de antecesores $G_{\pi,i}$ es un árbol de caminos de costo mínimo con raíz en i.

Respuesta

Falta mencionar en el enunciado que es respecto a la lista Π del algoritmo *Floyd-Warshall*.
Pista: Para demostrar que $G_{\pi,i}$ es acíclico es aconsejable demostrar primero que si $\pi_{ij} = l$ entonces $D[i, j] \geq D[i, l] + w(l, j)$ y luego adaptar la demostración a la de caminos de costo mínimo desde un origen que se encuentra en el lema 24.16 (página 674) del libro *Introduction to Algorithms*, Cormen, 3ra Edición.

8. Sea $G = \langle V, E \rangle$ un grafo dirigido y ponderado. Se define la excentricidad de un vértice v como el mayor costo de todos los costos de los caminos de costo mínimo que terminan en v, o sea:

$$\text{excentricidad}(v) = \max_{u \in V} \{\text{costo del camino de costo mínimo de } u \text{ a } v\}$$

Proponga un algoritmo que encuentre un vértice con la menor excentricidad entre todos los vértices.

Respuesta

Este problema es sencillo al notar que es posible obtener las excentricidades dada la matriz D, salida del algoritmo *Floyd-Warshall*. La excentricidad de un vértice $v \in V$ es $-\infty$ si v pertenece a algún ciclo de costo negativo pues no existe camino de costo mínimo (siempre hay uno más pequeño). La solución del ejercicio 5 muestra cómo averiguar el valor de la verdad del enunciado. Si v no pertenece a un ciclo de costo negativo entonces $D[v, v] = 0$ por lo que la excentricidad de v es mayor o igual a 0. Luego, buscar el máximo valor sobre la columna v de D distinto de ∞ (implica la no existencia de camino) garantiza la respuesta correcta de excentricidad de v pues solo se analizan los caminos de costo mínimo entre los vértices de G y v.

La complejidad temporal es $O(|V|^3)$ pues encontrar la excentricidad de cada vértice es a lo sumo $O(|V|)$ ($(O|V|^2)$ en total) y la complejidad temporal del algoritmo *Floyd-Warshall* es $O(|V|^3)$.

```
1: D ← Floyd-Warshall(G)
2: excentricidad ← [0] × |V[G]|
3: for i ← 1 hasta |V[G]| do
4:    if D[i, i] < 0 then
5:       excentricidad[i] ← −∞
6:    else
7:       for j ← 1 hasta |V[G]| do
8:          if D[j, i] ≠ ∞ then
9:             excentricidad[i] ← max(excentricidad[i], D[j, i])
10:         end if
11:      end for
12:   end if
13: end for
```

9. Sea $G = \langle V, E \rangle$ un grafo dirigido y ponderado. Implemente un algoritmo que encuentre la longitud (cantidad de aristas) de un ciclo de costo negativo en G de menor longitud. La complejidad temporal de su algoritmo debe ser $O(|V|^4)$.

Respuesta

El algoritmo de *Floyd-Warshall* en el paso x sirve para detectar ciclos de costo negativo que usen los x primeros vértices. Sin embargo, no optimiza el resultado por longitud de las aristas. Luego, queda implementar un algoritmo cuya invariante sea que en el paso x los resultados en su matriz de distancia sean los caminos de costo mínimo entre todo par de vértices con a lo sumo distancia x.
Tal algoritmo se llama SLOW-ALL-PAIRS-SHORTEST-PATHS(W) y está explicado en el epígrafe 24.4 (específicamente la página 689) del libro *Introduction to Algorithms*, Cormen, 3ra Edición. Se hace una modificación donde al finalizar cada paso se revisa la diagonal principal en búsqueda de elementos negativos que aseguren la existencia de un ciclo de costo negativo (demostrado en el ejercicio 5) y se devuelve, en caso de que exista el paso en que ocurra. Dicho valor es la longitud de los ciclos de costo negativo de menor longitud.

Sea la matriz L_x, en el pseudocódigo, que representa los caminos de costo mínimo entre todo par de vértices usando a lo sumo x aristas. Además, sea m^* longitud de los ciclos de costo negativo de menor longitud, donde $m^* = \infty$ si el grafo no tiene ciclos de costo negativo.
Asumamos que para algún valor $m^* \leq |V|$ y algún vértice $i \in V$, se cumple que $L_{m^*}[i, i] < 0$. Entonces el grafo tiene un ciclo de m^* aristas que van del vértice i a sí mismo. Tal ciclo es de costo negativo pues hasta el paso $m^* - 1$ todos los caminos de costo mínimo de longitud a lo sumo $m^* - 1$ están bien calculados por la correctitud del algoritmo y al añadir una arista al camino se forma el ciclo cuyo costo está en $L_{m^*}[i, i]$. Esta es la menor cantidad de aristas que tiene un ciclo de costo negativo en G porque el algoritmo computa todos los caminos de costo mínimo de 1 arista, 2 aristas, ..., y todos los ciclos con longitud menor que m^* fueron comprobados anteriormente y no tienen costo negativo.
Si para todo $m \leq n$ no hay valor de i para el que $L_m[i, i] < 0$ entonces no existen ciclos de costo negativo en G pues todos los ciclos simples tienen longitud a lo sumo $|V|$.

El orden de complejidad temporal de la modificación igual a la del algoritmo, $O(|V|^4)$ o más precisamente $O(|V|^3 * min(|V|, m^*))$, puesto que añade una comprobación lineal fuera y dentro del ciclo exterior del algoritmo, en total $O(|V|^2)$.

```
1: SLOW-ALL-PAIRS-SHORTEST-PATHS(G)
2: A ← matriz de adyacencia de G
3: L_1 ← A
4: for i ← 1 hasta |V[G]| do
5:    if L_1[i, i] < 0 then
6:       return 1
7:    end if
8: end for
9: for m ← 2 hasta |V[G]| do
10:   L_m ← EXTEND-SHORTEST-PATHS(L_{m−1}, A)
11:   for i ← 1 hasta |V[G]| do
12:      if L_m[i, i] < 0 then
13:         return m
14:      end if
15:   end for
16: end for
17: return ∞
```

10. Un sistema de n variables y m restricciones es factible si existe una asignación de valores en las variables tal que todas las restricciones se cumplan. Sean n variables reales $v_1, v_2, ..., v_n$ y m restricciones de diferencias de la forma $v_i - v_j \leq b$ donde b pertenece a los enteros. Diseñe un algoritmo que determine si el sistema de desigualdades es factible.

Respuesta

Este problema es muy interesante pues se resuelve con el algoritmo *Bellman-Ford* en $O((n+1)(m+n))$ o haciendo el truco explicado en el ejercicio 2 en $O(nm)$. El problema y su solución se encuentran explicados en sumo detalle en el epígrafe 24.4 del libro *Introduction to Algorithms*, Cormen, 3ra Edición, por lo que no se hace una transcripción literal del libro.