Componentes fuertemente conexas

Colectivo Estructuras de Datos y Algoritmos

Octubre 2020

1. ¿Cómo se puede afectar el número de componentes fuertemente conexas de un grafo si se añade o se elimina un arco?

Respuesta

Analicemos primeramente qué pasa cuando se añade un arco al grafo G. Noten que esta acción no puede 'separar' componentes fuertemente conexas que ya existían, puesto que añadir un arco solo crea nuevos caminos en G. Entonces lo único que puede pasar es que varias de las SCC que había, se unan en una sola. Pueden ocurrir dos cosas:

a) Que el arco que estemos añadiendo una a dos vértices de la misma componente fuertemente conexa, en cuyo caso el número de componentes fuertemente conexas se mantiene exactamente igual.

b) Que el arco que estamos añadiendo une a dos vértices en componentes fuertemente conexas distintas.

Veamos qué sucede en el segundo caso. Ubiquémonos en el grafo reducido de G. Sea $\langle u, v\rangle$ el arco que estamos añadiendo y v_i y v_j los vértices que representan a sus respectivas componentes fuertemente conexas en el grafo reducido de G. Entonces, si pensamos en ubicar el arco $\langle u, v\rangle$ en G, esto implica la existencia de un arco que une v_i y v_j en G^{SCC}. Recuerden que el grafo reducido es un DAG, luego no tiene ciclos. Al añadir el arco $\langle v_i, v_j\rangle$ pueden crearse varios ciclos en G^{SCC}, a los cuales pertenecen v_i y v_j. Noten que todos los vértices que estén en un ciclo al cual pertenece el arco $\langle v_i, v_j\rangle$, van a pertenecer ahora a la misma componente fuertemente conexa. Si no se crea ningún ciclo con la adición del arco $\langle v_i, v_j\rangle$, entonces el número de componentes fuertemente conexas se mantiene igual. ■

Veamos ahora qué pasa cuando se elimina un arco de G. Noten que si esto sucede, no puede disminuir el número de componentes fuertemente conexas, porque eliminar un arco nunca crea caminos nuevos. Análogamente a lo que razonamos anteriormente, el número de componentes fuertemente conexas puede aumentar arbitrariamente al quitar un arco, en dependencia de los ciclos que 'cerraba' dicho arco. Por ejemplo, un grafo formado por un solo ciclo tiene una sola componente fuertemente conexa. Al quitar un arco cualquiera, se forman $|V|$ componentes fuertemente conexas.

2. Demuestre que para cualquier grafo se cumple que $((G^T)^{SCC})^T = G^{SCC}$, o sea que el grafo transpuesto del grafo reducido de G^T es igual al grafo reducido de G.

Respuesta

En aras de ser lo más formales posible, recordemos que un grafo dirigido es un par de conjuntos $\langle V, E\rangle$. Para demostrar la igualdad entre dos grafos, demostremos que ambos conjuntos son iguales. Sea $((G^T)^{SCC})^T = \langle V_1, E_1\rangle$ y $G^{SCC} = \langle V_2, E_2\rangle$.
Demostremos que $V_1 = V_2$.

Noten que el grafo G^T tiene exactamente las mismas componentes fuertemente conexas que G, ya que al cambiar todos los arcos de G, todo camino que iba de u a v ahora va de v a u y viceversa. Luego, un par de vértices u, v que estaban en la misma componente fuertemente conexa en G, están también en la misma componente fuertemente conexa en G^T. Esto es:

$$V((G^T)^{SCC}) = V(G^{SCC})$$

Además, aplicarle el transpuesto a un grafo no cambia su conjunto de vértices. Luego:

$$V_2 = V(((G^T)^{SCC})^T) = V((G^T)^{SCC}) = V(G^{SCC}) = V_1$$

■

Ahora, demostremos que $E_1 = E_2$. Para ello vamos a demostrar que $\forall \langle u, v\rangle \in E_1$, se cumple que $\langle u, v\rangle \in E_2$ (y viceversa).

($\Rightarrow$)

Sea $\langle u, v\rangle \in E_1$. Como $V_1 = V_2$ entonces en V_2 también existen los vértices u, v, que además corresponden a las mismas componentes fuertemente conexas que u y v en V_1. O sea, todos los vértices que están en la componente fuertemente conexa asociada a u en $((G^T)^{SCC})^T$, también pertenecen a la componente fuertemente conexa asociada a u en G^{SCC}.

Ahora, si existe el arco $\langle u, v\rangle \in E_1$, entonces existe el arco $\langle v, u\rangle \in E((G^T)^{SCC})$. Y si existe dicho arco y sean los vértices y y x, que pertenecen a las componentes fuertemente conexas asociadas a v y u respectivamente, entonces existe en G^T un camino que va de y a x. Esto implica que en G existe un camino que va de x a y, por lo que también existe el arco $\langle u, v\rangle$ en G^{SCC}. ■

($\Leftarrow$)

(Queda indicado completar esta parte de la demostración, que es análoga al razonamiento hecho arriba)

3. Se dice que un grafo dirigido $G = \langle V, E\rangle$ es **simplemente conexo** si, para todo par de vértices $u, v \in V$, existe en G un camino de u a v o un camino de v a u. Proponga un algoritmo que dado un grafo dirigido $G = \langle V, E\rangle$ permita determinar en $O(|E|)$ si G es simplemente conexo.

Respuesta

Para garantizar que un grafo sea simplemente conexo, es necesario que para todo par de vértices $u, v \in V$, existe en G un camino de u a v o un camino de v a u.

Noten que todos los pares de vértices ubicados en una misma componente fuertemente conexa del grafo cumplen con esta condición. Además, dos vértices en dos componentes fuertemente conexas distintas lo cumplen si y solo si existe un camino entre los nodos que representan dichas componentes en el grafo reducido. Por lo que el problema se reduce a analizar si para todo par de vértices u, v del grafo reducido de G existe un camino de u a v (o de v a u). Como no pueden ser ambos, puesto que el grafo reducido es un DAG, este problema es equivalente a determinar si G^{SCC} es semiconexo (ejercicio 6 de la clase práctica anterior).

```
1: G^SCC ← grafo reducido de G
2: return semiconexo(G^SCC)
```

Pero planteado de esta forma, la solución sería $O(|V| + |E|)$, puesto que el algoritmo que obtiene el grafo reducido es $O(|V| + |E|)$. Sin embargo, una pequeña observación nos permite darnos cuenta de que si G tiene menos de $|V| - 1$ arcos, entonces no puede ser simplemente conexo, porque con menos de $|V| - 1$ arcos, el grafo subyacente no es conexo; en cuyo caso, en G necesariamente existirían al menos dos vértices que estarían desconectados en ambos sentidos.

Entonces, ajustando un poco el pseudocódigo a la idea anterior:

```
1: if |E| < |V| − 1 then
2:     return False
3: else
4:     G^SCC ← grafo reducido de G
5:     return semiconexo(G^SCC)
6: end if
```

Luego de este cambio, el cuerpo del **else** solo se ejecuta cuando $|E| \geq |V|-1$, por lo que $|V|+|E| \in O(|E|)$.

4. En un grafo dirigido $G = \langle V, E\rangle$ se dice que un vértice v está **hundido** cuando para todo vértice w alcanzable desde v, se cumple que v también es alcanzable desde w. Se define el **fondo** de un grafo como el conjunto de vértices $H = \{v \mid v \in V(G) \text{ y } \mathit{hundido}(v)\}$. Diseñe un algoritmo que dado un grafo dirigido $G = \langle V, E\rangle$ permita determinar en $O(|V| + |E|)$ el conjunto H definido anteriormente.

Respuesta

Primeramente analicemos qué vértices son los que están hundidos en un grafo. Noten que todo vértice w en la misma componente fuertemente conexa de v es alcanzable desde v y viceversa. Por tanto, cualquier vértice u situado en otra componente fuertemente conexa, si es alcanzable desde v entonces v no es alcanzable desde u, porque si lo fuera estuvieran en la misma componente fuertemente conexa.
A partir de este razonamiento, noten que todos los vértices que pertenezcan a una componente fuertemente conexa tal que su nodo respectivo en el grafo reducido tenga *outdegree* 0 cumplen que están hundidos. Esta condición es además necesaria y suficiente.

Demostración: Sea G el grafo original y G^{SCC} su grafo reducido.
($\Rightarrow$)
Sea v un vértice en la componente fuertemente conexa C_i, y sea v_i el nodo correspondiente a C_i en el grafo reducido. Sabemos que $\mathit{outdegree}(v_i) = 0$, entonces no existe ninguna componente fuertemente conexa alcanzable desde v_i, por tanto ningún vértice alcanzable desde v puede estar en otra componente fuertemente conexa. Ahora, sea u un vértice cualquiera de G distinto de v y alcanzable desde v. Según lo anterior, necesariamente $u \in C_i$. Entonces por definición de componente fuertemente conexa, v es alcanzable desde u. Finalmente, v está hundido.

($\Leftarrow$)
Sea v un vértice hundido cualquiera, C_i su componente fuertemente conexa y v_i el vértice asociado en G^{SCC}. Asumamos que $\mathit{outdegree}(v_i) > 0$. Sea v_j tal que $\langle v_i, v_j\rangle \in E(G^{SCC})$ y sea w un vértice cualquiera de la componente fuertemente conexa asociada a C_j. Noten que w es alcanzable desde v. Además, v no es alcanzable desde w porque si lo fuera estarían en la misma componente fuertemente conexa.
Llegamos a una contradicción, por lo que lo asumido es falso. Por tanto $\mathit{outdegree}(v_i) = 0$. ■

Esta propiedad nos da un algoritmo sencillo para determinar el conjunto de vértices hundidos en un grafo G.

```
1: result ← lista vacía
2: G^SCC ← grafo reducido de G
3: for C_i ∈ V(G^SCC) tal que outdegree(C_i) = 0 do
4:     for v ∈ C_i do
5:         result.append(v)
6:     end for
7: end for
8: return result
```

Lo complejidad temporal de este algoritmo la podemos analizar de la siguiente manera. La línea 2 es

$O(|V| + |E|)$. Mientras, los ciclos anidados que se declaran en las líneas 3 y 4, tienen una complejidad de $|V|$, puesto que visitan una y solo una vez a cada vértice del grafo (ya que cada vértice se encuentra en una y solo una componente fuertemente conexa). Por regla de la suma, el tiempo de ejecución del algoritmo es $O(|V| + |E|)$.

5. Sea $G = \langle V, E\rangle$ un grafo dirigido. Se dice que un arco $e = \langle u, v\rangle$ es fuerte si y solo si existe un camino de v a u en G, de lo contrario se dice que e es débil. Diseñe un algoritmo que permita, dado un grafo dirigido $G = \langle V, E\rangle$, determinar el camino con mayor cantidad de arcos débiles en G (en caso de existir varios, uno cualquiera) en $O(|V| + |E|)$.

Respuesta

Al igual que para los problemas anteriores, tratemos de encontrar alguna reducción de este problema relacionada con las componentes fuertemente conexas de un grafo. Noten que todo arco $\langle u, v\rangle$ entre dos vértices pertenecientes a la misma componente fuertemente conexa, es un arco fuerte, porque tiene que existir en G un camino de v a u. Pero, si u y v están en componentes fuertemente conexas distintas, entonces $\langle u, v\rangle$ tiene que ser débil, puesto que no podría existir un camino de v a u. Por tanto, un arco es fuerte si y solo si une a dos vértices en la misma componente fuertemente conexa.

¿Qué significa esto en términos del grafo reducido de G? Bueno, la propiedad que tiene que notar es que por cada camino en G con k arcos débiles existe un camino en G^{SCC} de longitud k. Y viceversa, para todo camino en G^{SCC} de longitud k existe un camino en G con k arcos débiles. Expliquemos un poco mejor esta idea.

Sea $C = v_1, v_2, ..., v_m$ un camino de longitud m en G con k arcos débiles. Y sea $D = e_1, e_2, ..., e_k$ el conjunto de sus arcos débiles. Noten que un arco débil $e_x = \langle v_i, v_{i+1}\rangle$ conecta a dos vértices en distintas componentes fuertemente conexas. Entonces, todos los vértices que se encuentran entre dos arcos débiles de C pertenecen a la misma componente fuertemente conexa. Luego, sea el camino C^{SCC} de G^{SCC}, donde cada vértice es aquel de la componente asociada a un conjunto maximal de vértices consecutivos de C que están en la misma componente fuertemente conexa, y sus arcos los arcos débiles en C, de tal forma que si un arco débil de C conecta a los vértices v_i, v_{i+1}, entonces en C^{SCC} conecta a los vértices asociados a las componentes fuertemente conexas de v_i, v_{i+1} respectivamente. Naturalmente, $|C^{SCC}| = k$.

En el otro sentido, para cada camino C^{SCC}en G^{SCC} de longitud k, podemos encontrar un camino en G con k arcos débiles. Si los vértices de C^{SCC} representan a las componentes $C_1, C_2, ..., C_k$ donde C_i está asociada al vértice i-ésimo en C^{SCC}. Entonces, C se forma de la siguiente manera:

Cada componente C_i se sustituye por dos vértices $u_i, v_i \in C_i$ de tal forma que existan en G los arcos $\langle u_{i-1}, u_i\rangle$, si $i > 0$; y $\langle v_i, v_{i+1}\rangle$, si $i < k$. Se agregan $\forall i > 0$ los arcos $\langle u_{i-1}, u_i\rangle$ y $\forall i < k$ los arcos $\langle v_i, v_{i+1}\rangle$. Y para unir los vértices u_i, v_i se agrega uno de los caminos (sus vértices y arcos) que van de u_i a v_i en C_i.

Esta biyección reduce el problema a encontrar el camino de mayor longitud en el grafo reducido de G. Hallar el camino de mayor longitud en un DAG fue resuelto en la clase práctica anterior.

6. Sea $G = \langle V, E\rangle$ un grafo dirigido y ponderado por una función de costo $w : E \to \mathbb{R}$ tal que $w(e = \langle u, v\rangle) = 0$ si existe un camino de v a u, de lo contrario es un número real cualquiera. Plantee un algoritmo que dado un vértice s y un conjunto de vértices T encuentre el camino de mayor costo en G desde el vértice de origen s hacia un vértice $t \in T$, en $O(|V| + |E|)$.

Respuesta

Observaciones

- Sea $e = \langle u, v\rangle \in E(G)$, si u y v pertenecen a la misma componente fuertemente conexa $SCC_i(G) \Rightarrow w(e) = 0$ y en otro caso $w(e) = x_e$. Si e pertenece a una componente fuertemente conexa entonces

se cumple que existe un camino de v a u, y si es un arco que une dos componentes fuertemente conexas no existe un camino entre v y u. Por tanto $w(e) = 0 \Leftrightarrow e \in SCC_i(G)$

- Sea $GR = G^{SCC}$, $u, v \in V(G)$, $scc_i = SCC[u], scc_j = SCC[v]$ el costo del camino de u a v es exactamente igual al costo del camino de scc_i a scc_j en GR siguiendo los arcos que unen las componentes fuertemente conexas que componen el camino de u a v en G, debido a que los arcos internos a una componente fuertemente conexa tienen costo 0.
- El camino de mayor costo en G de s a v es igual al camino de mayor costo de $SCC[s]$ a $SCC[v]$ en GR
- Como GR es un **DAG** se puede calcular el valor del camino de costo máximo desde $SCC[s]$ a los restantes vértices siguiendo la técnica de **Dinámica en DAG** vista en la clase práctica pasada, usando la función F sobre los vértices de GR, donde $ss = SCC[s]$:

$$F(v) = \begin{cases} 0 & \text{si } v = ss \\ -\infty & \text{si } indegree(v) = 0 \\ \text{máx}_{u_i \mid \langle u_i, v\rangle \in E} \left(F(u_i) + w(\langle u_i, v\rangle)\right) & \text{en otro caso} \end{cases}$$

- Luego el camino de costo máximo que empieza en s en G es el camino de costo máximo que empieza en $SCC[s]$ en GR.

El pseudocódigo a continuación brinda solución a este problema, reflejando las observaciones realizadas anteriormente y el uso de la técnica vista en la clase anterior. Por el principio de la suma este algoritmo es $O(|V| + |E|)$.

```
1: GR ← G^SCC
2: ss ← SCC[s]
3: ts ← TopologicalSort(GR)
4: f[v] ← −∞  ∀v ∈ V(GR)
5: f[ss] = 0
6: for i = 0 to ccs − 1 do
7:    v ← ts[i]
8:    for u_i tal que ⟨u_i, v⟩ ∈ E do
9:       if f[u_i] ≥ 0 then
10:         f[v] ← máx(f[v], w(⟨u_i, v⟩) + f[u_i])
11:      end if
12:   end for
13: end for
14: return max(f)
```

7. En una ciudad donde todos los habitantes tienen teléfonos celulares, la compañía de telecomunicaciones impone un protocolo de mensajería que consiste en que todo mensaje que le llegue a un habitante se le reenvía directamente a cada uno de los contactos que estén en el teléfono del mismo. El protocolo garantiza, además, que a cada habitante no le llegue el mismo mensaje más de una vez (debido a que antes de enviarlo consulta si este ya lo tiene). El gobernador de la cuidad necesita dar un comunicado a toda la población, pero por razones que se desconocen, las tarifas de mensajería de la compañía de telecomunicaciones son muy costosas y el gobernador necesita ahorrar. Él desea saber la mínima cantidad de mensajes que tiene que mandar para que toda la población se entere del comunicado. Diseñe un algoritmo que permita al gobernador, dada la lista de contactos de cada habitante de la ciudad, conocer la menor cantidad de mensajes necesarios para garantizar que todos los habitantes obtengan la información (partiendo del protocolo de mensajería de la compañía de telecomunicaciones).

Respuesta

En este tipo de problemas lo primero es modelar computacionalmente el mismo, haciendo todas las aclaraciones necesarias y llevando el problema a una estructura matemática sobre la cual se pueda resolver.
Sea G un grafo dirigido donde los teléfonos celulares son nodos y existe un arco del nodo u al v si v pertenece a la lista de contactos de u. De esta forma queda definida una instancia del problema, como un grafo dirigido.
Observaciones:

- Mandar un mensaje a un nodo v equivale (por la descripción del problema) a que todos los nodos alcanzables desde v también les llegue ese mensaje.
- La menor cantidad de mensajes a mandar para que a todos los nodos les llegue el mensaje es equivalente a minimizar el conjunto S tal que todos los nodos que puedan ser alcanzados desde los nodos de S sea $V(G)$.
- Si en una componente fuertemente conexa a un nodo se le manda un mensaje todos los nodos de esa componente tienen el mensaje, por lo que sería conveniente englobar a todos los nodos de una componente en un solo nodo, para esto nos auxiliaremos de G^{SCC}.
- Sea $GR = G^{SCC}$, en este grafo todos los nodos con $indegree(scc_u) = 0$ necesitan un mensaje individual para cada uno, ya que no existe $v \in G, v \notin scc_v$ tal que exista un camino de v a algún vértice de scc_u en G.
- Sea s una solución al problema, por lo visto anteriormente $s \geq |\{scc_u \in V(GR) \mid indegree(scc_u) = 0\}|$

Sea $GR = G^{SCC}$, $S = |\{scc_u \in GR \mid indegree(scc_u) = 0\}|$, $S^c = V(GR) - S$, se cumple que $\forall v \in S^c \Rightarrow \exists u \in S$ tal que existe un camino desde u hasta v.

Demostración:
Se asume que $\exists v \in S^c$ y $\neg\exists u \in S$ tal que existe un camino desde u hasta v, sea $v_x \in V(GR)$ tal que exista un camino de v_x a v y la longitud del camino es la máxima de todos los que llegan a v, $v_x \notin S$ por lo que $indigree(v_x) \neq 0$ y al ser GR acíclico, existe un vértice v_y tal que $\langle v_y, v_x \rangle \in E(GR)$ y no pertenece al camino de v_x a v, contradiciendo el hecho de que este sea el camino máximo que termina en v. Luego por reducción al absurdo se cumple que $\forall v \in S^c \Rightarrow \exists u \in S$ tal que existe un camino desde u hasta v.

Por tanto mandando los mensajes a los nodos del conjunto S se alcanzan todos los nodos de GR y a su vez todos los nodos de G. Luego como esta es una solución que satisface los requerimientos del problema, se cumple que $s \leq |S|$.
Como se tiene que $|S| \leq s \leq |S| \Rightarrow s = |S|$.
El pseudocódigo siguiente brinda solución a este problema. Por el principio de la suma este algoritmo es $O(|V| + |E|)$.

```
1: GR ← G^SCC
2: return |{scc_u ∈ V(GR) | indegree(scc_u) = 0}|
```

8. En una cueva, dividida en secciones, existen tesoros escondidos con distintos valores. Existe un conjunto de secciones en las que se puede entrar a la cueva y otro en las que se puede salir de la misma. Entre las secciones existen túneles que conectan unidireccionalmente las mismas. En estas secciones pueden existir tesoros con un valor positivo. Diseñe un algoritmo que permita determinar la mayor ganancia que se puede obtener en un recorrido sobre esta cueva dado que necesariamente hay que entrar por una de las secciones de entrada y salir por una de las de salida (la ganancia está determinada por la suma de los valores de todos los tesoros recogidos).

Respuesta

Modelación del problema:
Sea G un grafo dirigido, donde cada nodo es una sección de la cueva, existe un arco entre el nodo u y v si existe un túnel entre la sección u y la sección v. Donde existe una función $f(v)$ que para cada $v \in V(G)$ contiene el valor del tesoro que existe en esta sección, en caso de no existir tiene valor 0. Sea S el conjunto de nodos desde los que se pueden entrar a la cueva y T el conjunto de nodos desde los que se puede salir.
Observaciones:

- La solución del problema es un camino $C = \{c_i\} = \{c_0 ..., c_t\}$ donde $c_0 \in S$, $c_t \in T$, y $F(C) = \sum_{i=0}^{t} f(c_i)$ es máxima.
- Desde un nodo u se pueden recoger todos los tesoros de todos los nodos que estén en la misma componente fuertemente conexa en la que se encuentra u, por tanto como la solución exige maximizar la suma de todos los costos, si se visita la componente fuertemente conexa scc_u es óptimo pasar por todos los vértices que estén contenidos en esta, ya que todos los costos son positivos y no importa repetir arcos.

Sea $GR = G^{SCC}$,

$$f(scc_v) = \sum_{v \in scc_v} f(v)$$

,

$$T_{GR} = \{scc_u \mid \exists u \in T, u \in scc_u\}$$

$$S_{GR} = \{scc_u \mid \exists u \in S, u \in scc_u\}$$

una solución del problema es un camino $Cs = \{cs_i\} = \{cs_0 ..., cs_p\}$ en GR donde $cs_0 \in S_{GR}$, $cs_p \in T_{GR}$, y $Fs(C) = \sum_{i=0}^{p} f(cs_i)$ es máxima.
Es común en estos casos, donde se quiere partir de un conjunto de vértices y llegar a otro conjunto, el uso de vértices virtuales en el grafo donde uno este conectado a todos los iniciales y otro a todos finales.
Sea $GR' = \langle V(GR) + \{s\} + \{t\}, E(GR) + \langle s, su\rangle_{su \in S} + \langle tu, t\rangle_{tu \in T}\rangle$, donde $f(s) = f(t) = 0$.
Luego el camino $Cs = \{cs_i\} = \{cs_0, ..., cs_p\}$ en GR tiene un equivalente en GR': $Cs' = \{s, cs_0, ..., cs_p, t\}$ con el mismo costo.
Este problema se resuelve usando la técnica de **Dinámica en DAG** con la diferencia que ahora el valor esta en los nodos, donde se quiere calcular el camino de costo máximo desde s hasta t en GR' y el costo del camino es Fs. Se define la función que resuelve el camino de costo máximo desde s a $v \in V(GR')$ de la siguiente forma:

$$FF(v) = \begin{cases} 0 & \text{si } v = s \\ -\infty & \text{si } indegree(v) = 0 \\ \text{máx}_{w_i \mid \langle w_i, v\rangle \in E} FF(w_i) + f(v) & \text{en otro caso} \end{cases}$$

El pseudocódigo siguiente brinda solución al problema y en este están contenidas todas las observaciones realizadas anteriormente. Por el principio de la suma este algoritmo es $O(|V| + |E|)$.

```
1: GR ← G^SCC
2: GR' ← ⟨V(GR) + {s} + {t}, E(GR) + ⟨s, su⟩_{su∈S} + ⟨tu, t⟩_{tu∈T}⟩
3: ts ← TopologicalSort(GR')
4: FF[v] ← −∞  ∀v ∈ V(GR')
5: FF[s] = 0
6: for i = 0 to ccs − 1 do
7:    v ← ts[i]
8:    for u_i tal que ⟨u_i, v⟩ ∈ E(GR') do
9:       if FF[u_i] ≥ 0 then
10:         FF[v] ← máx(FF[v], f(v) + FF[u_i])
11:      end if
12:   end for
13: end for
14: return  FF[t]
```