

Digrafos acíclicos y Kosaraju

Algoritmos y Estructuras de Datos III

Santiago Cifuentes

Departamento de computación
FCEN – UBA

Abril 2023

Plan

- ① Digrafos acíclicos y órdenes topológicos.
- ② Algoritmo basado en DFS para encontrar un orden topológico.
- ③ Componentes fuertemente conexas.
- ④ Algoritmo de Kosaraju para determinar componentes fuertemente conexas.

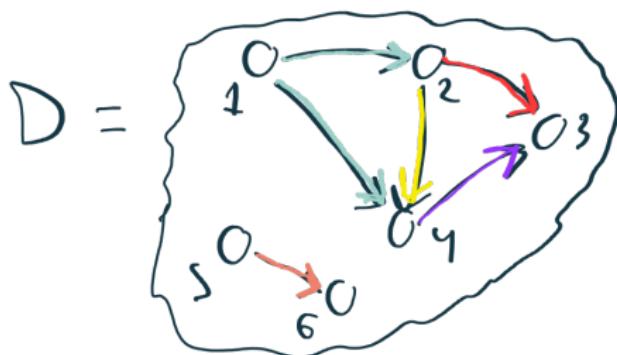
Digrafos acíclicos y orden topológico

Enunciado

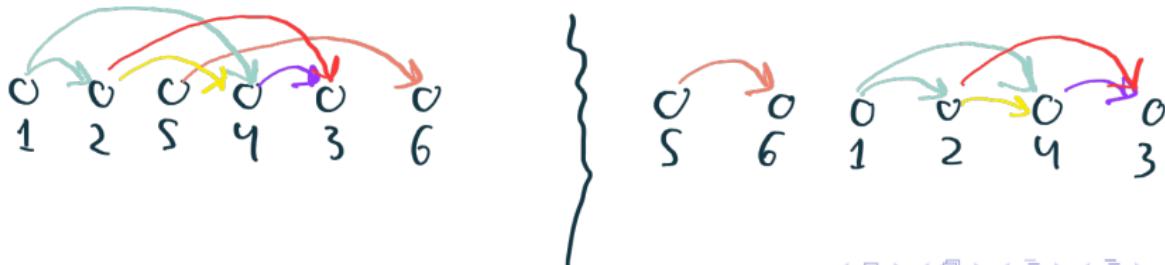
Dado un digrafo D , un orden topológico de D es un ordenamiento $v_1 \dots v_n$ de sus nodos que cumple que todo eje queda de la forma $v_i v_j$ con $i < j$.

Probar que un grafo tiene un orden topológico si y solamente si no tiene ciclos.

Dibujo: orden topológico



Dos posibles órdenes topológicos



Idea

- Para la ida, si el grafo tiene ciclos, entonces ningún orden de los nodos va a poder evitar que quede un eje $v_i v_j$ con $j > i$.

Idea

- Para la ida, si el grafo tiene ciclos, entonces ningún orden de los nodos va a poder evitar que quede un eje $v_i v_j$ con $j > i$.
- Para la vuelta, si quisiéramos armar el orden... ¿Qué tiene que cumplir el primer nodo del orden?

Idea

- Para la ida, si el grafo tiene ciclos, entonces ningún orden de los nodos va a poder evitar que quede un eje $v_i v_j$ con $j > i$.
- Para la vuelta, si quisiéramos armar el orden... ¿Qué tiene que cumplir el primer nodo del orden? Tiene que tener grado de entrada 0 ¿Siempre existe un nodo con esta propiedad?

Idea

- Para la ida, si el grafo tiene ciclos, entonces ningún orden de los nodos va a poder evitar que quede un eje $v_i v_j$ con $j > i$.
- Para la vuelta, si quisiéramos armar el orden... ¿Qué tiene que cumplir el primer nodo del orden? Tiene que tener grado de entrada 0 ¿Siempre existe un nodo con esta propiedad?
- Si el grafo no tiene ciclos, entonces hay un nodo con grado de entrada 0. Ese nodo puede ser el primero del orden topológico

Idea

- Para la ida, si el grafo tiene ciclos, entonces ningún orden de los nodos va a poder evitar que quede un eje $v_i v_j$ con $j > i$.
- Para la vuelta, si quisiéramos armar el orden... ¿Qué tiene que cumplir el primer nodo del orden? Tiene que tener grado de entrada 0 ¿Siempre existe un nodo con esta propiedad?
- Si el grafo no tiene ciclos, entonces hay un nodo con grado de entrada 0. Ese nodo puede ser el primero del orden topológico ¿Y luego?

Idea

- Para la ida, si el grafo tiene ciclos, entonces ningún orden de los nodos va a poder evitar que quede un eje $v_i v_j$ con $j > i$.
- Para la vuelta, si quisiéramos armar el orden... ¿Qué tiene que cumplir el primer nodo del orden? Tiene que tener grado de entrada 0 ¿Siempre existe un nodo con esta propiedad?
- Si el grafo no tiene ciclos, entonces hay un nodo con grado de entrada 0. Ese nodo puede ser el primero del orden topológico ¿Y luego?
- Luego el resto del grafo restante tampoco tiene ciclos, y por lo tanto contiene un nodo con grado de entrada 0. Ese puede ser el segundo nodo del orden topológico. Inducción...

Demostración

- Formalizando, sea $P(k)$ la proposición “Todo dígrafo acíclico de k nodos tiene un orden topológico”. Vale $P(1)$.

Demostración

- Formalizando, sea $P(k)$ la proposición “Todo dígrafo acíclico de k nodos tiene un orden topológico”. Vale $P(1)$.
- Probemos $P(n) \implies P(n + 1)$: Sea D un dígrafo acíclico de $n + 1$ nodos. Queremos ver que tiene un orden topológico.

Demostración

- Formalizando, sea $P(k)$ la proposición “Todo dígrafo acíclico de k nodos tiene un orden topológico”. Vale $P(1)$.
- Probemos $P(n) \implies P(n+1)$: Sea D un dígrafo acíclico de $n+1$ nodos. Queremos ver que tiene un orden topológico.
- Como no tiene ciclos, tiene que tener un nodo con grado de entrada 0, al cual llamaremos v . Sea v este nodo.

Demostración

- Formalizando, sea $P(k)$ la proposición “Todo dígrafo acíclico de k nodos tiene un orden topológico”. Vale $P(1)$.
- Probemos $P(n) \implies P(n+1)$: Sea D un dígrafo acíclico de $n+1$ nodos. Queremos ver que tiene un orden topológico.
- Como no tiene ciclos, tiene que tener un nodo con grado de entrada 0, al cual llamaremos v . Sea v este nodo.
- Notemos que $G \setminus \{v\}$ es un dígrafo acíclico, y como tiene n nodos y vale $P(n)$ concluimos que tiene un orden topológico $w_1 \dots w_n$.

Demostración

- Formalizando, sea $P(k)$ la proposición “Todo dígrafo acíclico de k nodos tiene un orden topológico”. Vale $P(1)$.
- Probemos $P(n) \implies P(n+1)$: Sea D un dígrafo acíclico de $n+1$ nodos. Queremos ver que tiene un orden topológico.
- Como no tiene ciclos, tiene que tener un nodo con grado de entrada 0, al cual llamaremos v . Sea v este nodo.
- Notemos que $G \setminus \{v\}$ es un dígrafo acíclico, y como tiene n nodos y vale $P(n)$ concluimos que tiene un orden topológico $w_1 \dots w_n$.
- Luego, $vw_1 \dots w_n$ es un orden topológico para G .

Orden topológico con DFS

Enunciado

Dar un algoritmo basado en DFS que encuentre un orden topológico de un dígrafo D en tiempo lineal, si lo tiene.

Orden topológico con DFS

Enunciado

Dar un algoritmo basado en DFS que encuentre un orden topológico de un dígrafo D en tiempo lineal, si lo tiene.

Ayuda: usar un stack para apilar los nodos en cierto orden

Ideas

- Primero podemos verificar si el dígrafo tiene ciclos (¿Cómo?), y si los tiene devolver que no hay un orden topológico.

Ideas

- Primero podemos verificar si el dígrafo tiene ciclos (¿Cómo?), y si los tiene devolver que no hay un orden topológico.
- Ahora pensemos la siguiente propiedad de DFS: si v es un nodo que se está empezando a procesar, entonces se va a terminar de procesar luego de que todos los nodos blancos alcanzables por él sean procesados.

Ideas

- Primero podemos verificar si el dígrafo tiene ciclos (¿Cómo?), y si los tiene devolver que no hay un orden topológico.
- Ahora pensemos la siguiente propiedad de DFS: si v es un nodo que se está empezando a procesar, entonces se va a terminar de procesar luego de que todos los nodos blancos alcanzables por él sean procesados.
- Por lo tanto, si terminamos de procesar v , ¿Qué sabemos de los nodos hacia los que v tiene ejes?

Ideas

- Primero podemos verificar si el dígrafo tiene ciclos (¿Cómo?), y si los tiene devolver que no hay un orden topológico.
- Ahora pensemos la siguiente propiedad de DFS: si v es un nodo que se está empezando a procesar, entonces se va a terminar de procesar luego de que todos los nodos blancos alcanzables por él sean procesados.
- Por lo tanto, si terminamos de procesar v , ¿Qué sabemos de los nodos hacia los que v tiene ejes?
- Ya terminaron de procesarse.

Ideas

- Primero podemos verificar si el dígrafo tiene ciclos (¿Cómo?), y si los tiene devolver que no hay un orden topológico.
- Ahora pensemos la siguiente propiedad de DFS: si v es un nodo que se está empezando a procesar, entonces se va a terminar de procesar luego de que todos los nodos blancos alcanzables por él sean procesados.
- Por lo tanto, si terminamos de procesar v , ¿Qué sabemos de los nodos hacia los que v tiene ejes?
- Ya terminaron de procesarse. ¿Cómo podríamos “anotar” ese orden?

Ideas

- Primero podemos verificar si el dígrafo tiene ciclos (¿Cómo?), y si los tiene devolver que no hay un orden topológico.
- Ahora pensemos la siguiente propiedad de DFS: si v es un nodo que se está empezando a procesar, entonces se va a terminar de procesar luego de que todos los nodos blancos alcanzables por él sean procesados.
- Por lo tanto, si terminamos de procesar v , ¿Qué sabemos de los nodos hacia los que v tiene ejes?
- Ya terminaron de procesarse. ¿Cómo podríamos “anotar” ese orden?
- En la parte de post-procesamiento pusheamos los nodos a un stack. Luego, el orden topológico sería el del stack “desde arriba hasta abajo”.

Afilando ideas

- ¿Desde dónde ejecutamos el DFS?

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).
- En el caso de digrafos, ¿Cómo sabemos que encontramos un ciclo?

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).
- En el caso de digrafos, ¿Cómo sabemos que encontramos un ciclo? Supongamos que estamos procesando un eje vw (y estamos parados en v), hay 3 opciones.

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).
- En el caso de digrafos, ¿Cómo sabemos que encontramos un ciclo? Supongamos que estamos procesando un eje vw (y estamos parados en v), hay 3 opciones.
- w está pintado de blanco, y por lo tanto este eje nunca fue procesado. Entonces el eje vw no puede formar un ciclo dentro de la parte del digrafo que ya vimos.

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).
- En el caso de digrafos, ¿Cómo sabemos que encontramos un ciclo? Supongamos que estamos procesando un eje vw (y estamos parados en v), hay 3 opciones.
- w está pintado de blanco, y por lo tanto este eje nunca fue procesado. Entonces el eje vw no puede formar un ciclo dentro de la parte del digrafo que ya vimos.
- w está pintado de negro, en cuyo caso ya terminó de procesarse. Si hubiera un camino de w a v entonces v se habría terminado de procesar antes que w (propiedad de DFS). Por lo tanto, el eje vw no forma un ciclo.

Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).
- En el caso de digrafos, ¿Cómo sabemos que encontramos un ciclo? Supongamos que estamos procesando un eje vw (y estamos parados en v), hay 3 opciones.
- w está pintado de blanco, y por lo tanto este eje nunca fue procesado. Entonces el eje vw no puede formar un ciclo dentro de la parte del digrafo que ya vimos.
- w está pintado de negro, en cuyo caso ya terminó de procesarse. Si hubiera un camino de w a v entonces v se habría terminado de procesar antes que w (propiedad de DFS). Por lo tanto, el eje vw no forma un ciclo.
- w está pintado de gris. En ese caso, hay un camino de nodos pintados de gris de w a v , y luego el eje vw cierra el ciclo (recordemos que los nodos pintados de gris en todo momento forman un camino dentro del digrafo o grafo).

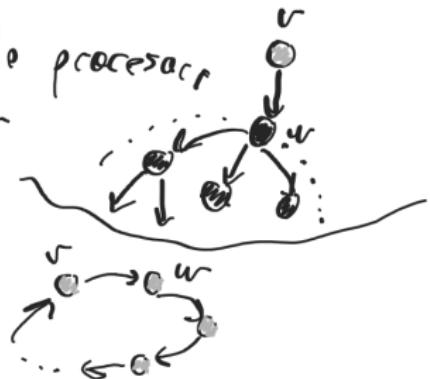
Afilando ideas

- ¿Desde dónde ejecutamos el DFS? Notemos que, a diferencia del caso con grafos, no es cierto que un DFS alcance para procesar todos los nodos.
- Resolvemos este problema agregando un nodo universal con ejes hacia todos los nodos, o bien corriendo el DFS secuencialmente en todos los nodos (ambas ideas son análogas, y tienen complejidad lineal).
- En el caso de digrafos, ¿Cómo sabemos que encontramos un ciclo? Supongamos que estamos procesando un eje vw (y estamos parados en v), hay 3 opciones.
- w está pintado de blanco, y por lo tanto este eje nunca fue procesado. Entonces el eje vw no puede formar un ciclo dentro de la parte del digrafo que ya vimos.
- w está pintado de negro, en cuyo caso ya terminó de procesarse. Si hubiera un camino de w a v entonces v se habría terminado de procesar antes que w (propiedad de DFS). Por lo tanto, el eje vw no forma un ciclo.
- w está pintado de gris. En ese caso, hay un camino de nodos pintados de gris de w a v , y luego el eje vw cierra el ciclo (recordemos que los nodos pintados de gris en todo momento forman un camino dentro del digrafo o grafo).
- Por lo tanto, podemos detectar ciclos si modificamos DFS para que se fije, al descubrir un eje, si apunta hacia un eje pintado de gris.

Dibujo: ciclos en digrafos

Tres opciones al procesar $v \rightarrow w$: $\{ v \text{ es gris porque se está procesando} \}$

- 1) $v \rightarrow w$ } w blanco } No se forma un ciclo, veo a w por primera vez
- 2) $v \rightarrow w$ } w negro } w ya se terminó de procesar, y no hay camino de w a v
- 3) $v \rightarrow w$ } w gris } Hay ciclo siguiendo los grises



Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris?

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris? Hay un ciclo en el dígrafo, lo cual es absurdo.

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris? Hay un ciclo en el dígrafo, lo cual es absurdo.
- ¿Si está negro?

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
 - ¿Qué pasa si está pintado de gris? Hay un ciclo en el dígrafo, lo cual es absurdo.
 - ¿Si está negro? Ya fue procesado, y por lo tanto está en el stack.

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris? Hay un ciclo en el dígrafo, lo cual es absurdo.
- ¿Si está negro? Ya fue procesado, y por lo tanto está en el stack.
- ¿Si está blanco? Entonces w se va a terminar de procesar antes que v .

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris? Hay un ciclo en el dígrafo, lo cual es absurdo.
- ¿Si está negro? Ya fue procesado, y por lo tanto está en el stack.
- ¿Si está blanco? Entonces w se va a terminar de procesar antes que v .
- En todos los casos v se termina de procesar luego que w , y por lo tanto los ejes están siempre “de arriba para abajo” en el stack.

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el digrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris? Hay un ciclo en el digrafo, lo cual es absurdo.
- ¿Si está negro? Ya fue procesado, y por lo tanto está en el stack.
- ¿Si está blanco? Entonces w se va a terminar de procesar antes que v .
- En todos los casos v se termina de procesar luego que w , y por lo tanto los ejes están siempre “de arriba para abajo” en el stack.
- ¿Complejidad del algoritmo?

Demostración

- Volviendo al algoritmo del orden topológico, queremos probar que el stack apila los nodos en un orden que nos gusta. Recuerden que para este momento estamos asumiendo que el dígrafo no tiene ciclos (ya lo revisamos).
- Para probar esto alcanza con probar que cuando se procesa un eje vw se termina de procesar w antes que v .
- Supongamos que el DFS empezó a procesar el eje vw (y por lo tanto, está procesando al nodo v). Pensemos de qué colores puede estar pintado w .
- ¿Qué pasa si está pintado de gris? Hay un ciclo en el dígrafo, lo cual es absurdo.
- ¿Si está negro? Ya fue procesado, y por lo tanto está en el stack.
- ¿Si está blanco? Entonces w se va a terminar de procesar antes que v .
- En todos los casos v se termina de procesar luego que w , y por lo tanto los ejes están siempre “de arriba para abajo” en el stack.
- ¿Complejidad del algoritmo? $O(n + m)$.
- Ejercicio: se podía hacer el chequeo de ciclos al mismo tiempo que el DFS con stack. Convencerse de que eso funciona.

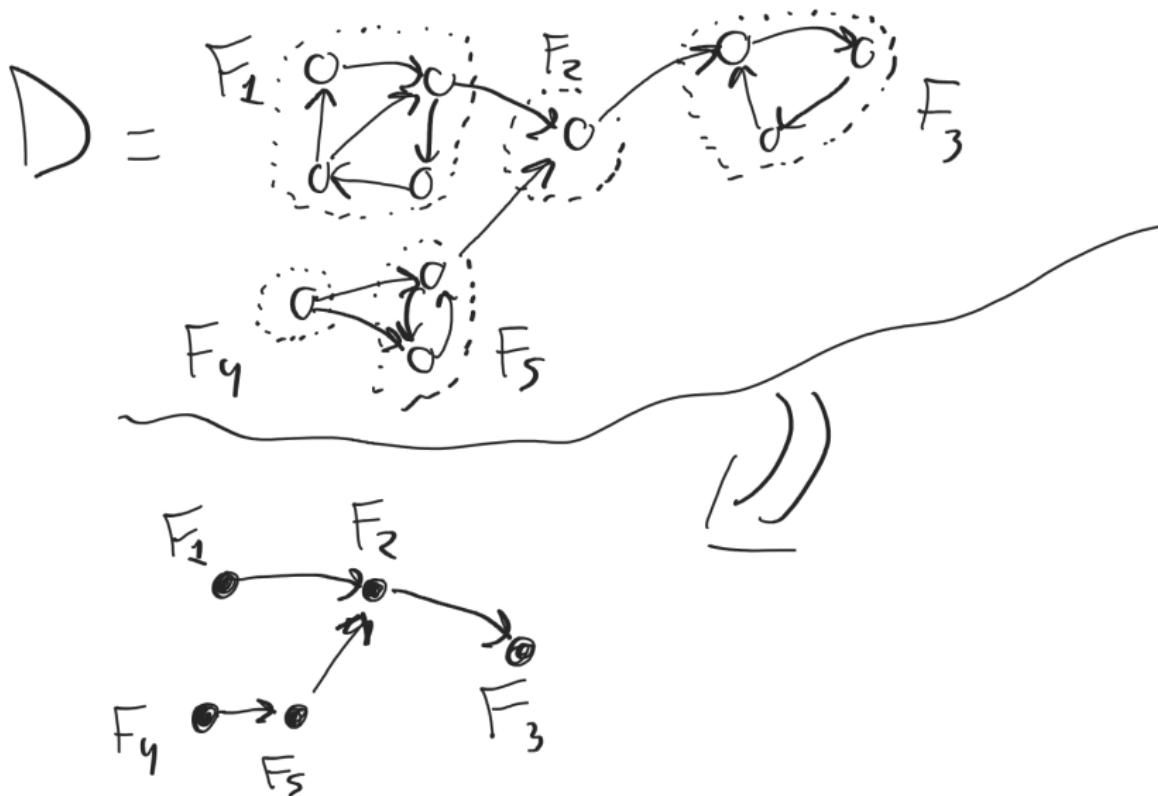
¿Y si no es un DAG?

- Cuando un dígrafo tiene ciclos, aún así podemos pensar en un cierto DAG que surge de “colapsar” los ciclos en nodos.

¿Y si no es un DAG?

- Cuando un dígrafo tiene ciclos, aún así podemos pensar en un cierto DAG que surge de “colapsar” los ciclos en nodos.
- La noción de componente fuertemente conexa formaliza esta idea.

Dibujo: colapsando componentes



Definición

Enunciado

Dado un digrafo D , una componente fuertemente conexa de D es un subconjunto maximal F de sus nodos tal que para todo par de nodos $v, w \in F$ existe un camino de v a w . Maximal quiere decir que no se puede agregar otro nodo y que la componente siga siendo fuertemente conexa.

Probar que todo nodo pertenece a una única componente fuertemente conexa, y que por lo tanto las componentes fuertemente conexas definen una partición de los nodos.

Definición

Enunciado

Dado un digrafo D , una componente fuertemente conexa de D es un subconjunto maximal F de sus nodos tal que para todo par de nodos $v, w \in F$ existe un camino de v a w . Maximal quiere decir que no se puede agregar otro nodo y que la componente siga siendo fuertemente conexa.

Probar que todo nodo pertenece a una única componente fuertemente conexa, y que por lo tanto las componentes fuertemente conexas definen una partición de los nodos.

Ayuda: Por absurdo.

Definición

- Supongamos que $v \in F_1$ y $v \in F_2$ y $F_1 \neq F_2$.

Definición

- Supongamos que $v \in F_1$ y $v \in F_2$ y $F_1 \neq F_2$.
- Como F_1 es una componente fuertemente conexa, todos los nodos $w \in F_1$ tienen un camino a v . Por otro lado, todos los nodos de F_2 son alcanzables por v (por ser F_2 una componente fuertemente conexa), por lo que todos los nodos de F_1 alcanzan a todos los nodos de F_2 (a través de un camino por v).

Definición

- Supongamos que $v \in F_1$ y $v \in F_2$ y $F_1 \neq F_2$.
- Como F_1 es una componente fuertemente conexa, todos los nodos $w \in F_1$ tienen un camino a v . Por otro lado, todos los nodos de F_2 son alcanzables por v (por ser F_2 una componente fuertemente conexa), por lo que todos los nodos de F_1 alcanzan a todos los nodos de F_2 (a través de un camino por v).
- El mismo argumento se puede hacer pero en el sentido contrario.

Definición

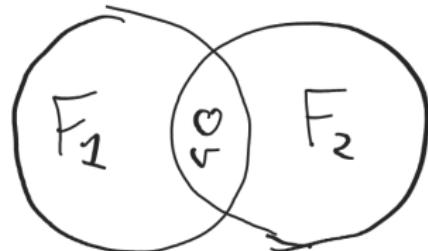
- Supongamos que $v \in F_1$ y $v \in F_2$ y $F_1 \neq F_2$.
- Como F_1 es una componente fuertemente conexa, todos los nodos $w \in F_1$ tienen un camino a v . Por otro lado, todos los nodos de F_2 son alcanzables por v (por ser F_2 una componente fuertemente conexa), por lo que todos los nodos de F_1 alcanzan a todos los nodos de F_2 (a través de un camino por v).
- El mismo argumento se puede hacer pero en el sentido contrario.
- Por lo tanto, $F_1 = F_2$ o bien uno de los F_i no es maximal.
- Esto prueba que cada nodo pertenece a **lo sumo** a una componente fuertemente conexa ¿Cómo probamos que pertenece por lo menos a una?

Definición

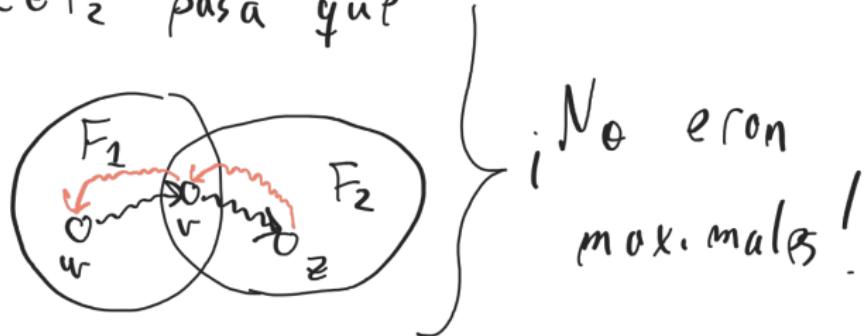
- Supongamos que $v \in F_1$ y $v \in F_2$ y $F_1 \neq F_2$.
- Como F_1 es una componente fuertemente conexa, todos los nodos $w \in F_1$ tienen un camino a v . Por otro lado, todos los nodos de F_2 son alcanzables por v (por ser F_2 una componente fuertemente conexa), por lo que todos los nodos de F_1 alcanzan a todos los nodos de F_2 (a través de un camino por v).
- El mismo argumento se puede hacer pero en el sentido contrario.
- Por lo tanto, $F_1 = F_2$ o bien uno de los F_i no es maximal. *Absurdo!*
- Esto prueba que cada nodo pertenece a **lo sumo** a una componente fuertemente conexa. ¿Cómo probamos que pertenece por lo menos a una? Todo conjunto de la forma $\{v\}$ es un subconjunto de nodos conectados entre si, y por lo tanto hay una componente maximal que extiende a $\{v\}$.

Dibujo: las cfcs no comparten nodos

Supongamos



Luego $\forall w \in F_1, z \in F_2$ pasa que



El digrafo de cfcs

Enunciado

Dado un digrafo D , sean $F_1 \dots F_k$ las componentes fuertemente conexas de G . Definimos el *digrafo de componentes fuertemente conexas* de D como el digrafo que tiene un nodo por componente fuertemente conexa de D , y dos de estos nodos F_i, F_j tiene un eje dirigido $F_i; F_j$ si y solamente si hay nodos $v_i \in F_i$ y $v_j \in F_j$ tales que $v_i; v_j$ es un eje del digrafo.

Probar que este digrafo de componentes fuertemente conexas no tiene ciclos.

El digrafo de cfcs

Enunciado

Dado un digrafo D , sean $F_1 \dots F_k$ las componentes fuertemente conexas de G . Definimos el *digrafo de componentes fuertemente conexas* de D como el digrafo que tiene un nodo por componente fuertemente conexa de D , y dos de estos nodos F_i, F_j tiene un eje dirigido $F_i; F_j$ si y solamente si hay nodos $v_i \in F_i$ y $v_j \in F_j$ tales que $v_i; v_j$ es un eje del digrafo.

Probar que este digrafo de componentes fuertemente conexas no tiene ciclos.

Ayuda: por absurdo.

El digrafo de cfcs

- Si hay un ciclo $F_1 \dots F_l F_1$ luego hay un ciclo en el digrafo original $s_1 \dots t_1 s_2 \dots t_2 s_3 \dots t_{l-1} s_l \dots t_l s_1$ con $s_i, t_i \in F_i$ y todos los nodos intermedios entre s_i y t_i pertenecen a F_i .

El digrafo de cfcs

- Si hay un ciclo $F_1 \dots F_l F_1$ luego hay un ciclo en el digrafo original $s_1 \dots t_1 s_2 \dots t_2 s_3 \dots t_{l-1} s_l \dots t_l s_1$ con $s_i, t_i \in F_i$ y todos los nodos intermedios entre s_i y t_i pertenecen a F_i .
- Notemos entonces que dados dos nodos $w_i \in F_i, w_j \in F_j$ siempre hay un camino de w_i a w_j siguiendo el ciclo que describimos y moviéndonos dentro de las componentes fuertemente conexas.

El digrafo de cfcs

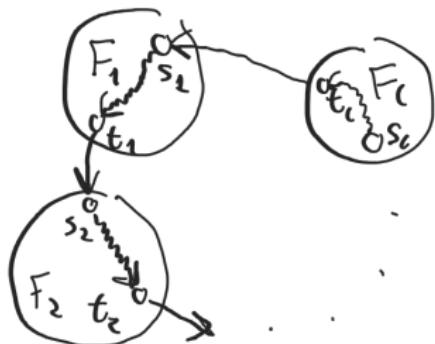
- Si hay un ciclo $F_1 \dots F_l F_1$ luego hay un ciclo en el digrafo original $s_1 \dots t_1 s_2 \dots t_2 s_3 \dots t_{l-1} s_l \dots t_l s_1$ con $s_i, t_i \in F_i$ y todos los nodos intermedios entre s_i y t_i pertenecen a F_i .
- Notemos entonces que dados dos nodos $w_i \in F_i, w_j \in F_j$ siempre hay un camino de w_i a w_j siguiendo el ciclo que describimos y moviéndonos dentro de las componentes fuertemente conexas.
- Por lo tanto, $\bigcup_{i=1}^l F_i$ es una componente fuertemente conexa más grande, lo cual es absurdo.

Dibujo: el digrafo de cfcs no tiene ciclos

Supongamos



Haciendo zoom...



} Cada eje t_i, s_i , tiene
 que existir porque $F_i F_{i+1}$ es
 un eje del digrafo de componentes
 El comando $s_i \rightarrow t_i$ existe por
 ser F_i cfcs.

Hacia el algoritmo

- Acabamos de probar que el digrafo de componentes fuertemente conexas tiene la pinta que queremos.

Hacia el algoritmo

- Acabamos de probar que el digrafo de componentes fuertemente conexas tiene la pinta que queremos.
- Ahora diseñemos un algoritmo para encontrar las particiones en cfcs.

Hacia el algoritmo

- Acabamos de probar que el digrafo de componentes fuertemente conexas tiene la pinta que queremos.
- Ahora diseñemos un algoritmo para encontrar las particiones en cfcs.
- Pensemos en el algoritmo anterior... ¿Podemos decir algo respecto al orden en que se pushean los nodos al stack, teniendo en cuenta las cfcs en las que están contenidos?

Lema

Enunciado

Supongamos que hacemos DFS sobre el digrafo D , pusheando a un stack los nodos cuando los pintamos de negro.

Probar que si una cfc F_1 tiene un eje hacia otra cfc F_2 en el digrafo de cfcs entonces algún nodo de F_1 está pusheado arriba de todos los nodos de F_2 .

Lema

Enunciado

Supongamos que hacemos DFS sobre el digrafo D , pusheando a un stack los nodos cuando los pintamos de negro.

Probar que si una cfc F_1 tiene un eje hacia otra cfc F_2 en el digrafo de cfcs entonces algún nodo de F_1 esta pusheado arriba de todos los nodos de F_2 .

Ayuda: ir por casos, pensando primero que pasa si se descubre un nodo de F_2 antes que cualquiera de F_1 .

Lema

- ¿Qué pasa si se descubre primero un nodo de F_2 ?

Lema

- ¿Qué pasa si se descubre primero un nodo de F_2 ?

Lema

- ¿Qué pasa si se descubre primero un nodo de F_2 ? Toda la componente F_2 va a ser pusheada antes que F_1 .

Lema

- ¿Qué pasa si se descubre primero un nodo de F_2 ? Toda la componente F_2 va a ser pusheada antes que F_1 .
- ¿Y si se descubre primero un nodo $v \in F_1$?

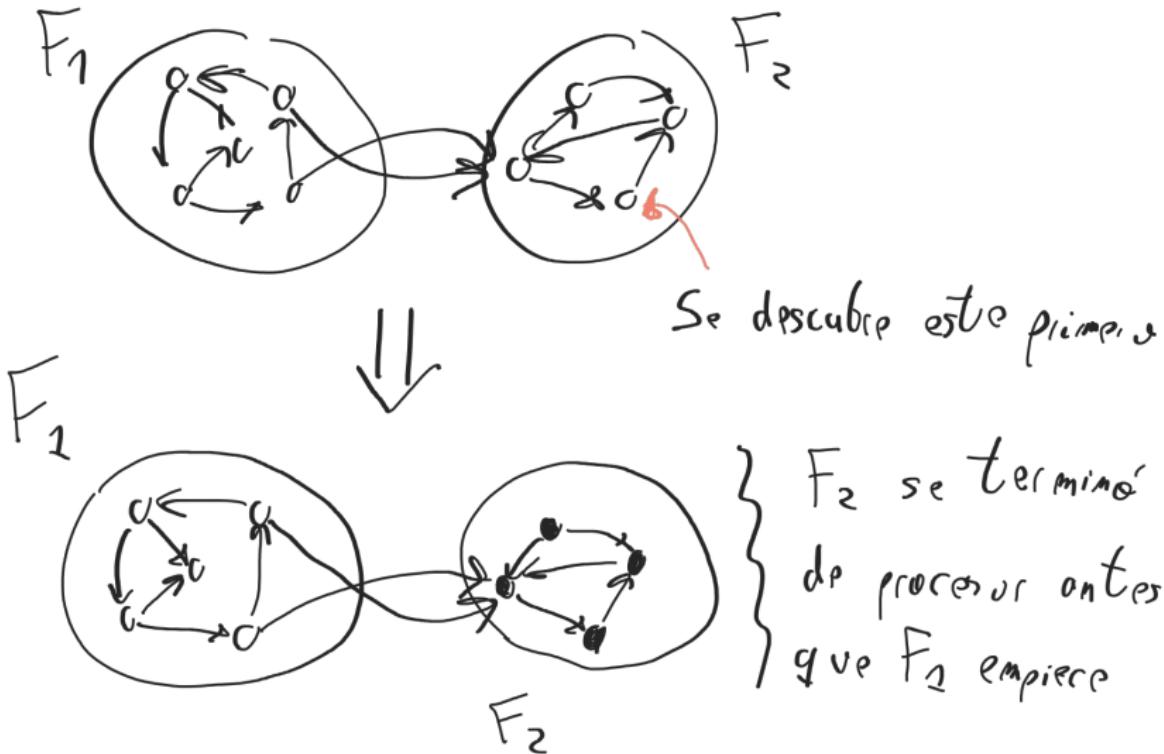
Lema

- ¿Qué pasa si se descubre primero un nodo de F_2 ? Toda la componente F_2 va a ser pusheada antes que F_1 .
- ¿Y si se descubre primero un nodo $v \in F_1$?

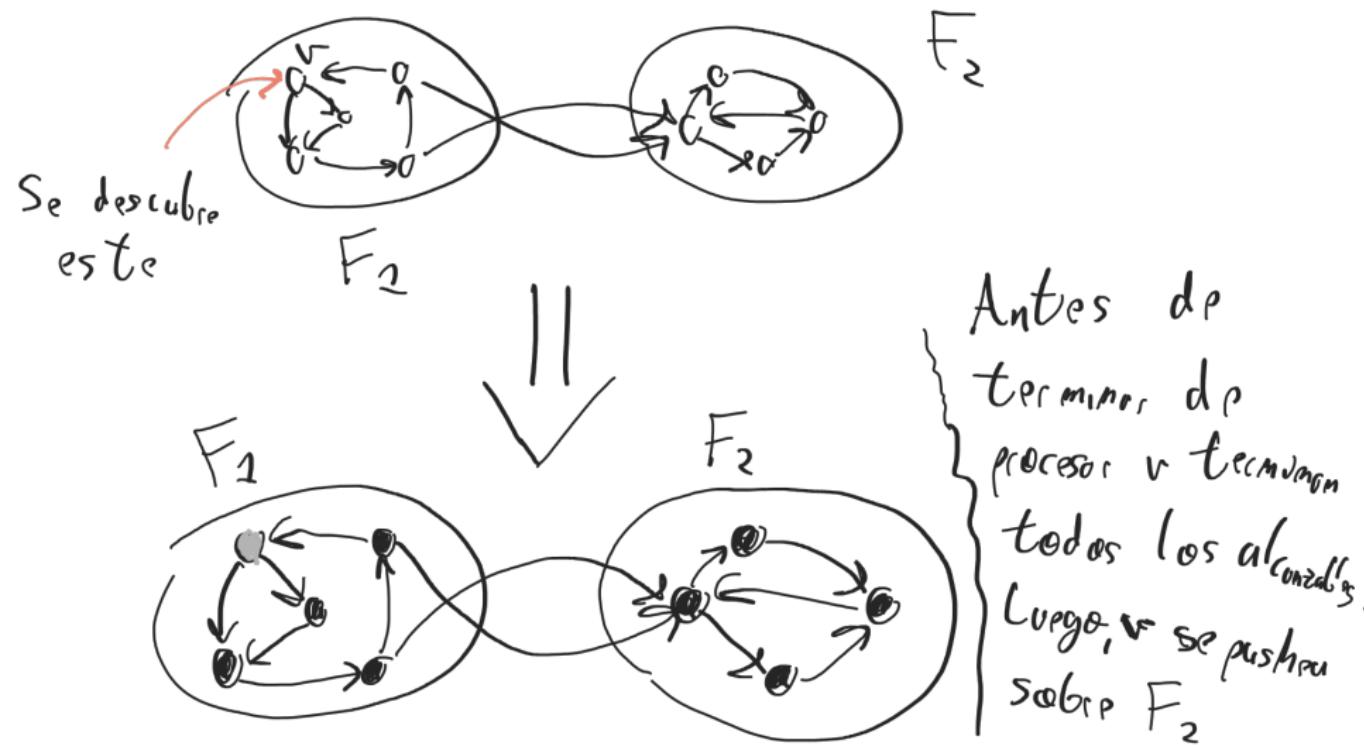
Lema

- ¿Qué pasa si se descubre primero un nodo de F_2 ? Toda la componente F_2 va a ser pusheada antes que F_1 .
- ¿Y si se descubre primero un nodo $v \in F_1$? Entonces F_2 se va a terminar de procesar antes que v , y entonces el nodo v va a quedar arriba de F_2 .

Dibujo: si se descubre primero F_2



Dibujo: si se descubre primero F_1



Últimos pasos

- Supongamos que tenemos el stack S ya armado ¿Qué sabemos del nodo v que está arriba del todo?

Últimos pasos

- Supongamos que tenemos el stack S ya armado ¿Qué sabemos del nodo v que está arriba del todo? Pertenece a una de las componentes de grado de entrada 0 en el digrafo de cfcs.

Últimos pasos

- Supongamos que tenemos el stack S ya armado ¿Qué sabemos del nodo v que está arriba del todo? Pertenece a una de las componentes de grado de entrada 0 en el digrafo de cfcs.
- ¿Cómo la podríamos recuperar usando v ?

Últimos pasos

- Supongamos que tenemos el stack S ya armado ¿Qué sabemos del nodo v que está arriba del todo? Pertenece a una de las componentes de grado de entrada 0 en el digrafo de cfcs.
- ¿Cómo la podríamos recuperar usando v ? Si giramos los ejes y hacemos DFS desde v las deberíamos alcanzar ¿Por qué?

Últimos pasos

- Supongamos que tenemos el stack S ya armado ¿Qué sabemos del nodo v que está arriba del todo? Pertenece a una de las componentes de grado de entrada 0 en el digrafo de cfcs.
- ¿Cómo la podríamos recuperar usando v ? Si giramos los ejes y hacemos DFS desde v las deberíamos alcanzar ¿Por qué?
- Fundamentalmente, porque las cfcs del digrafo traspuesto son las mismas que del digrafo original, pero las componentes con grado de entrada 0 ahora tienen grado de salida 0.

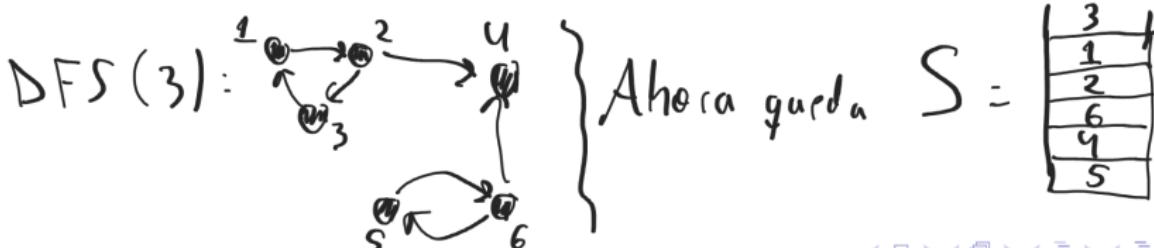
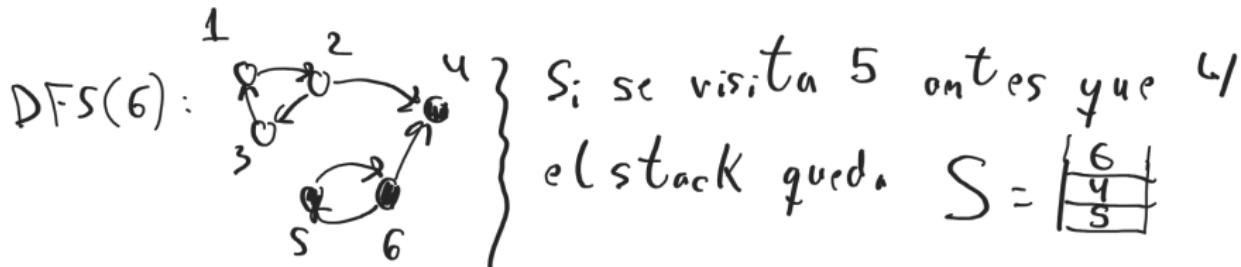
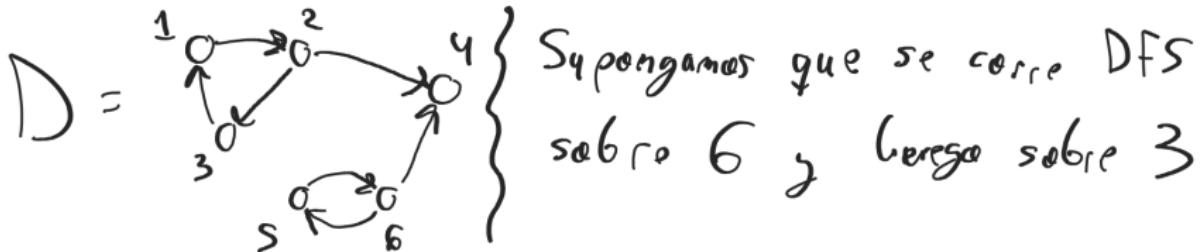
El algoritmo

Enunciado

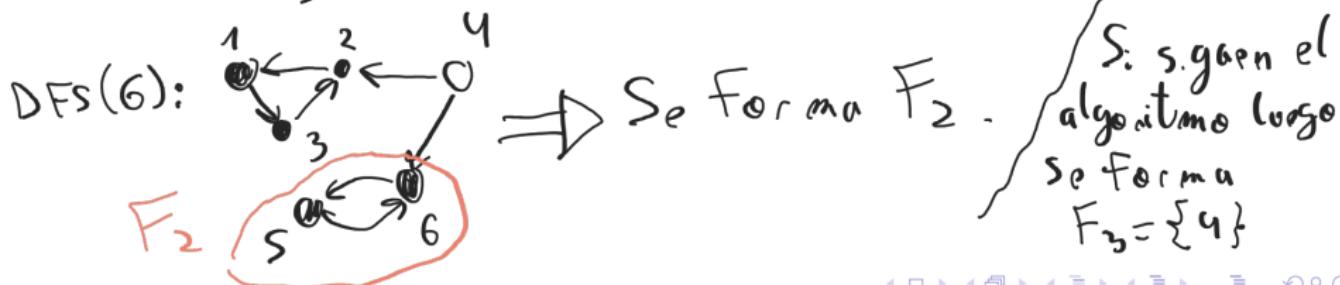
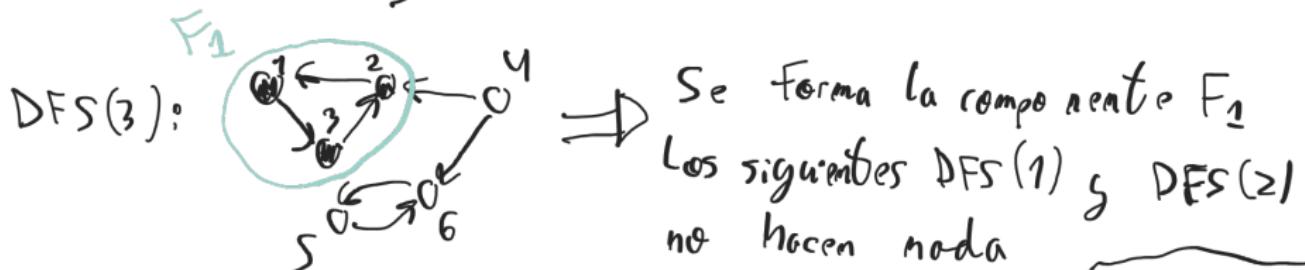
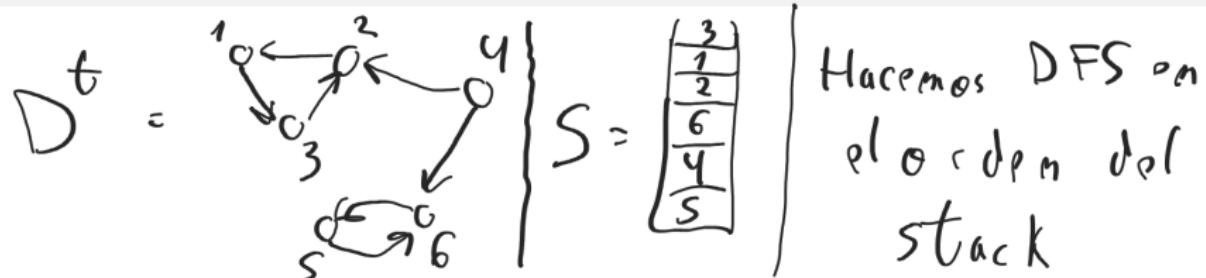
Ejecutemos DFS sobre D^T usando como orden para los nodos el orden en el cual se polean los nodos de la pila S .

Probar que cuando se procesa un nodo no visitado u los nodos descubiertos por primera vez por ese DFS forman exactamente la cfc que incluye a u .

Dibujo: ejemplo del algoritmo



Dibujo: ejemplo del algoritmo



El algoritmo

- Llamemos F a la componente fuertemente conexa de u .

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ?

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más?

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más? Supongamos un nodo v fuera de F que u alcanza. v pertenece a otra componente F' .

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más? Supongamos un nodo v fuera de F que u alcanza. v pertenece a otra componente F' .
- ¿Qué podemos decir de F' ?
- El eje $F'F$ está en el grafo de componentes fuertemente conexas (¿Por qué?)

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más? Supongamos un nodo v fuera de F que u alcanza. v pertenece a otra componente F' .
- ¿Qué podemos decir de F' ?
- El eje $F'F$ está en el grafo de componentes fuertemente conexas (¿Por qué?), y por lo tanto hay un nodo de F' arriba de u en el stack, que fue procesado antes.

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más? Supongamos un nodo v fuera de F que u alcanza. v pertenece a otra componente F' .
- ¿Qué podemos decir de F' ?
- El eje $F'F$ está en el grafo de componentes fuertemente conexas (¿Por qué?), y por lo tanto hay un nodo de F' arriba de u en el stack, que fue procesado antes.
- Entonces, toda F' ya fue descubierta (¿Por qué?)

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más? Supongamos un nodo v fuera de F que u alcanza. v pertenece a otra componente F' .
- ¿Qué podemos decir de F' ?
- El eje $F'F$ está en el grafo de componentes fuertemente conexas (¿Por qué?), y por lo tanto hay un nodo de F' arriba de u en el stack, que fue procesado antes.
- Entonces, toda F' ya fue descubierta (¿Por qué?) y luego el DFS desde u no la va a descubrir.

El algoritmo

- Llamemos F a la componente fuertemente conexa de u .
- ¿Se va a descubrir la componente F ? Si, porque u alcanza a toda la componente F , y todos esos nodos están en blanco.
- ¿Va a alcanzar algo más? Supongamos un nodo v fuera de F que u alcanza. v pertenece a otra componente F' .
- ¿Qué podemos decir de F' ?
- El eje $F'F$ está en el grafo de componentes fuertemente conexas (¿Por qué?), y por lo tanto hay un nodo de F' arriba de u en el stack, que fue procesado antes.
- Entonces, toda F' ya fue descubierta (¿Por qué?) y luego el DFS desde u no la va a descubrir.
- Como esto valía para cualquier v , concluimos que el DFS desde u solo descubre a F , su componente fuertemente conexa.

Componentes fuertemente conexas

Resumen: Algoritmo de Kosaraju

- ① Hacer DFS en D pusheando en un stack S los nodos cuando se terminan de procesar.
- ② Hacer DFS sobre D^T tomando los nodos en el orden en que se poppean de S . Cuando se procesa un nodo nuevo u , todos los nodos que se descubren forman la cfc de u .

Componentes fuertemente conexas

Resumen: Algoritmo de Kosaraju

- ① Hacer DFS en D pusheando en un stack S los nodos cuando se terminan de procesar.
- ② Hacer DFS sobre D^T tomando los nodos en el orden en que se poppean de S . Cuando se procesa un nodo nuevo u , todos los nodos que se descubren forman la cfc de u .

¿Cuál es la complejidad?

Componentes fuertemente conexas

Resumen: Algoritmo de Kosaraju

- ① Hacer DFS en D pusheando en un stack S los nodos cuando se terminan de procesar.
- ② Hacer DFS sobre D^T tomando los nodos en el orden en que se poppean de S . Cuando se procesa un nodo nuevo u , todos los nodos que se descubren forman la cfc de u .

¿Cuál es la complejidad? $O(n + m)$

Referencias

- ① Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.
- ② <https://cp-algorithms.com/graph/strongly-connected-components.html>