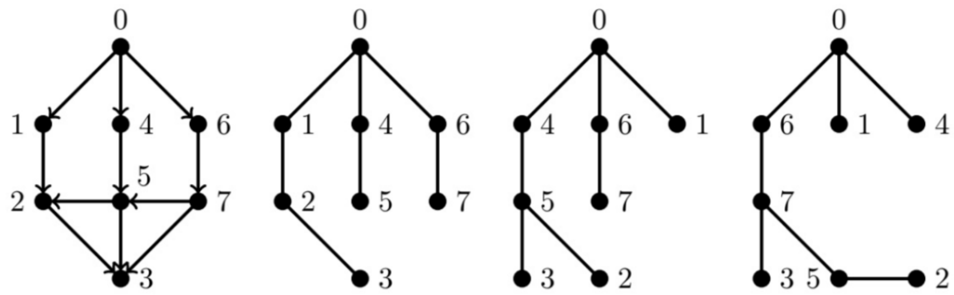
17. Dado un digrafo D, un ordenamiento $v_1, \ldots, v_n$ de $V(D)$ es un *orden topológico* de D cuando para toda arista $v_i \to v_j$ de D ocurre que $i < j$ (Figura 3). En la guía pasada vimos que D admite un orden topológico si y solo si D es acíclico. En este ejercicio vemos cómo determinar si D es acíclico y obtener un orden topológico usando DFS.

Sea v un vértice que alcanza todos los otros vértices de un digrafo D y sea T un árbol generador que se obtiene al ejecutar DFS desde v. Más aun, supongamos que los vértices hermanos de T



postorder: 3,2,1,5,4,7,6,0 3,2,5,4,7,6,1,0 3,2,5,7,6,1,4,0

Figura 3. *Árboles DFS de un digrafo acíclico D y sus correspondientes post-órdenes, cada una de las cuales es el reverso de un orden topológico de D*

están ordenados de forma tal que u aparece antes que su hermano w cuando u fue descubierto antes que w por el algoritmo DFS (por lo tanto el vecindario de u fue procesado antes que el de w). Finalmente, sea S la secuencia que se obtiene al revisar T en sentido postorder (Figura 3; recordar que para todo árbol con raíz r y **secuencia** de subárboles $T_1, \ldots, T_k$ se tiene postorder(r) = postorder(T_1) + ... + postorder(T_k) + r).

a) Demostrar que D es acíclico si y solo si el reverso de S es un orden topológico de D.

b) Describir el algoritmo resultante para determinar si D es acíclico y obtener el orden topológico correspondiente.

c) Modificar el algortimo anterior para evitar la suposición de que existe un vértice que alcanza a todos los otros vértices.

ⓐ QVQ D es acíclico $\longleftrightarrow$ el reverso de S es orden topológico de D

$\leftarrow$) Como S es un orden topológico entonces
D admite un orden topológico $\rightarrow$ es acíclico

$\rightarrow$) D es acíclico $\rightarrow$ admite un orden topológico. Veamos que
el reverso de S es uno

$S = \{v_1, \ldots, v_n\}$ es la secuencia que se obtiene de recorrer T en un sentido post-order y T es el árbol generador de un digrafo D que se obtuvo a partir de DFS donde además si tenemos u y w vértices hermanos entonces si u está antes que w, u fue descubierto antes.

$\rightarrow$ Como S y T están definidos de esta manera

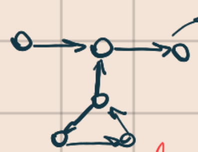
Si tomamos $v_i v_j \in S$ (con $i<j$) entonces se cumple que: o v_j es padre de v_i o v_j es hermano de v_i → o tenemos $v_j \to v_i \in T_v$ ∴ a D o directamente NO tenemos arista entre v_i y v_j en el árbol.

Entonces $\forall v_i, v_j \in S / i<j \longrightarrow \exists\, v_j \to v_i \in D$ o

∴ Si considero el reverso de S (S^*) vale que $\forall v_i^* v_j^* \in S^* / i<j$
$\exists\, v_i \to v_j \in D \to S^*$ es orden topológico

(b) Hay 2 partes de mi algoritmo: Chequeo de ciclos y orden topológico

↪ puedo hacer ambas con un mismo DFS. Es más, hasta el problema más general sale con ese DFS



Des conexo pero NO fuertemente podría ser problemático

Algoritmo

vamos a suponer que para c/ componente fuertemente conexa tomamos un vértice que alcanza a todos los de la componente

1. Para c/vértice en mi digrafo

Hago un DFS de 3 estados: No descubierto, descubierto, explorado y que tome un vértice y su padre (cuando el algo empieza es -1).

Va a funcionar así:

Dado un vértice y su padre si está marcado como NO descubierto, lo marco como descubierto y exploro como el DFS.

Si el vértice fue descubierto y su padre es un vértice distinto quiere decir que encontramos un ciclo: devuelvo que NO es acíclico

Si el vértice fue descubierto y su padre es él mismo o no tiene hijos o no tiene más hijos para explorar lo marcamos como explorado y lo agrego a un stack S. Si NO devolví que tiene ciclos devuelvo el stack invertido → orden topológico

→ Podría dividir cada respuesta x separado según componente