

Ejercitación Teórica N° 4: Grafos.

Ejercicio 1.

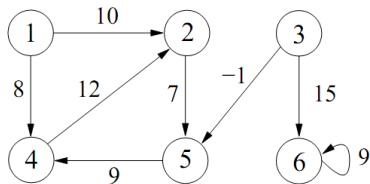


Figura 4

(a) Aplicando el recorrido DFS al grafo dirigido de la Figura 4, ¿cuáles son los vértices alcanzables desde el vértice 1 y en qué orden?

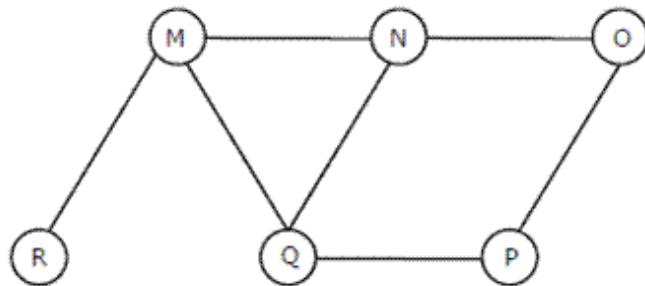
Aplicando el recorrido DFS al grafo dirigido de la Figura 4, los vértices alcanzables desde el vértice 1 son, en orden, 1, 2, 5 y 4.

(b) Aplicando el recorrido BFS al grafo dirigido de la Figura 4, ¿cuáles son los vértices alcanzables desde el vértice 1 y en qué orden?

Aplicando el recorrido BFS al grafo dirigido de la Figura 4, los vértices alcanzables desde el vértice 1 son, en orden, 1, 2, 4 y 5.

Ejercicio 2.

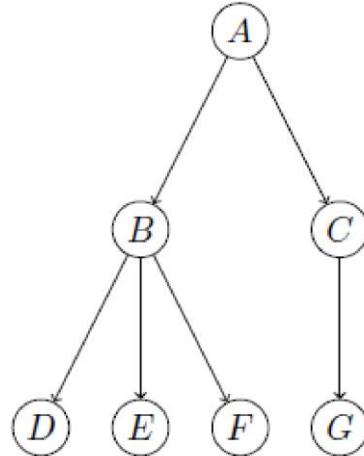
¿Cuál de los siguientes es un recorrido BFS válido para el grafo de la figura?



- (a) $MNOPQR$. (b) $NQMPOR$. (c) $QMNPOR$. (d) $QMNPOR$.

Ejercicio 3.

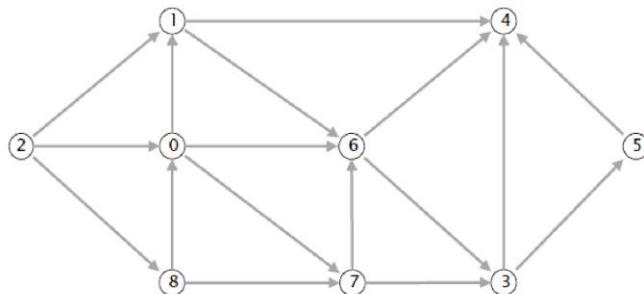
El siguiente árbol es el árbol que deriva de un recorrido BFS de un grafo dirigido G . ¿Cuál de las siguientes aristas no puede estar en G ?



- (a) (F, C) . (b) (D, A) . (c) (A, E) . (d) (G, E) .

Ejercicio 4.

Se aplicó el recorrido DFS sobre el grafo dirigido de la siguiente figura, comenzando en el vértice 2. Asumir que las listas de adyacencias están ordenadas de menor a mayor.

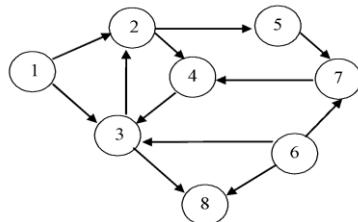


¿Cuál de las siguientes opciones corresponde al listado postorden de los vértices del grafo?

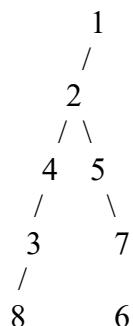
- (a) 2 0 6 4 3 5 7 1 8.
- (b) 4 5 3 6 7 0 1 8 2.
- (c) 4 5 3 6 1 7 0 8 2.
- (d) 2 0 1 8 6 7 4 3 5.

Ejercicio 5.

Dado el siguiente grafo dirigido, en el siguiente bosque abarcador del DFS realizado a partir del vértice (1): 1, 2, 4, 3, 8, 5, 7, 6, habrá ...



- (a) 1 arco de cruce.
- (b) 2 arcos de cruce.
- (c) Más de 2 arcos de cruce.
- (d) Ninguna de las opciones.



Arcos del árbol: (1, 2), (2, 4), (4, 3), (3, 8), (2, 5), (5, 7).

Arcos de cruce: (7, 4), (6, 3), (6, 7), (6, 8).

Arcos de avance: (1, 3).

Arcos de retroceso: (3, 2).

Ejercicio 6.

Dado el grafo de la Figura 5, indicar cuál de las siguientes posibilidades es una ordenación topológica válida.

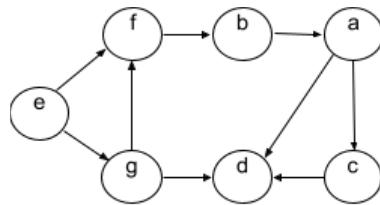


Figura 5

- (a) e, g, d, f, b, a, c.
- (b) e, g, f, b, a, c, d.**
- (c) Existe más de una posible ordenación topológica válida.
- (d) Ninguna de las otras respuestas es correcta.

Aristas del grafo:

$$\begin{aligned}e &\rightarrow f. \\e &\rightarrow g. \\f &\rightarrow b. \\g &\rightarrow f. \\g &\rightarrow d. \\b &\rightarrow a. \\a &\rightarrow c. \\a &\rightarrow d. \\c &\rightarrow d.\end{aligned}$$

Única ordenación topológica válida:

$$e, g, f, b, a, c, d.$$

Ejercicio 7.

Dado el grafo de la Figura 6, indicar cuál de las siguientes posibilidades es una ordenación topológica válida.

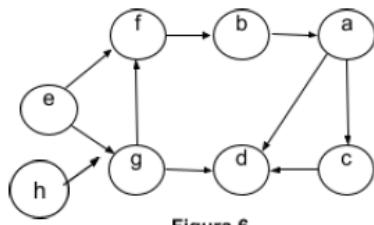


Figura 6

- (a) e, h, g, d, f, b, a, c .
- (b) e, g, f, b, a, c, d, h .
- (c) Existe más de una posible ordenación topológica válida.
- (d) Ninguna de las otras respuestas es correcta.

Aristas del grafo:

$$\begin{aligned} e &\rightarrow f. \\ e &\rightarrow g. \\ h &\rightarrow g. \\ f &\rightarrow b. \\ g &\rightarrow f. \\ g &\rightarrow d. \\ b &\rightarrow a. \\ a &\rightarrow c. \\ a &\rightarrow d. \\ c &\rightarrow d. \end{aligned}$$

Ordenaciones topológicas válidas:

- (i) h, e, g, f, b, a, c, d .
- (ii) e, h, g, f, b, a, c, d .

Ejercicio 8.

Aplicar las versiones de ordenación topológica vistas en clase. 1 (usando arreglo), 2 (usando Cola o Pila) y 3 (usando DFS) del algoritmo que permite obtener la ordenación topológica del DAG de la Figura 7.

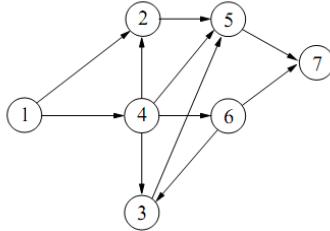


Figura 7

(a) Usando Arreglo.

Paso 1:

1 tiene 0 grado_in
2 tiene 2 grado_in
4 tiene 1 grado_in
5 tiene 3 grado_in
7 tiene 2 grado_in
6 tiene 1 grado_in
3 tiene 2 grado_in

Orden: [1].

Paso 2:

2 tiene 1 grado_in
4 tiene 0 grado_in
5 tiene 3 grado_in
7 tiene 2 grado_in
6 tiene 1 grado_in
3 tiene 2 grado_in

Orden: [1, 4].

Paso 3:

2 tiene 0 grado_in
5 tiene 2 grado_in
7 tiene 2 grado_in
6 tiene 0 grado_in
3 tiene 1 grado_in

Orden: [1, 4, 2].

Paso 4:

5 tiene 1 grado_in
7 tiene 2 grado_in
6 tiene 0 grado_in
3 tiene 1 grado_in

Orden: [1, 4, 2, 6].

Paso 5:

5 tiene 1 grado_in
7 tiene 1 grado_in
3 tiene 0 grado_in

Orden: [1, 4, 2, 6, 3].

Paso 6:

5 tiene 0 grado_in
7 tiene 1 grado_in

Orden: [1, 4, 2, 6, 3, 5].

Paso 7:

7 tiene 0 grado_in

Orden: [1, 4, 2, 6, 3, 5, 7].

(b) Usando Cola.

Paso 1:

1 tiene 0 grado_in
2 tiene 2 grado_in
4 tiene 1 grado_in
5 tiene 3 grado_in
7 tiene 2 grado_in
6 tiene 1 grado_in
3 tiene 2 grado_in

Cola: [1].

Cola: [].

Orden: [1].

Paso 2:

2 tiene 1 grado_in
4 tiene 0 grado_in
5 tiene 3 grado_in
7 tiene 2 grado_in
6 tiene 1 grado_in
3 tiene 2 grado_in

Cola: [4].

Cola: [].

Orden: [1, 4].

Paso 3:

2 tiene 0 grado_in
5 tiene 2 grado_in
7 tiene 2 grado_in
6 tiene 0 grado_in
3 tiene 1 grado_in

Cola: [2, 6].

Cola: [6].

Orden: [1, 4, 2].

Paso 4:

5 tiene 1 grado_in
7 tiene 2 grado_in
3 tiene 1 grado_in

Cola: [6].

Cola: [].

Orden: [1, 4, 2, 6].

Paso 5:

5 tiene 1 grado_in
7 tiene 1 grado_in
3 tiene 0 grado_in

Cola: [3].

Cola: [].

Orden: [1, 4, 2, 6, 3].

Paso 6:

5 tiene 0 grado_in
7 tiene 1 grado_in

Cola: [5].

Cola: [].
Orden: [1, 4, 2, 6, 3, 5].

Paso 7:

7 tiene 0 grado_in

Cola: [7].
Cola: [].
Orden: [1, 4, 2, 6, 3, 5, 7].

(c) Usando Pila.

Paso 1:

1 tiene 0 grado_in
2 tiene 2 grado_in
4 tiene 1 grado_in
5 tiene 3 grado_in
7 tiene 2 grado_in
6 tiene 1 grado_in
3 tiene 2 grado_in

Pila: [1].
Pila: [].
Orden: [1].

Paso 2:

2 tiene 1 grado_in
4 tiene 0 grado_in
5 tiene 3 grado_in
7 tiene 2 grado_in
6 tiene 1 grado_in
3 tiene 2 grado_in

Pila: [4].
Pila: [].
Orden: [1, 4].

Paso 3:

2 tiene 0 grado_in
5 tiene 2 grado_in
7 tiene 2 grado_in
6 tiene 0 grado_in
3 tiene 1 grado_in

Pila: [2, 6].

Pila: [2].

Orden: [1, 4, 6].

Paso 4:

5 tiene 2 grado_in

7 tiene 1 grado_in

3 tiene 0 grado_in

Pila: [2, 3].

Pila: [2].

Orden: [1, 4, 6, 3].

Paso 5:

5 tiene 1 grado_in

7 tiene 1 grado_in

Pila: [2].

Pila: [].

Orden: [1, 4, 6, 3, 2].

Paso 6:

5 tiene 0 grado_in

7 tiene 1 grado_in

Pila: [5].

Pila: [].

Orden: [1, 4, 6, 3, 2, 5].

Paso 7:

7 tiene 0 grado_in

Pila: [7].

Pila: [].

Orden: [1, 4, 6, 3, 2, 5, 7].

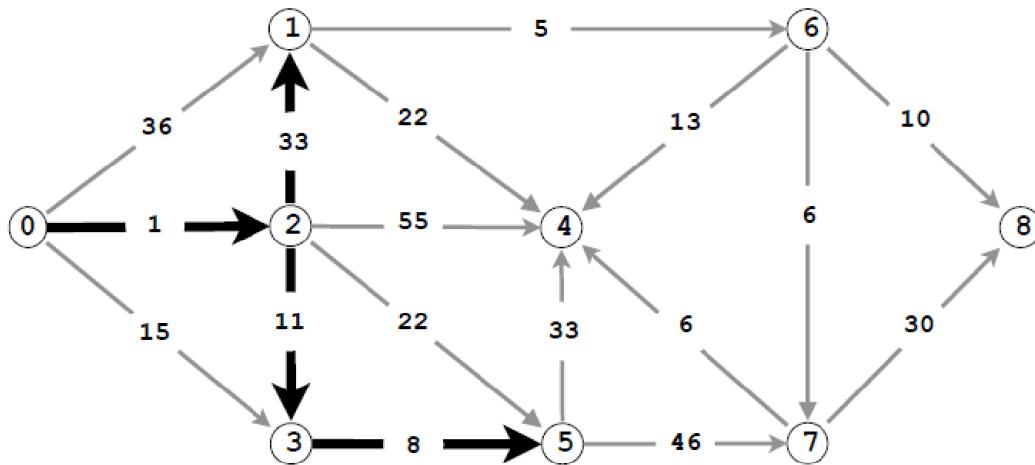
(d) Usando DFS.

Pila: [7, 5, 2, 3, 6, 4, 1].

Orden: [1, 4, 6, 3, 2, 5, 7].

Ejercicio 9.

Se ejecuta el algoritmo de Dijkstra sobre el siguiente dígrafo pesado.



(a) La siguiente tabla contiene los valores luego de haberse procesado los vértices: 0, 2, 3, 5 y 1. Continuar la ejecución del algoritmo completando la tabla con los valores correspondientes.

Nro. de iteración	Vértice	Distancia (0, v)	Vértice Previo	Visitado
1°	0	0	-	1
5°	1	∞ 36 34	0 2	1
2°	2	∞ 1	0	1
3°	3	∞ 15 12	0 2	1
9°	4	∞ 56 53 52 51	2 5 6 7	0 1
4°	5	∞ 23 20	2 3	1
6°	6	∞ 39	1	0 1
7°	7	∞ 66 45	5 6	0 1
8°	8	∞ 49	6	0 1

(b) Completar la secuencia de vértices según el orden en el que el algoritmo de Dijkstra los toma (es decir, los considera “visitados”). Recordar que la ejecución del algoritmo comenzó por el vértice “0”.

0 2 3 5 1 6 7 8 4

(c) Dibujar, sobre el grafo, los arcos (con trazo más grueso) del árbol abarcador resultante.



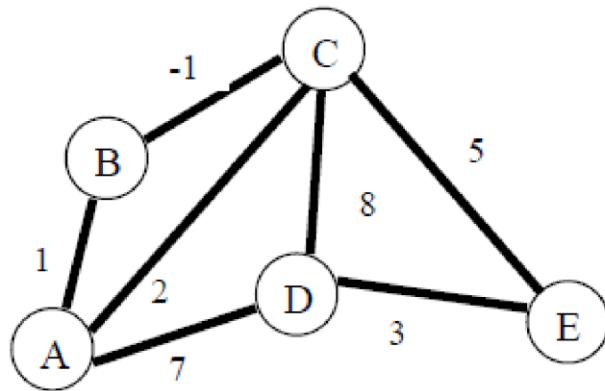
(d) Recuperar los vértices que componen los caminos de costo mínimo obtenidos con el algoritmo de Dijkstra para los siguientes pares: (0, 5), (0, 7).

(0, 5): 0 2 3 5.

(0, 7): 0 2 1 6 7.

Ejercicio 10.

Dado el grafo pesado de la figura.



(a) ¿El algoritmo de Dijkstra funciona correctamente en este caso en particular, tomando como vértice origen a A?

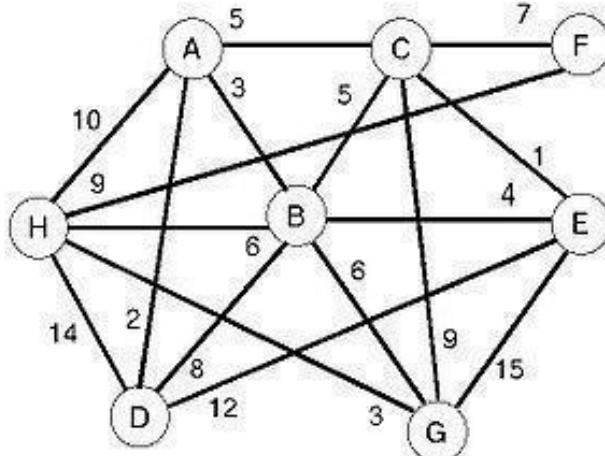
El algoritmo de Dijkstra no funciona correctamente en este caso en particular, tomando como vértice origen a A.

(b) Si la respuesta es afirmativa, aplicarlo. Si la respuesta es negativa, fundamentar por qué no funciona el algoritmo.

El algoritmo no funciona porque hay una arista con peso negativo, la arista de B a C con peso -1.

Ejercicio 11.

(a) Dado el siguiente grafo, ejecutar el algoritmo de Dijkstra, partiendo del vértice H.



Nro. de iteración	Vértice	Distancia (H, v)	Vértice Previo	Visitado
5°	A	∞ 10 9	H B	1
3°	B	∞ 6	H	1
8°	C	∞ 12 11	G E	1
7°	D	∞ 14 11	H A	1
6°	E	∞ 18 10	G B	1
4°	F	∞ 9	H	1
2°	G	∞ 3	H	1
1°	H	0	-	1

(b) ¿Cuáles fueron los costos intermedios encontrados por el algoritmo para encontrar el camino mínimo a E?

- (i) 14, 11.
- (ii) 18, 10.
- (iii) 18, 11, 10.
- (iv) 12, 11, 10.
- (v) Ninguna de las anteriores.

(c) ¿Cuáles fueron los vértices intermedios encontrados por el algoritmo para encontrar el camino mínimo a E?

- (i) A, C, B.
- (ii) A, B, G.
- (iii) G, B.

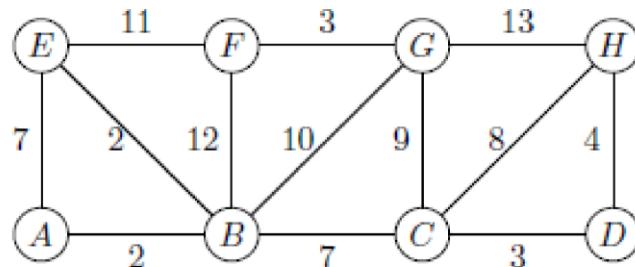
- (iv) D, B.
- (v) A, C.
- (vi) B.

(d) ¿En qué iteración del algoritmo fue tomado el vértice C?

- (i) 4° .
- (ii) 5° .
- (iii) 6° .
- (iv) 7° .
- (v) 8° .

Ejercicio 12.

Ejecutar el algoritmo de Prim en el siguiente grafo, partiendo del vértice A. ¿Cuál es la suma de los pesos de la primera, tercera y quinta arista seleccionadas según el algoritmo?

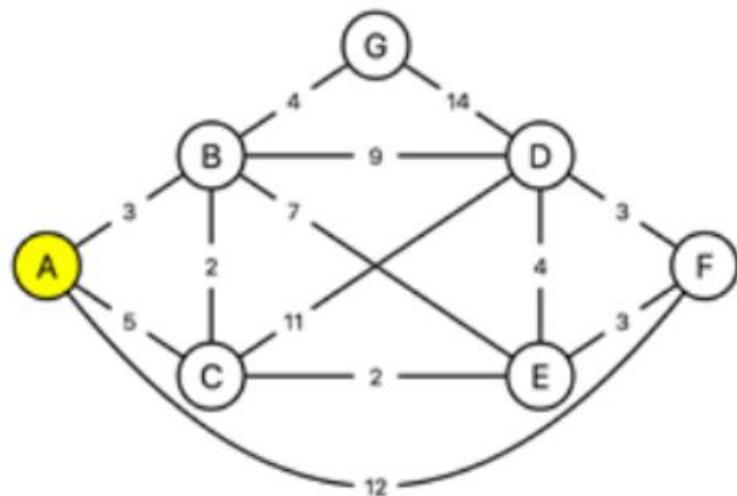


- (a) 9. (b) 10. (c) 11. (d) 12. (e) 13.

Nro. de iteración	Vértice v	Costo (v, w)	Vértice w	Visitado
1°	A	0	-	1
2°	B	∞ 2	A	1
4°	C	∞ 7	B	1
5°	D	∞ 3	C	1
3°	E	∞ 7 2	A B	1
8°	F	∞ 12 11 3	B E G	1
7°	G	∞ 10 9	B C	1
6°	H	∞ 8 4	C D	1

Ejercicio 13.

Obtener el árbol de expansión mínima utilizando el algoritmo de Prim en el siguiente grafo comenzando del vértice A.



(a) Dibujar cómo evoluciona la construcción del árbol en cada paso.



(b) Mostrar la ejecución del algoritmo en la tabla que aparece más abajo.

Nro. de iteración	Vértice v	Costo (v, w)	Vértice w	Visitado
1°	A	0	-	1
2°	B	∞ 3	A	1
3°	C	∞ 5 2	A B	1
4°	D	∞ 9 4 3	B E F	1
5°	E	∞ 7 2	B C	1
6°	F	∞ 12 3	A E	1
7°	G	∞ 4	B	1

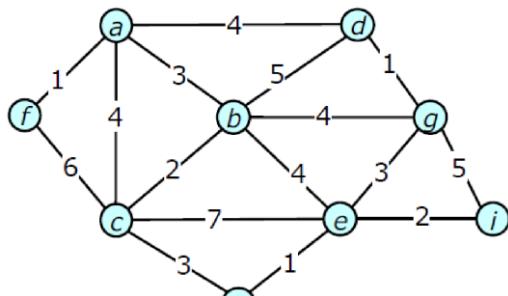
(c) *Expresar el orden de ejecución del algoritmo (en cuanto a su eficiencia). Justificar la respuesta.*

El orden de ejecución del algoritmo (en cuanto a su eficiencia) es:

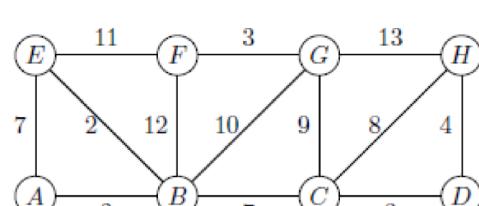
- Si se implementa con una tabla secuencial, $O(|V|^2)$.
- Si se implementa con *heap*, $O(|E| \log |V|)$.

Ejercicio 14.

Obtener el árbol de expansión mínima utilizando el algoritmo de Kruskal en los siguientes grafos, dibujando cómo evoluciona la construcción del árbol en cada paso. Expresar el orden de ejecución del algoritmo (en cuanto a su eficiencia). Justificar la respuesta.



Grafo 1

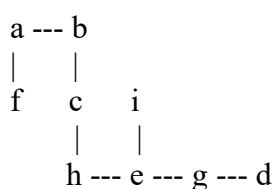


Grafo 2

(a) Completar la secuencia de arcos del árbol abarcador mínimo, según el orden en que el algoritmo los incluye en el árbol.

Grafo 1:

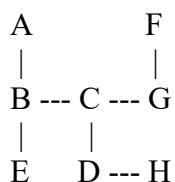
- (a, f) → 1.
- (d, g) → 1.
- (e, h) → 1.
- (b, c) → 2.
- (e, i) → 2.
- (a, b) → 3.
- (c, h) → 3.
- (e, g) → 3.
- (a, c) → 4.
- (a, d) → 4.
- (b, e) → 4.
- (b, g) → 4.
- (b, d) → 5.
- (g, i) → 5.
- (c, f) → 6.
- (c, e) → 7.



Grafo 2:

- (A, B) → 2.

(B, E) → 2.
(C, D) → 3.
(F, G) → 3.
(D, H) → 4.
(A, E) → 7.
(B, C) → 7.
(C, H) → 8.
(C, G) → 9.
(B, G) → 10.
(E, F) → 11.
(B, F) → 12.
(G, H) → 13.



(b) ¿Cuál es el costo del árbol abarcador resultante?

Grafo 1: 16.

$$1 + 1 + 1 + 2 + 2 + 3 + 3 + 3 = 16.$$

Grafo 2: 30.

$$2 + 2 + 3 + 3 + 4 + 7 + 9 = 30.$$

(c) ¿Cuántos arcos fueron descartados durante el desarrollo del algoritmo hasta encontrar el árbol abarcador resultante?

Grafo 1: 8.

(a, c) → 4.
(a, d) → 4.
(b, e) → 4.
(b, g) → 4.
(b, d) → 5.
(g, i) → 5.
(c, f) → 6.
(c, e) → 7.

Grafo 2: 6.

(A, E) → 7.

(C, H) → 8.

(B, G) → 10.

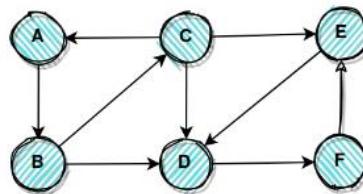
(E, F) → 11.

(B, F) → 12.

(G, H) → 13.

Ejercicio 15.

Indicar cuáles son las componentes fuertemente conexas para el siguiente grafo dirigido, utilizando el algoritmo de Kosaraju comenzando por el vértice “A” (tanto los vértices como los adyacentes se procesan alfabéticamente). Mostrar, paso a paso, la ejecución del algoritmo.



1. Aplicar DFS al grafo rotulando los vértices del grafo en post-orden (apilar):

Pila: [E, F, D, C, B, A].

2. Construir el grafo reverso del grafo (invertir los arcos):

B → A.
C → B.
D → B.
A → C.
D → C.
E → C.
F → D.
D → E.
E → F.

3. Aplicar DFS al grafo reverso comenzando por los vértices de mayor rótulo (tope de la pila):

Iteración 1 (vértice A): A → C → B → A.
Iteración 2 (vértice B): Ya visitado.
Iteración 3 (vértice C): Ya visitado.
Iteración 4 (vértice D): D → E → F → D.
Iteración 5 (vértice F): Ya visitado.
Iteración 6 (vértice E): Ya visitado.

4. Cada árbol de expansión resultante del paso 3 es una componente fuertemente conexa:

Las componentes fuertemente conexas del grafo dirigido son: {A, B, C} y {D, E, F}.