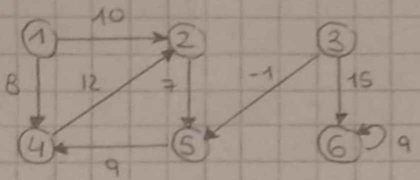
1)

TEORÍA GRAFOS
EJERCITACIÓN

DFS → Profundidad
BFS → Amplitud

(1)



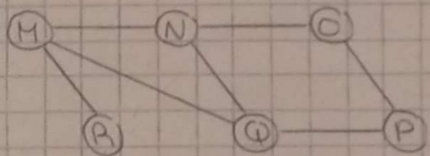
a) DFS desde V=1 (numerando en PRE-ORDEN) raíz, hMenor ...

1, 2, 5, 4

b) BFS desde V=1 (numerando en PRE-ORDEN)

1, 2, 4, 5

(2)



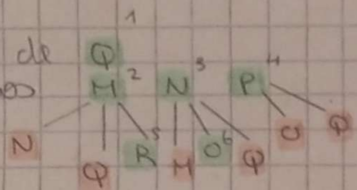
Recorrido BFS válido para la figura

M N O P Q R ✗ Q M N P R O ✓

N Q M P O R ✗ Q M N P O R ✗!

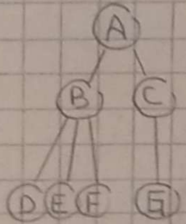
Nota numerando en PRE-ORDEN

resto de vecinos



→ Por eso vale QMNPRO

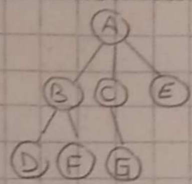
(3) Dado un árbol que deriva de un recorrido BFS de un grafo dirigido G

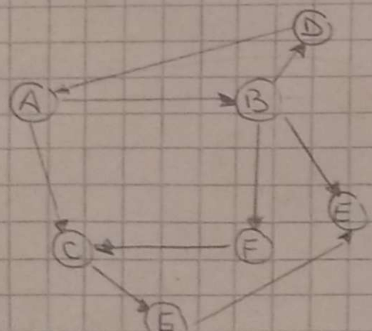


¿Qué arista NO está en G?

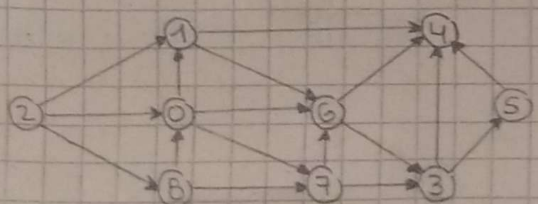
(F, C) ✓ (A, E) ✗ → si no el árbol debería ser

(D, A) ✓ (G, E) ✓





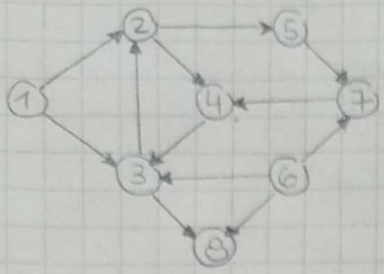
(4) Recorrido DFS desde V=2 (numerado de menor a mayor)



Listado post-orden? (raíz al final)

4 5 3 6 1 7 0 8 2

⑤ Grafo dirigido, bosque abarcador DFS desde V = 1 ¿Cuántos arcos de cruce tendré?

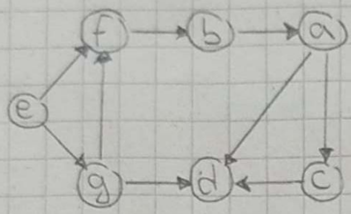


1 2 4 3 8 5 7 6

1 __ 2
2 __ 4
4 __ 3
3 __ 2 cruce
3 __ 8 → retroceso
2 __ 5
5 __ 7
7 __ 4 cruce → retroceso
1 __ 3 cruce
1 __ 6 avance
6 __ 3 cruce
6 __ 7 cruce
6 __ 8 cruce

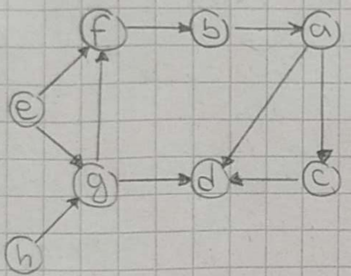
Rta Más de 2 arcos de cruce.

⑥



e g f b a c d (es el único válido)

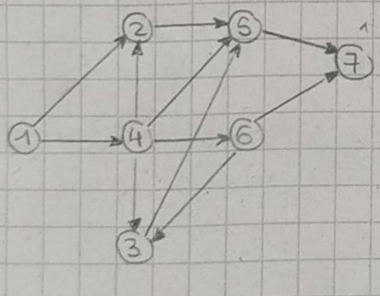
⑦



e/h g f b a c d

Rta c) ∃ más de una ordenación topológica válida

⑧



Ordenación topológica

V1 arreglo

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

V1-V4-V2-V6-V3-V5-V7

V2 Pila

Pila		1	2	3	4	5	6	7
		0	2	2	1	3	1	2
V7	1		1		0			
V5	4		0	1		2	0	
V3	6			0				1
V6	3					1		
V2	2					0		
V4	5							0
V1								

V1-V4-V6-V3-V2-V5-V7

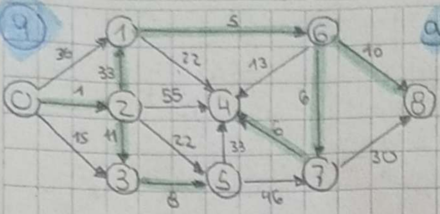
V3 DFS (en post-orden)

V1
V4
V6
V3
V2
V5
V7

V1-V4-V6-V3-V2-V5-V7

Recorro es post-orden, apilo y luego desapilo y luego el sort topológico

9)

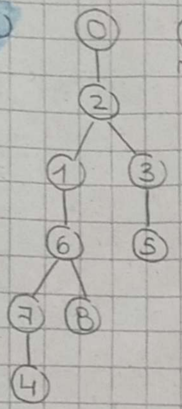


a)

Pos	V	Distancia (o,v)	Vertice Previo	con
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	1
4°	5	~~∞~~ ~~23~~ 20	~~2~~ 3	1
6°	6	~~∞~~ 39	1	1
7°	7	~~∞~~ ~~66~~ 45	~~5~~ 6	1
8°	8	~~∞~~ 49	6	1

b) 0 2 3 5 1 6 7 8 4

c)

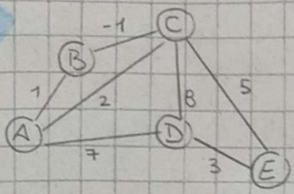


(en el grafo marcado en verde)

d) (0,5) = 0 - 2 - 3 - 5

(0,7) = 0 - 2 - 1 - 6 - 7

10)

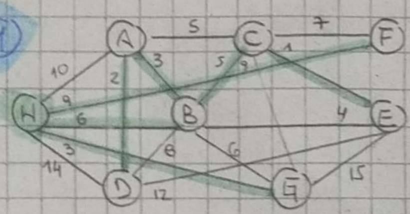


a) Si origen → A, puedo usar Dijkstra?

Pos	V	Distancia (o,v)	Vertice Previo	con
1°	A	0	–	1
2°	B	~~∞~~ 1	A	1
3°	C	~~∞~~ ~~2~~ 0	~~A~~ B	1
5°	D	~~∞~~ 7	A	1
4°	E	~~∞~~ 5	C	1

Vemos que en este caso puedo usar Dijkstra y funciona correctamente, pero no siempre sirve dicho algoritmo con pesos negativos

11)

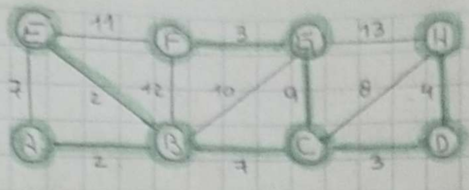


b) (H,E) = 6 - 5 - 1 (costos intermedios)

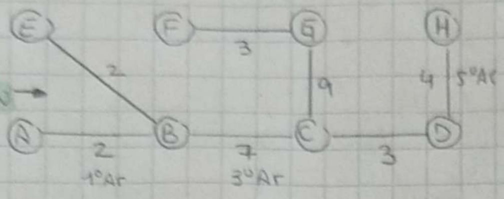
c) (H,E) = H - B - C - E (vertices intermedios)

Pos	V	Distancia (o,v)	Vertice Previo	con
5°	A	~~∞~~ ~~10~~ 9	~~0~~ ~~H~~ B	~~0~~ 1
3°	B	~~∞~~ 6	~~0~~ H	~~0~~ 1
4°	C	~~∞~~ ~~12~~ 8	~~0~~ ~~G~~ B	~~0~~ 1
8°	D	~~∞~~ ~~14~~ 11	~~0~~ ~~H~~ A	~~0~~ 1
6°	E	~~∞~~ ~~18~~ ~~10~~ 9	~~0~~ ~~G~~ ~~B~~ C	~~0~~ 1
7°	F	~~∞~~ 9	~~0~~ H	~~0~~ 1
2°	G	~~∞~~ 3	~~0~~ H	~~0~~ 1
1°	H	0	–	1

(12)



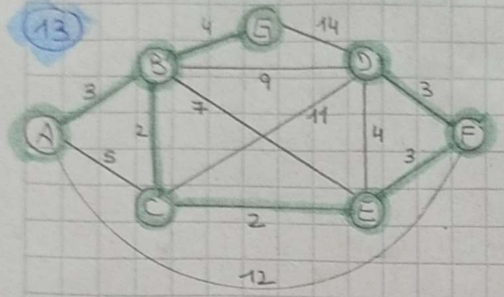
ARBOL DE EXPANSION MÍNIMA →



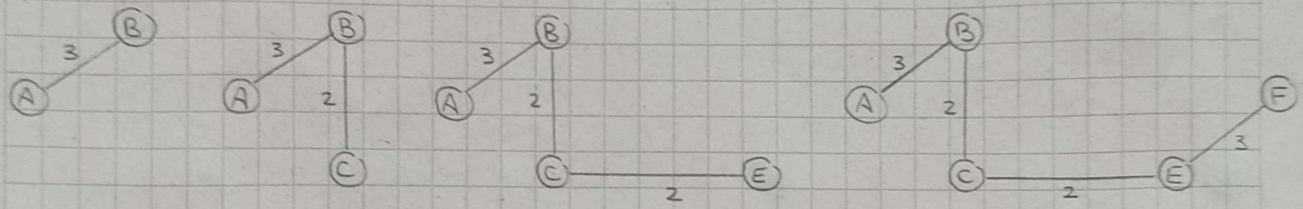
Pos	V	Costo	Vp	Con
1°	A	0	0	1
2°	B	~~∞~~ 2	~~0~~ A	~~0~~ 1
4°	C	~~∞~~ 7	~~0~~ B	~~0~~ 1
5°	D	~~∞~~ 3	~~0~~ C	0 1
3°	E	~~∞~~ ~~7~~ 2	~~0~~ ~~A~~ B	~~0~~ 1
8°	F	~~∞~~ ~~12~~ ~~11~~ 3	~~0~~ ~~B~~ ~~E~~ G	~~0~~ 1
7°	G	~~∞~~ ~~10~~ 9	~~0~~ ~~B~~ C	~~0~~ 1
6°	H	~~∞~~ ~~8~~ 4	~~0~~ ~~E~~ D	~~0~~ 1

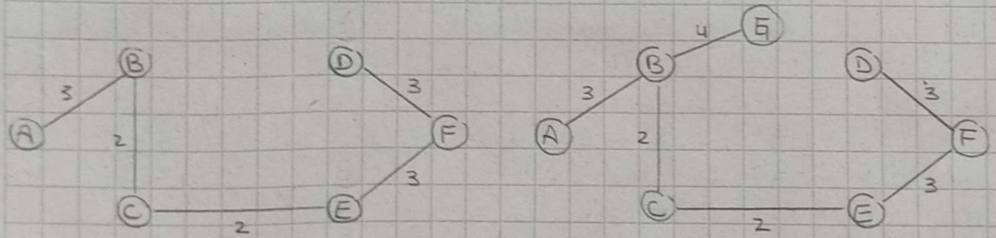
Suma de 1°, 3° y 5° Arista → 13 (2+7+4)

(13)



Pos	V	costo	Vp	con
1°	A	0	0	1
2°	B	~~∞~~ 3	~~0~~ A	~~0~~ 1
3°	C	~~∞~~ ~~5~~ 2	~~0~~ ~~A~~ B	~~0~~ 1
6°	D	~~∞~~ ~~9~~ ~~4~~ 3	~~0~~ ~~B~~ E	~~0~~ 1
4°	E	~~∞~~ ~~7~~ 2	~~0~~ ~~B~~ C	~~0~~ 1
5°	F	~~∞~~ ~~12~~ 3	~~0~~ ~~A~~ E	~~0~~ 1
7°	G	~~∞~~ 4	~~0~~ B	~~0~~ 1

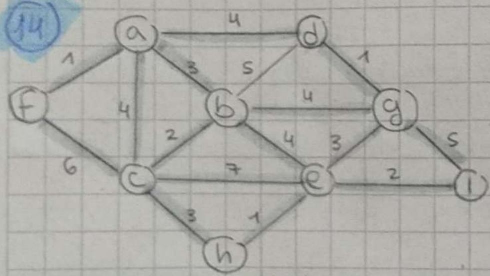




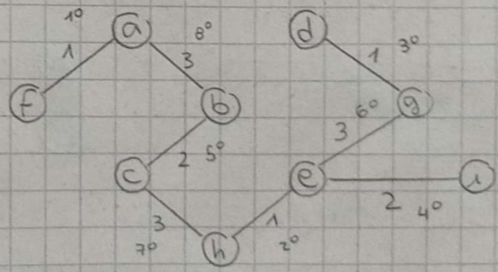
ARBOL EXPANSIÓN MÍNIMA

c) TE = es de $O(|V|^2)$, si implemento una Heap es de $O(|E| \log |V|)$

(14)



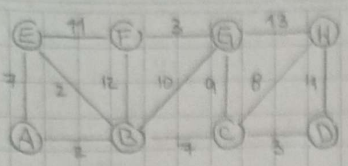
→



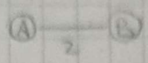
- 1 (a-f)
- 1 (e-h)
- 1 (d-g)
- 2 (e-i)
- 2 (c-b)
- 3 (e-g)
- 3 (c-h)
- 3 (a-b)
- 4 (a-c)

- 4 (a-d)
- 4 (b-g)
- 4 (b-d)
- 5 (g-i)
- 6 (f-c)
- 7 (e-c)

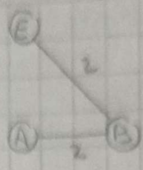
- TE → de $O(|E| \log |V|)$, pues tamaño de Heap = $|E|$ y extraer cada arista lleva $O(\log |V|)$
- Secuencia de arcos (pintado en verde)
- Costo del árbol resultante = 16
- Arcos descartados = 7 (pintados en rojo)



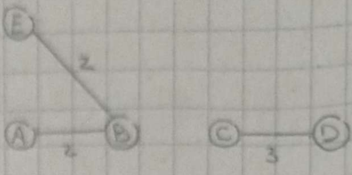
- 2 (A-B)
- 2 (E-B)
- 3 (C-D)
- 3 (F-G)
- 4 (D-H)
- 7 (A-E)
- 7 (B-C)
- 8 (C-H)
- 9 (C-G)
- 10 (B-G)
- 11 (E-F)
- 12 (B-F)
- 13 (G-H)



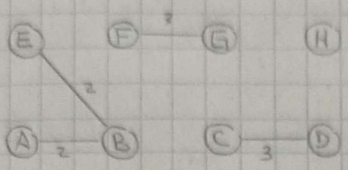
agrego arista (A-B)



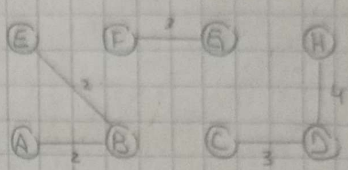
agrego arista (B-C)



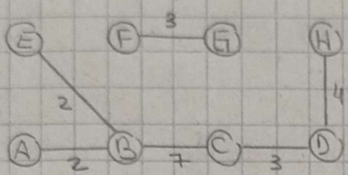
agrego arista (C-D)



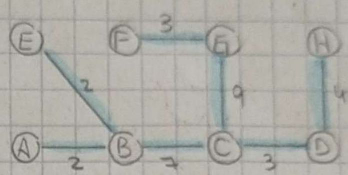
agrego arista (F-G)



agrego arista (D-H)



agrego arista (B-C)



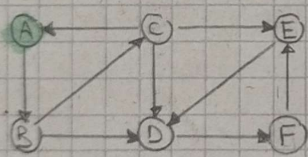
← ÁRBOL DE EXPANSIÓN MÍNIMA

agrego arista (C-G)

- secuencia de arcos en VERDE
- Costo del árbol resultante = 30
- Arcos descartados = 6 (en ROJO)

15) quiero buscar los componentes FUERTEMENTE CONEXAS con KOSARAJU

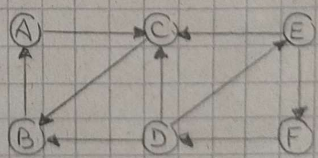
G →



1) Aplico DFS desde A y rotulo es post-orden

E - F - D - C - B - A
1° 2° 3° 4° 5° 6°

2) Construimos grafo reverso G^R →



3) Aplicamos DFS comenzando por los vértices de mayor rótulo en G

A. B - C - A
B. Ya visitado
C. Ya visitado
D. F - E - D
E. Ya visitado
F. Ya visitado

Cada árbol de expansión resultante es una componente fuertemente conexa

COMPONENTES FUERTEMENTE CONEXAS = {A, B, C} y {D, E, F}