



UNIVERSITAT DE
BARCELONA



Grafs III

Algorísmica Avançada | Enginyeria Informàtica

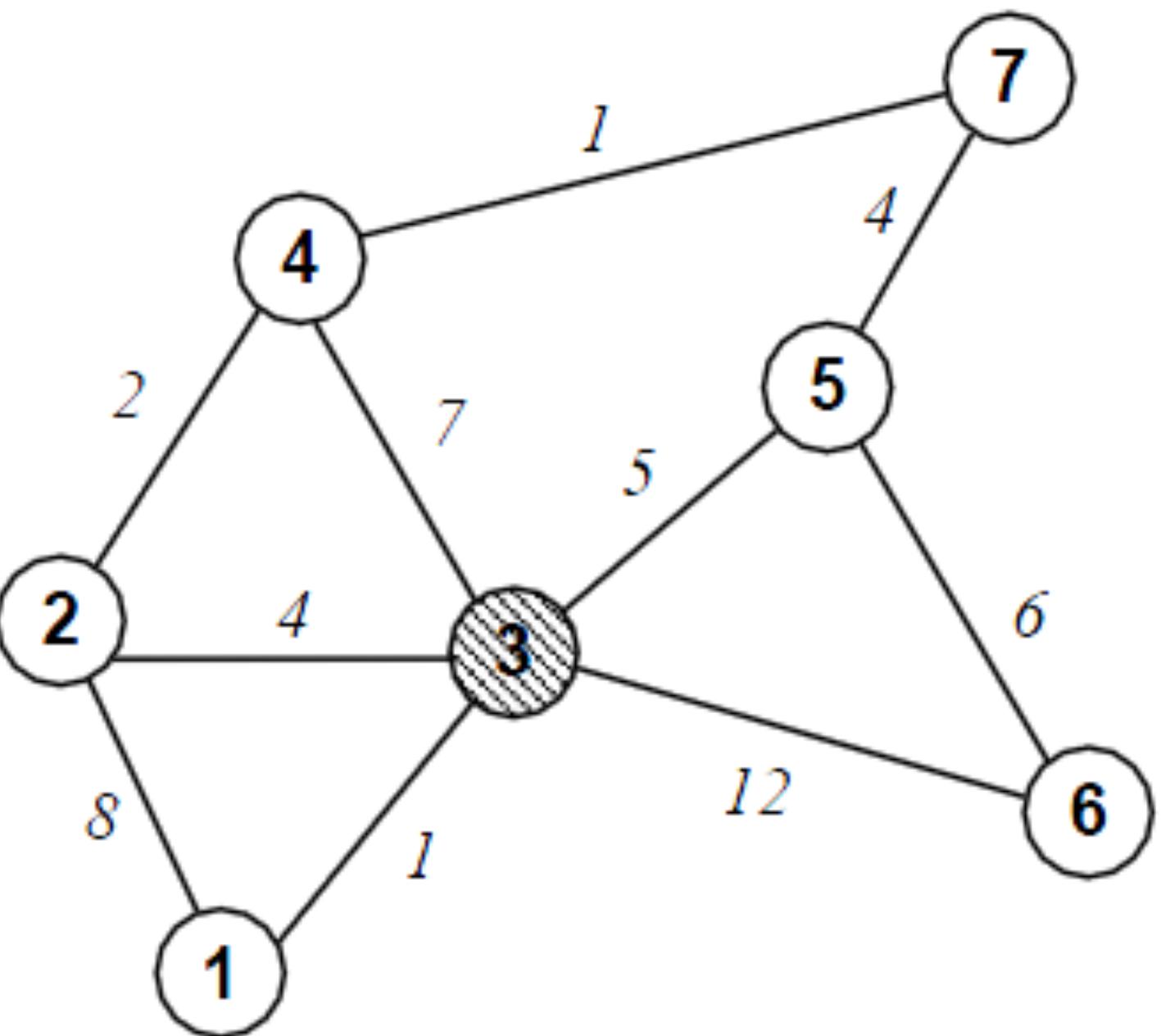
Santi Seguí | 2021-2022

Sessió

- 1. Wrap-up: Dijkstra
- 2. Bellman Ford
 - Trobar camí més curt amb grafs amb pesos
- 3. Ordre Topològic
 - Kahn's Algorithm

Dijkstra

- Algorisme de **Dijkstra**: exemple graf **NO** dirigit



Dijkstra

```
1 function Dijkstra(Graph, source):
2
3     create vertex set Q
4
5     for each vertex v in Graph:
6         dist[v]  $\leftarrow$  INFINITY
7         prev[v]  $\leftarrow$  UNDEFINED
8         add v to Q
10    dist[source]  $\leftarrow$  0
11
12    while Q is not empty:
13        u  $\leftarrow$  vertex in Q with min dist[u]
14
15        remove u from Q
16
17        for each neighbor v of u:          // only v that are still in Q
18            alt  $\leftarrow$  dist[u] + length(u, v)
19            if alt < dist[v]:
20                dist[v]  $\leftarrow$  alt
21                prev[v]  $\leftarrow$  u
22
23    return dist[], prev[]
```

Dijkstra (with HEAP)

```
1 function Dijkstra(Graph, source):
2     dist[source]  $\leftarrow$  0                                // Initialization
3
4     create vertex priority queue Q
5
6     for each vertex v in Graph:
7         if v  $\neq$  source
8             dist[v]  $\leftarrow$  INFINITY                    // Unknown distance from source to v
9             prev[v]  $\leftarrow$  UNDEFINED                  // Predecessor of v
10
11    Q.add_with_priority(v, dist[v])
12
13
14    while Q is not empty:                            // The main loop
15        u  $\leftarrow$  Q.extract_min()                   // Remove and return best vertex
16        for each neighbor v of u:                  // only v that are still in Q
17            alt  $\leftarrow$  dist[u] + length(u, v)
18            if alt < dist[v]
19                dist[v]  $\leftarrow$  alt
20                prev[v]  $\leftarrow$  u
21                Q.decrease_priority(v, alt)
22
23    return dist, prev
```

Dijkstra

- Complexitat
 - For any data structure for the vertex set Q , the running time is in

$$O(|E| \cdot T_{dk} + |V| \cdot T_{em}),$$

where T_{dk} and T_{em} are the complexities of the *decrease-key* and *extract-minimum* operations in Q , respectively.

Implementation	extractMin	Insert/decreasekey	Total
Array			
Binary Heap			

Dijkstra

- Complexitat
 - For any data structure for the vertex set Q , the running time is in

$$O(|E| \cdot T_{dk} + |V| \cdot T_{em}),$$

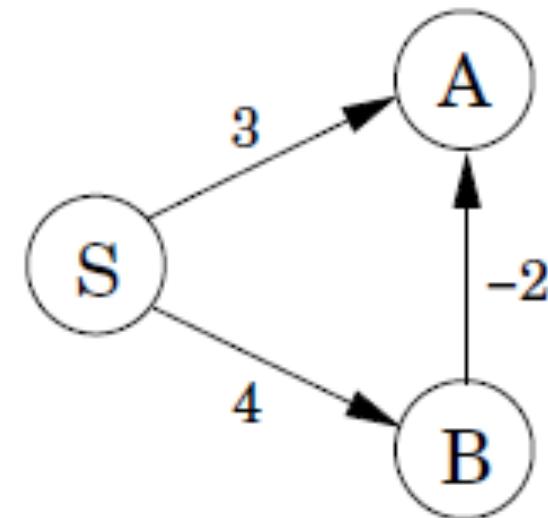
where T_{dk} and T_{em} are the complexities of the *decrease-key* and *extract-minimum* operations in Q , respectively.

Implementation	extractMin	Insert/decreasekey	Total
Array	$O(V)$	$O(1)$	$O(V ^2)$
Binary Heap	$O(\log V)$	$O(\log V)$	$O((V + E)\log(V))$

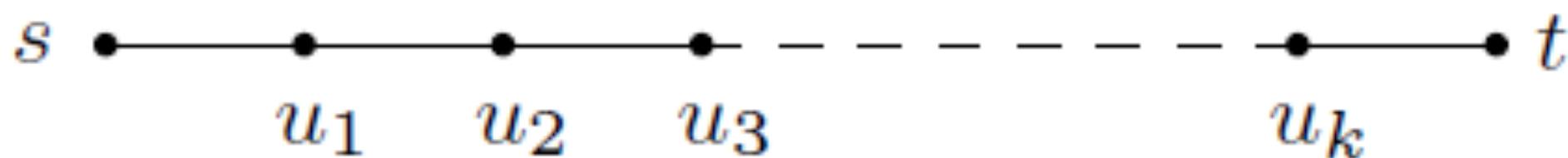
Dijkstra

- **Dijkstra** amb pesos negatius:

```
procedure update(( $u, v$ )  $\in E$ )
     $dist(v) = \min\{dist(v), dist(u) + l(u, v)\}$ 
```



- Amb Dijkstra sempre arribem de s a t amb camí mínim independentment de l'ordre dels pesos de les arestes si aquests són positius. **No amb negatius!**



Bellman-Ford

Bellman-Ford

- **Solució?** Canviem l'algorisme perquè es calculin les distàncies simultàniament.
→ actualitzar totes les arestes $|V|-1$ vegades ($O(|V| \cdot |E|)$)
- Bellman-Ford

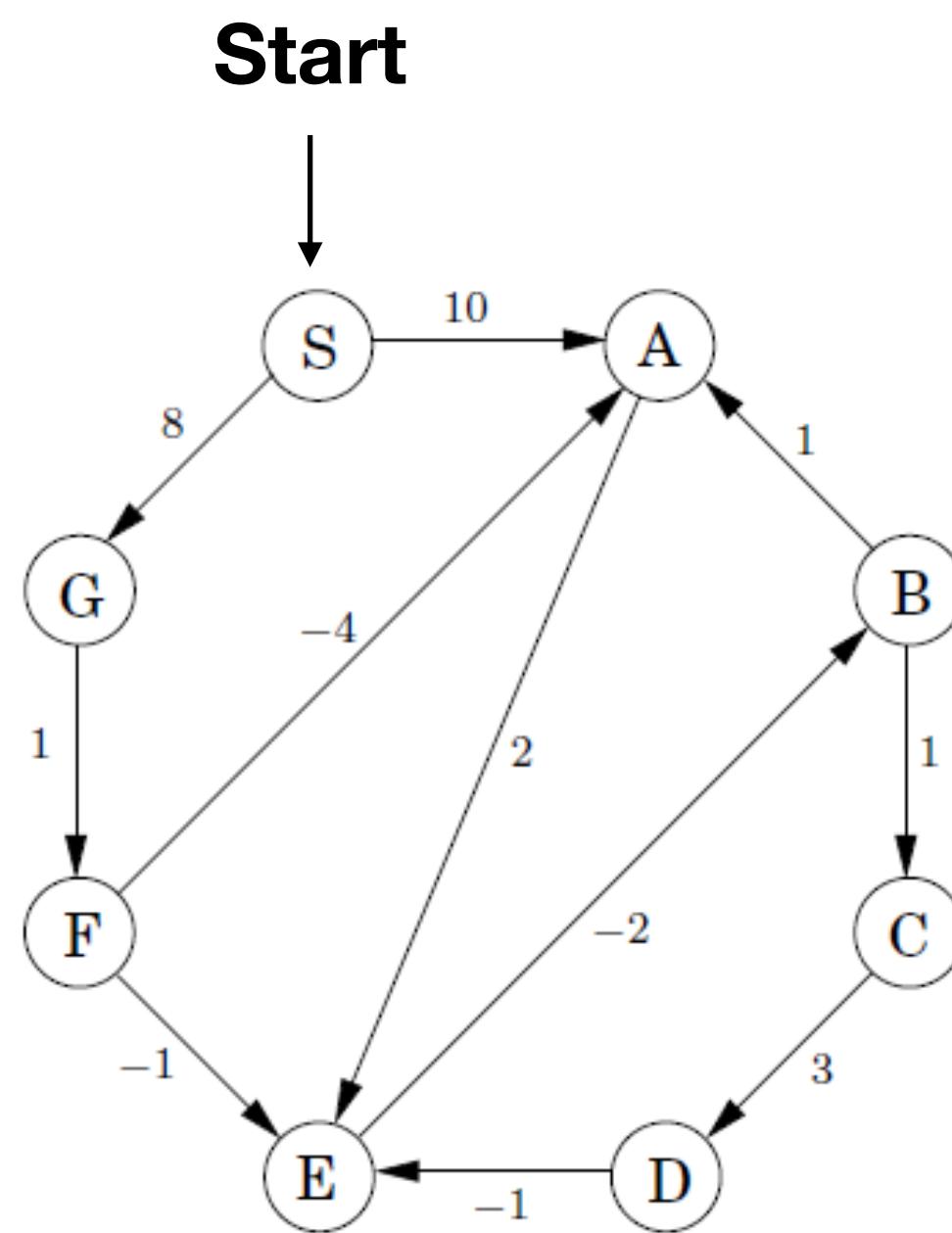
```
procedure shortest-paths( $G, l, s$ )
Input:   Directed graph  $G = (V, E)$ ;
         edge lengths  $\{l_e : e \in E\}$  with no negative cycles;
         vertex  $s \in V$ 
Output:  For all vertices  $u$  reachable from  $s$ ,  $\text{dist}(u)$  is set
         to the distance from  $s$  to  $u$ .

for all  $u \in V$ :
   $\text{dist}(u) = \infty$ 
   $\text{prev}(u) = \text{nil}$ 
  procedure update( $(u, v) \in E$ )
     $\text{dist}(v) = \min\{\text{dist}(v), \text{dist}(u) + l(u, v)\}$ 

 $\text{dist}(s) = 0$ 
repeat  $|V| - 1$  times:
  for all  $e \in E$ :
    update( $e$ )
```

Bellman-Ford

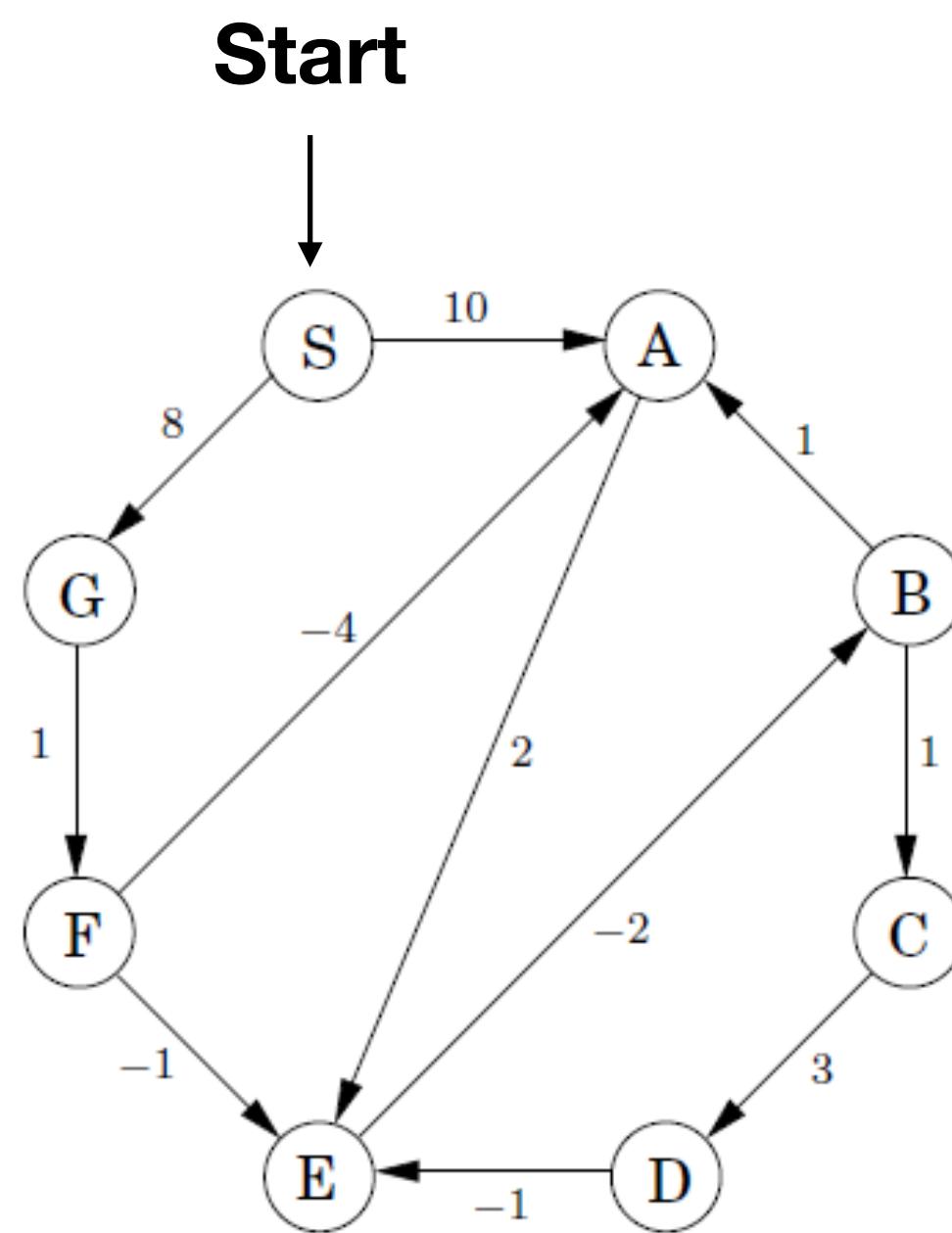
- Implementació: si en una iteració cap aresta e s'actualitza \rightarrow finalitzar



Node	Iteration							
	0	1	2	3	4	5	6	7
S								
A								
B								
C								
D								
E								
F								
G								

Bellman-Ford

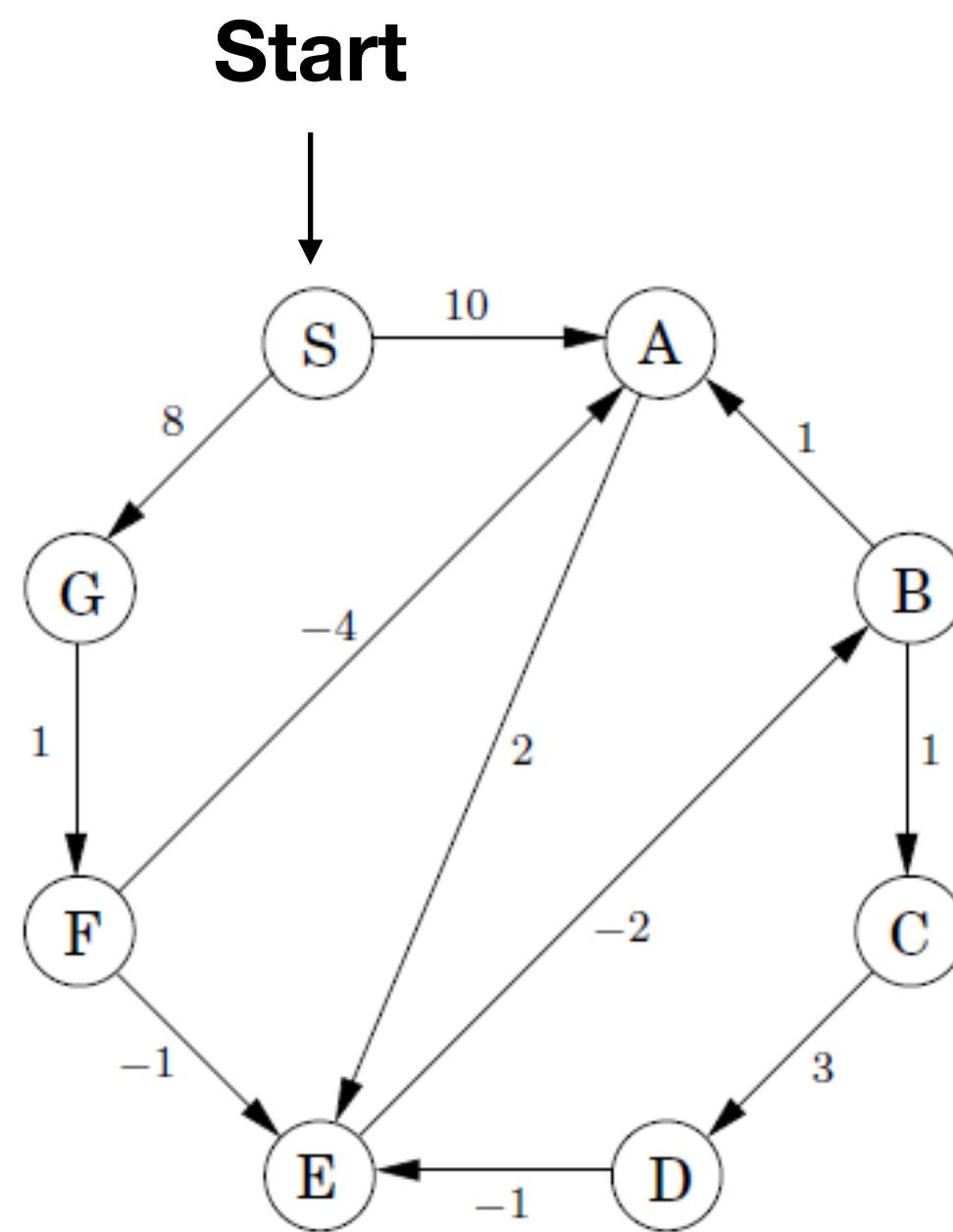
- Implementació: si en una iteració cap aresta e s'actualitza \rightarrow finalitzar



Node	Iteration							
	0	1	2	3	4	5	6	7
S	0							
A	∞							
B	∞							
C	∞							
D	∞							
E	∞							
F	∞							
G	∞							

Bellman-Ford

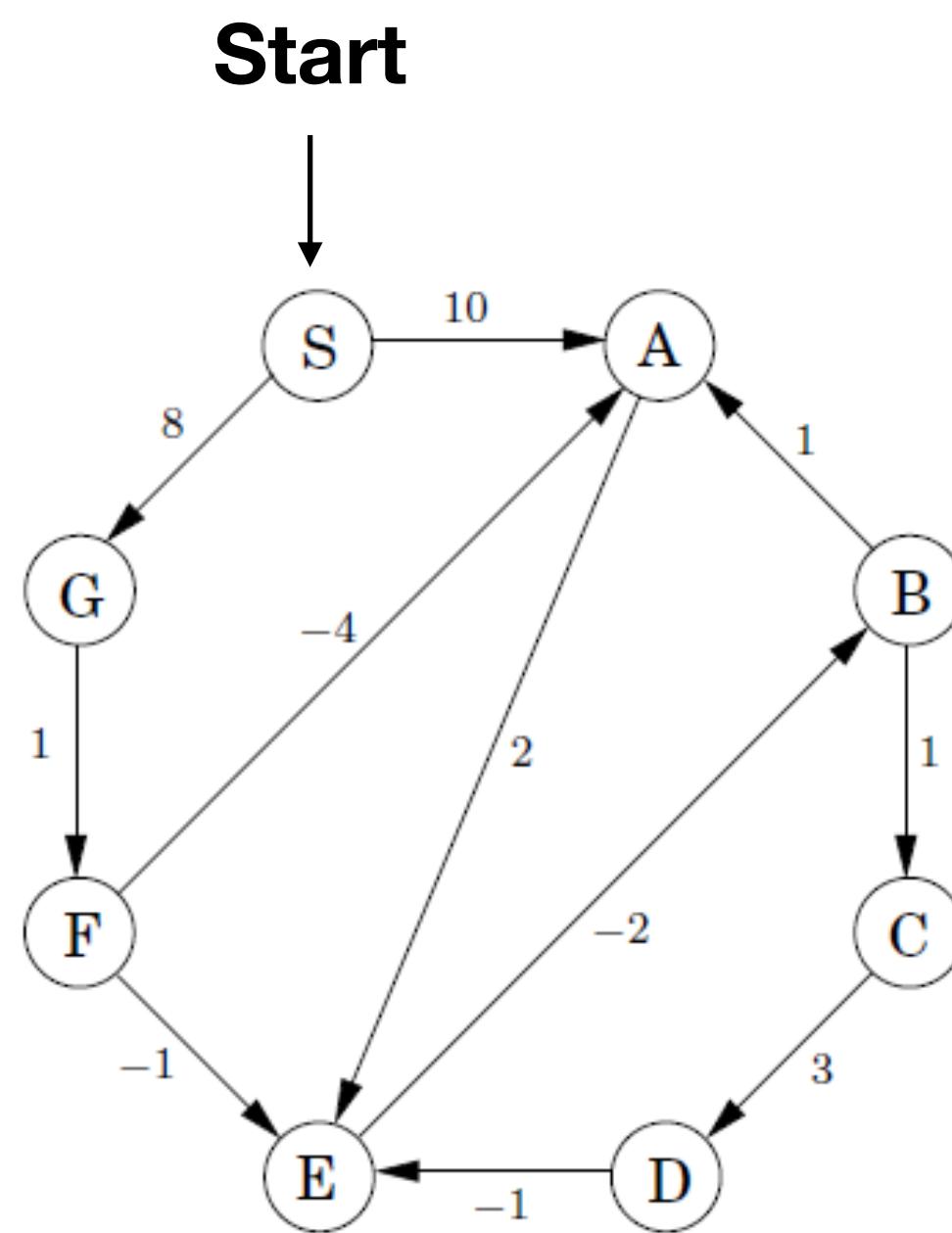
- Implementació: si en una iteració cap aresta e s'actualitza \rightarrow finalitzar



Node	Iteration							
	0	1	2	3	4	5	6	7
S	0	0						
A	∞	10						
B	∞	∞						
C	∞	∞						
D	∞	∞						
E	∞	∞						
F	∞	∞						
G	∞	8						

Bellman-Ford

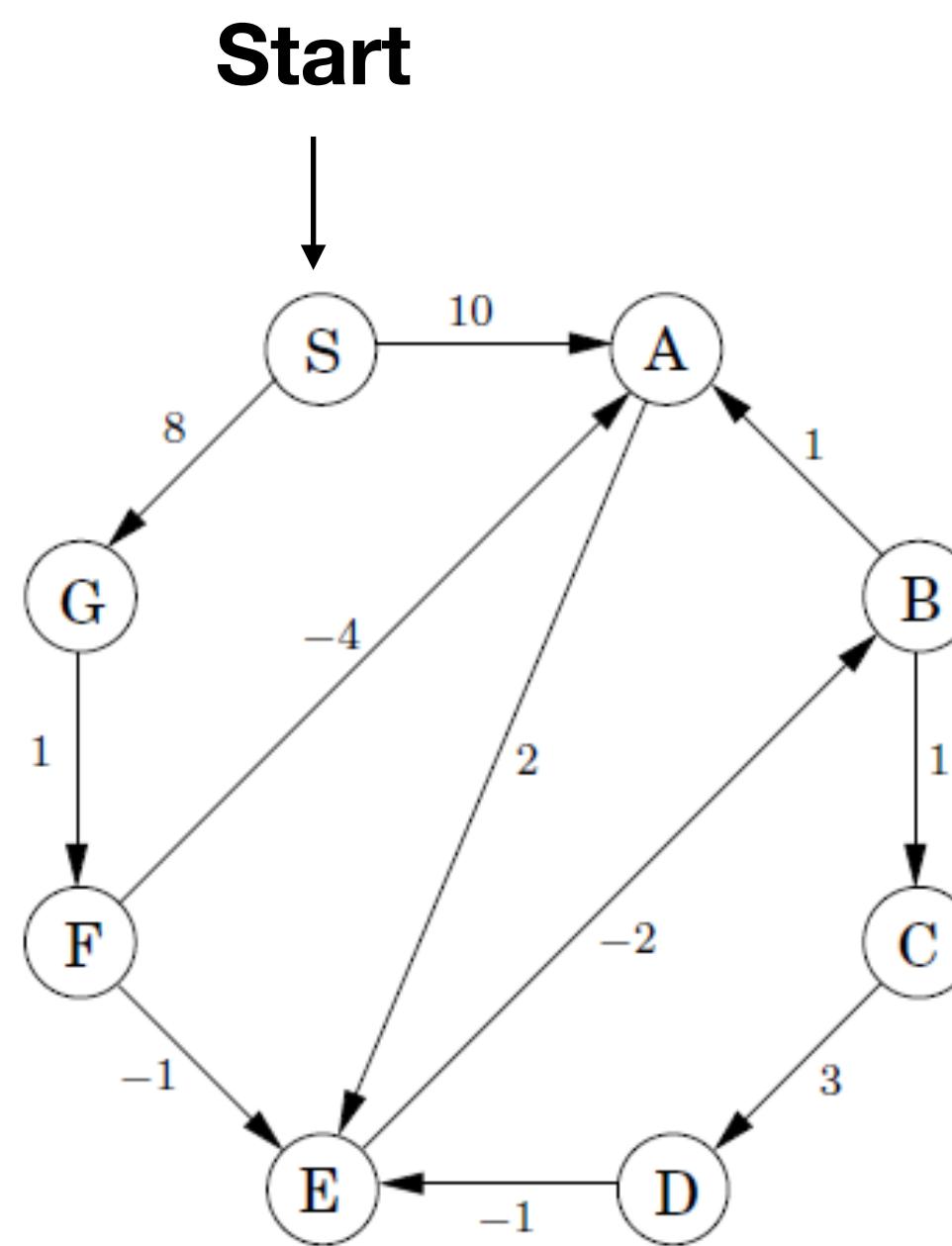
- Implementació: si en una iteració cap aresta e s'actualitza \rightarrow finalitzar



Node	Iteration							
	0	1	2	3	4	5	6	7
S	0	0	0					
A	∞	10	10					
B	∞	∞	∞					
C	∞	∞	∞					
D	∞	∞	∞					
E	∞	∞	12					
F	∞	∞	9					
G	∞	8	8					

Bellman-Ford

- Implementació: si en una iteració cap aresta e s'actualitza \rightarrow finalitzar



Node	Iteration							
	0	1	2	3	4	5	6	7
S	0	0	0	0				
A	∞	10	10	5				
B	∞	∞	∞	10				
C	∞	∞	∞	∞				
D	∞	∞	∞	∞				
E	∞	∞	12	8				
F	∞	∞	9	9				
G	∞	8	8	8				

Complexitat ?

```
procedure shortest-paths( $G, l, s$ )
Input:   Directed graph  $G = (V, E)$ ;
         edge lengths  $\{l_e : e \in E\}$  with no negative cycles;
         vertex  $s \in V$ 
Output:  For all vertices  $u$  reachable from  $s$ ,  $\text{dist}(u)$  is set
         to the distance from  $s$  to  $u$ .

for all  $u \in V$ :
     $\text{dist}(u) = \infty$ 
     $\text{prev}(u) = \text{nil}$ 

 $\text{dist}(s) = 0$ 
repeat  $|V| - 1$  times:
    for all  $e \in E$ :
        update( $e$ )
```

?

Complexitat

```
procedure shortest-paths( $G, l, s$ )
Input:   Directed graph  $G = (V, E)$ ;
         edge lengths  $\{l_e : e \in E\}$  with no negative cycles;
         vertex  $s \in V$ 
Output:  For all vertices  $u$  reachable from  $s$ ,  $\text{dist}(u)$  is set
         to the distance from  $s$  to  $u$ .

for all  $u \in V$ :
     $\text{dist}(u) = \infty$ 
     $\text{prev}(u) = \text{nil}$ 

 $\text{dist}(s) = 0$ 
repeat  $|V| - 1$  times:
    for all  $e \in E$ :
        update( $e$ )
```

$$O(|V| \cdot |E|)$$

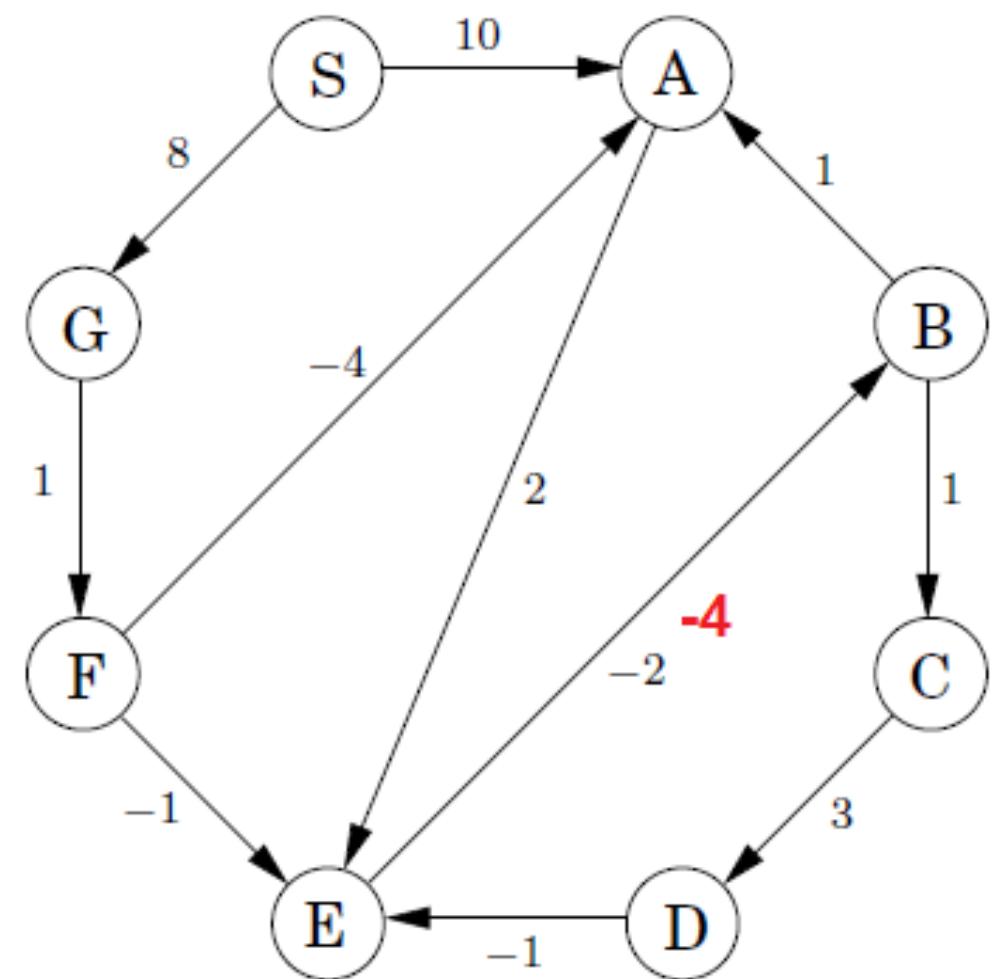
Problema:

Podem trobar **cicles amb cost negatiu** quant tenim
arestes amb costos negatius.

Bellman-Ford

- **Cicles Negatius**

$A \rightarrow E \rightarrow B \rightarrow A$



Els podem trobar amb l'algorisme Bellman-Ford

El camí mínim té com a màxim longitud $|V|-1$

Podem detectar cicles si fem una iteració extra $|V|$

-> Hi ha cicle negatiu si a la iteració $|V|$ alguna aresta és actualitzada

En un **graf general ponderat**, es pot calcular el camí més curt entre dos nodes en temps $O(VE)$ utilitzant l'algoritme de **Bellman-Ford**.

En un **graf ponderat sense pesos negatius**, podem fer-ho millor i calcular el camí més curt entre dos nodes en temps $O(E + V\log V)$ utilitzant l'algorisme de **Dijkstra**.

Si ens trobem amb un graf dirigit acíclic (DAG), encara ho podem millorar!!!

Camí més curt

- Hi ha dos tipus de grafs que no tenen **cicles negatius**:
 - sense pesos negatius
 - sense cicles
- El primer és directe. Per resoldre el camí mínim en grafs dirigits acíclics negatius:
 - Linealitzar usant DFS
 - **Temps lineal!**
- Si posem els negatius dels pesos podem trobar els camins de longitud màxima

Topological Sort

Què és l'ordre topològic?

- Què és l'ordre topològic?
- Quan és important?

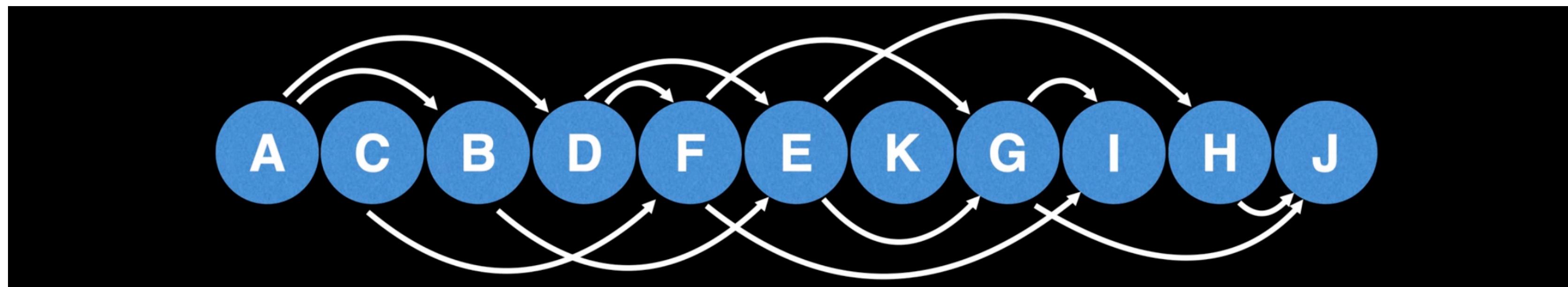


Topological Sort

- Hi ha multitud d'aplicacions reals que poden ser modelades mitjançant grafs dirigits on alguns esdeveniments s'han de ser realitzat abans d'alguns altres:
 - Matricula escolar (amb prerequisits)
 - Construir les dependències d'un programa informàtic
 - Planificació d'un esdeveniment
 - ...

Ordenació topològica

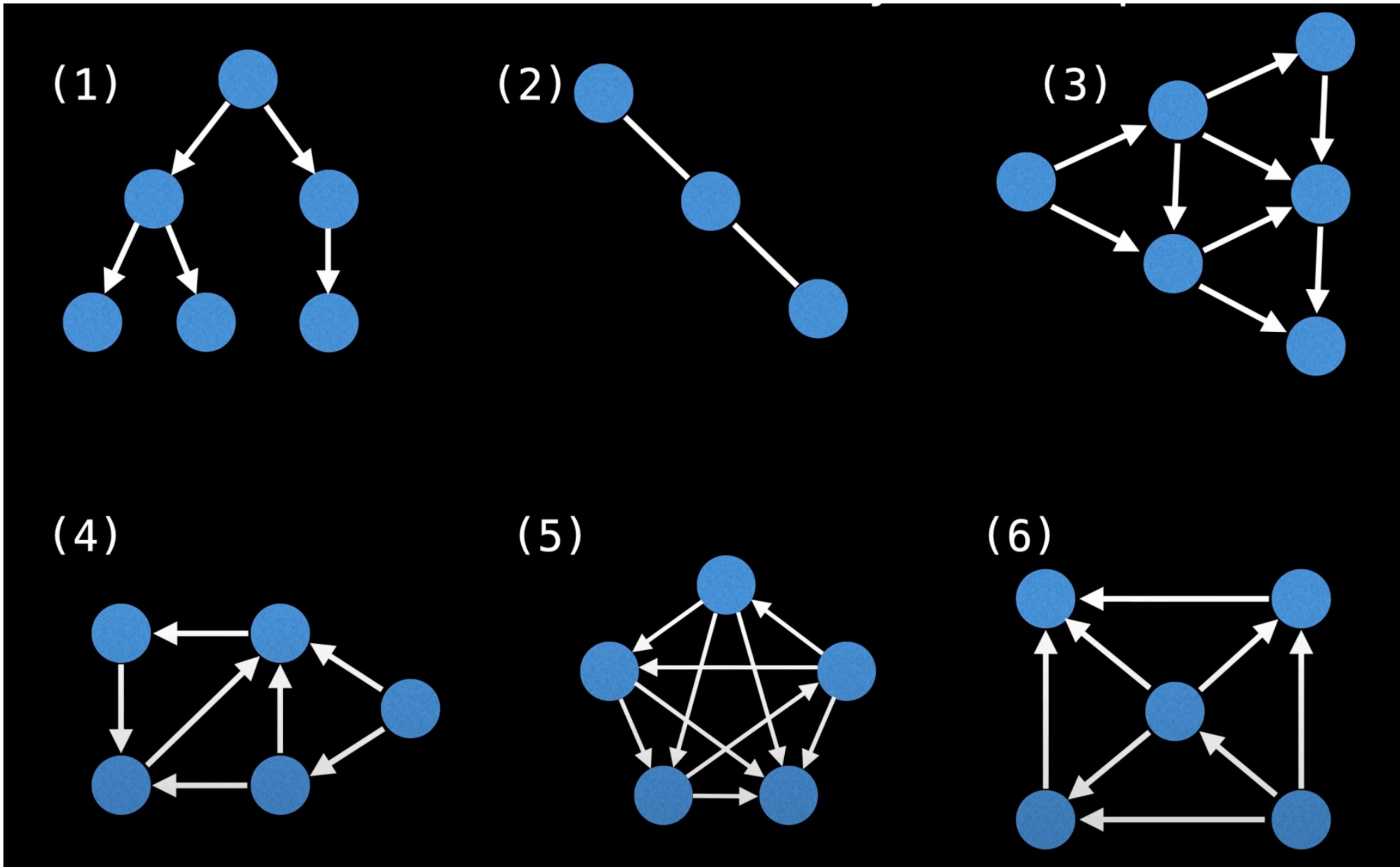
- Una **ordenació topològica** és una ordenació dels nodes en un graf dirigit on per a cada aresta dirigida del node A al node B apareix el node A abans del node B a la seva ordenació.
- L'**algoritme de KAHN** és un algoritme d'**ordenació topològica** simple. Podem trobar l'ordenació topològica del graf amb una complexitat **$O(V+E)$** .
- NOTA: l'ordenació topològica pot ser no única!



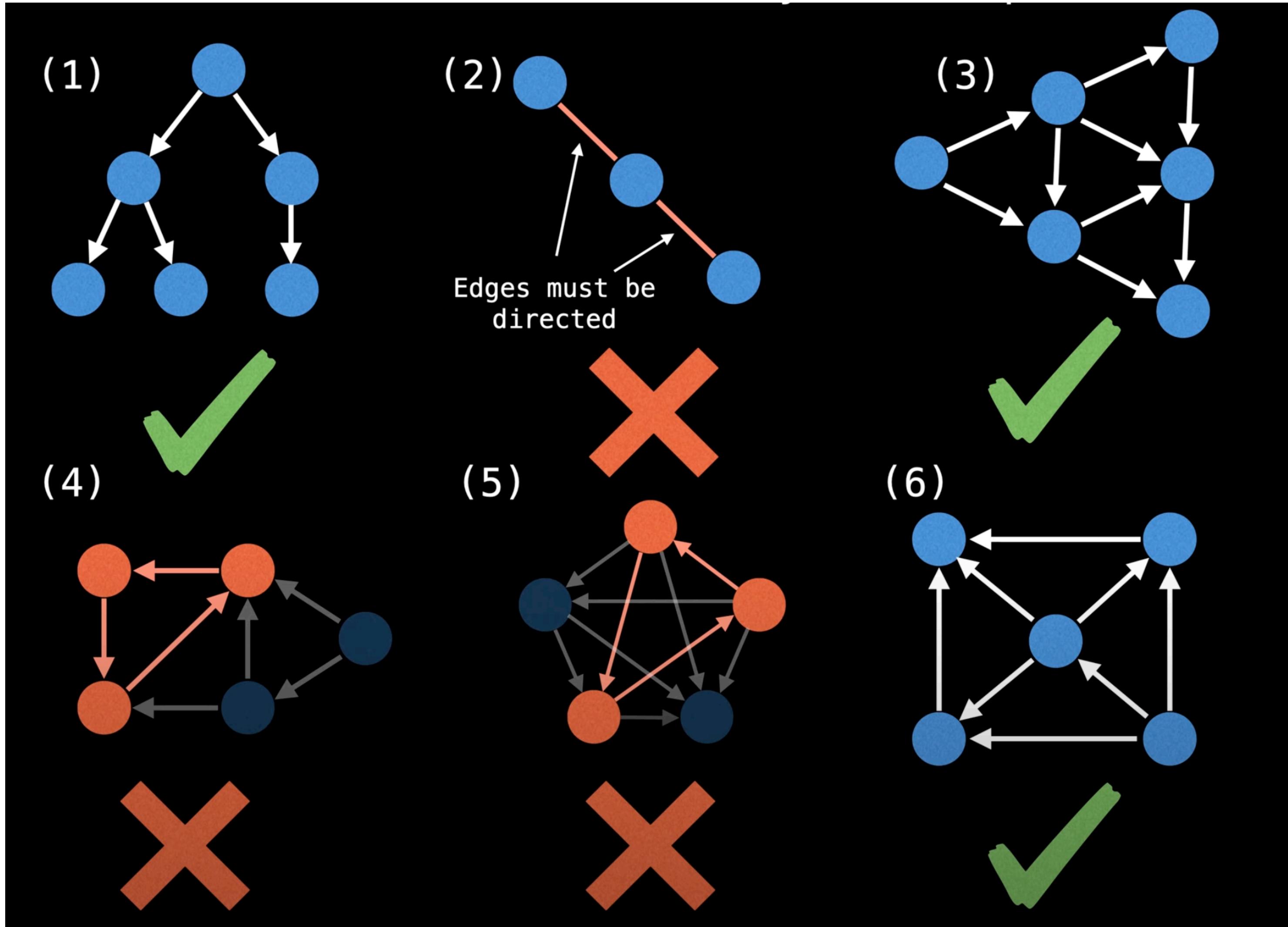
Ordenació topològica

- Només determinats tipus de grafs tenen un ordre topològic. Aquests grafs son anomenats **Grafs Dirigits Acíclics (DAGs)**.
- From wikipedia: “Un graf dirigit acíclic és un **graf dirigit sense cicles**”

Quins dels següents grafs son DAGs?

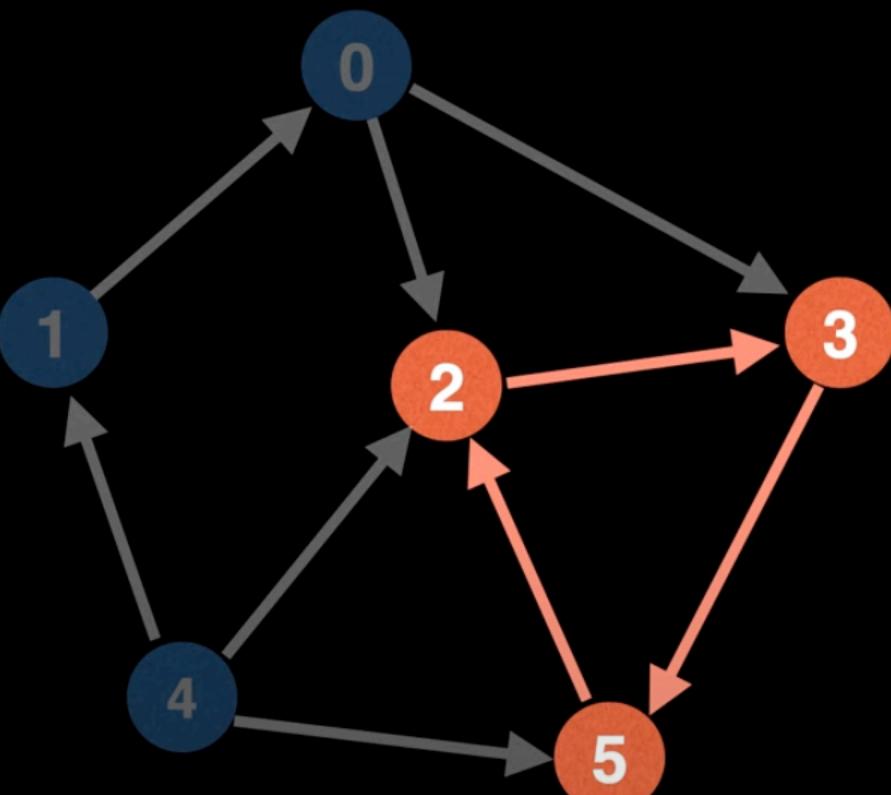


Quins dels següents grafs son DAGs?



Which graphs have topological sorts?

Then you get stuck in a cycle. Every node depends on another node 🤯



4, 1, 0, ???

Algoritme de Kahn

- La intuïció darrere de l'**algorisme de Kahn** és eliminar repetidament els nodes sense cap dependència del graf i afegir-los a l'ordenació topològica.
- Donat que els nodes sense dependència (i les seves de sortida) son eliminats del graf, nous nodes sense cap dependència apareixeran.
- Repetim aquest procés fins que tots els nodes del graf han estat processats, o el graf acíclic hagi estat descobert.

Grafs amb cicles

- Linealitzar usant DFS

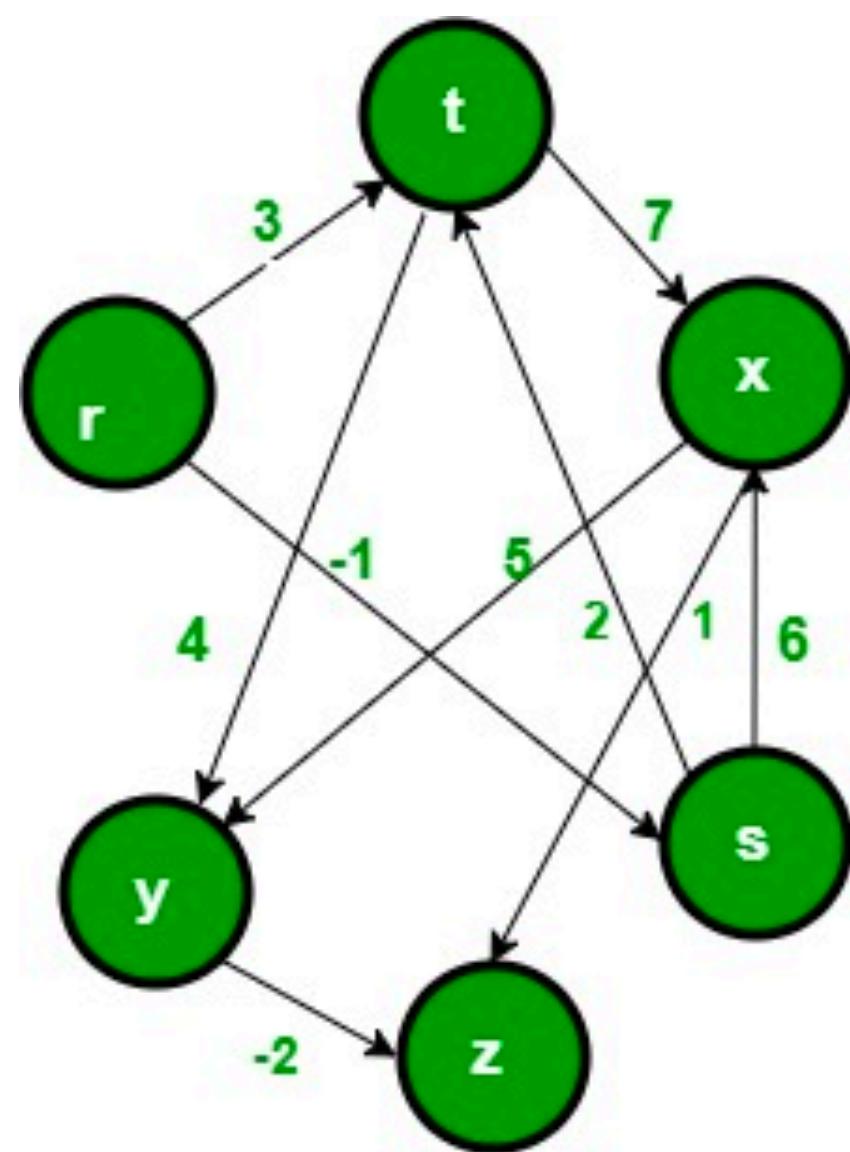
```
procedure dag-shortest-paths( $G, l, s$ )
Input:    Dag  $G = (V, E)$ ;
          edge lengths  $\{l_e : e \in E\}$ ; vertex  $s \in V$ 
Output:   For all vertices  $u$  reachable from  $s$ ,  $\text{dist}(u)$  is set
          to the distance from  $s$  to  $u$ .

for all  $u \in V$ :
   $\text{dist}(u) = \infty$ 
   $\text{prev}(u) = \text{nil}$ 

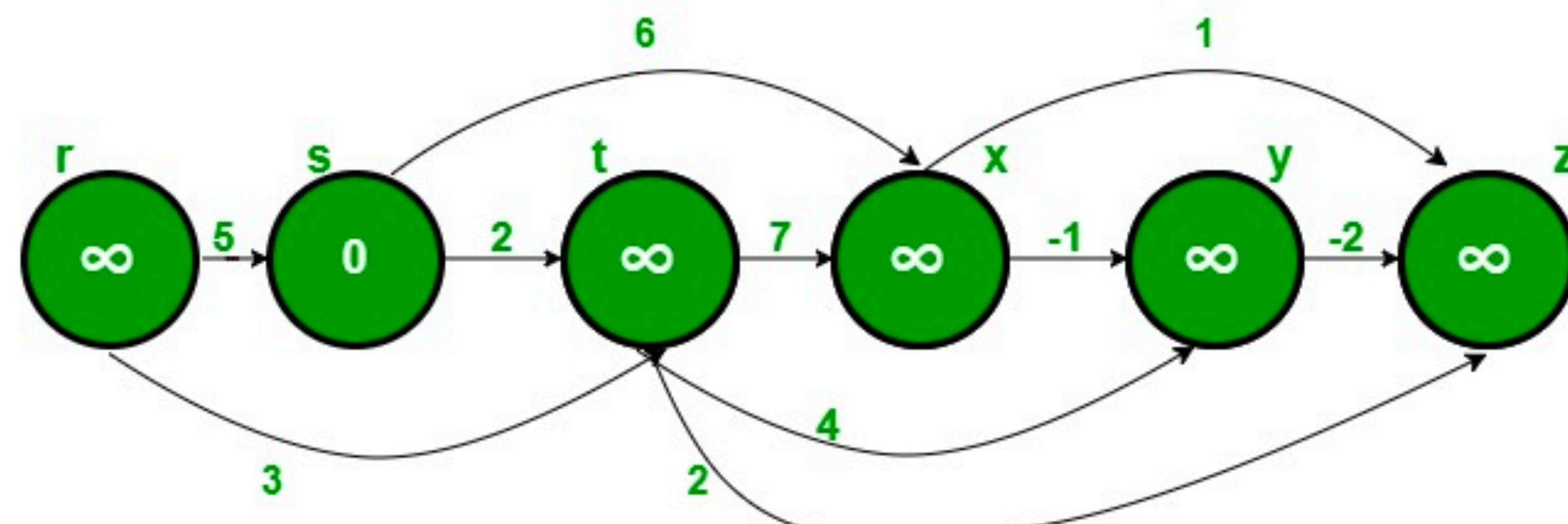
 $\text{dist}(s) = 0$ 
Linearize  $G$ 
for each  $u \in V$ , in linearized order:
  for all edges  $(u, v) \in E$ :
    update  $(u, v)$ 
```

Graf amb cicles

- Pas 1: linearitzar el graf (topological SORT amb DFS)

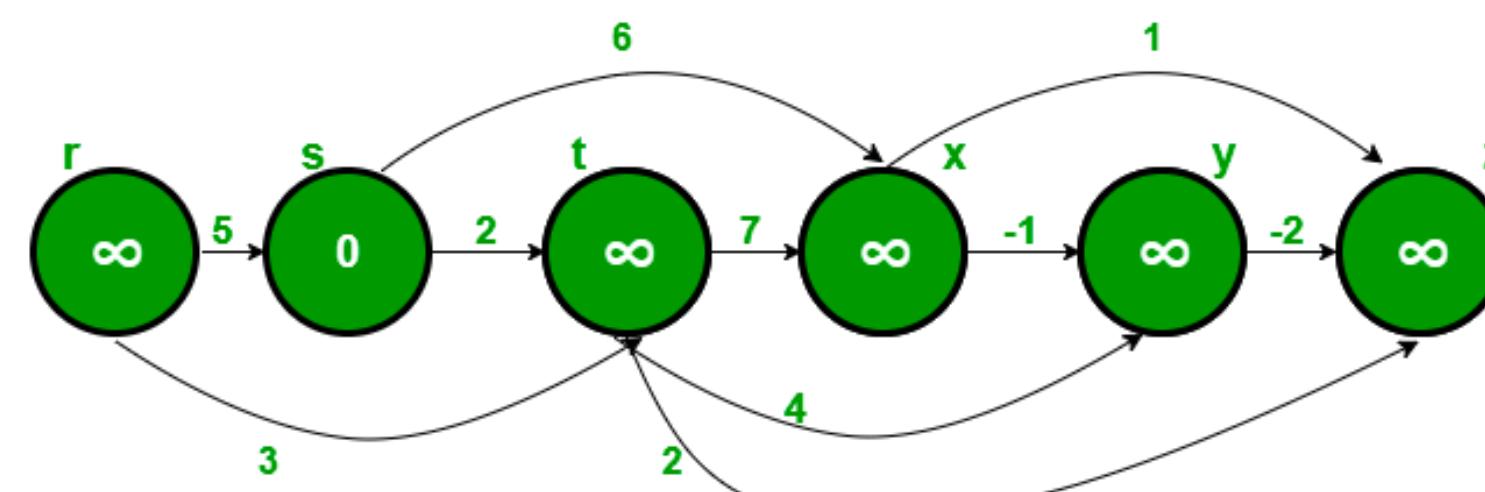


(a)

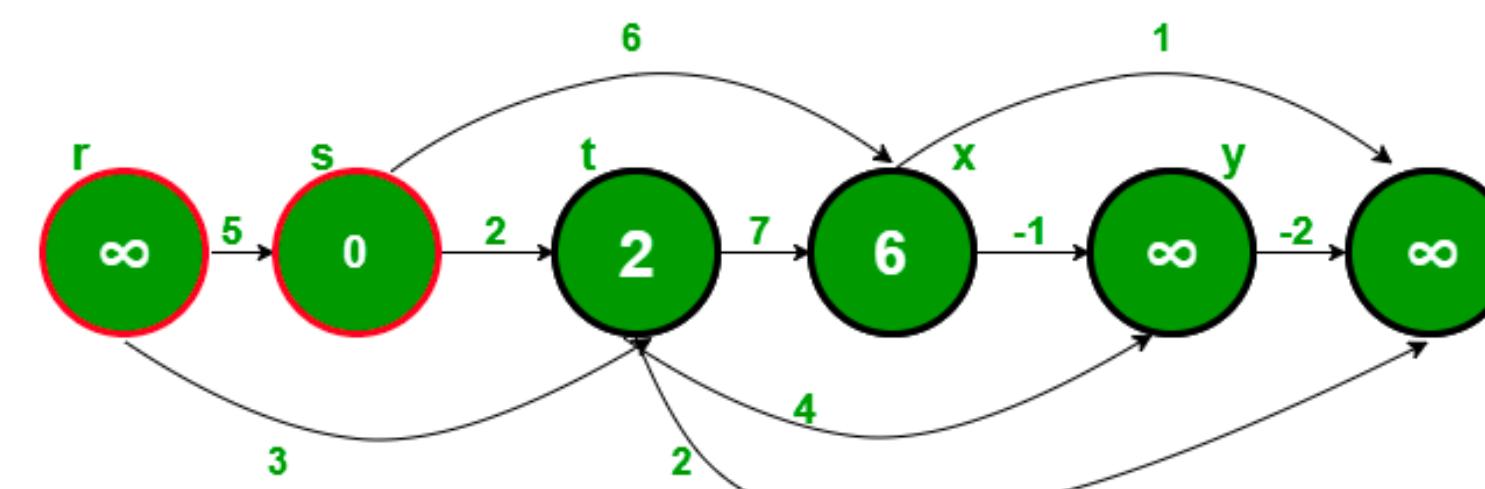


(b)

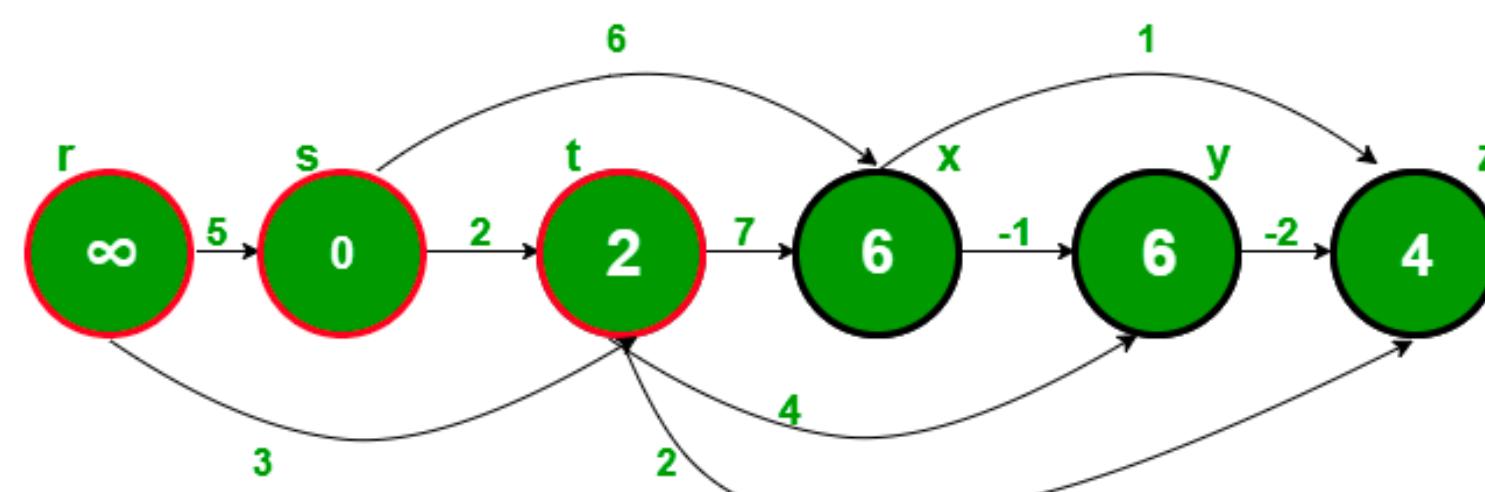
Graf amb cicles



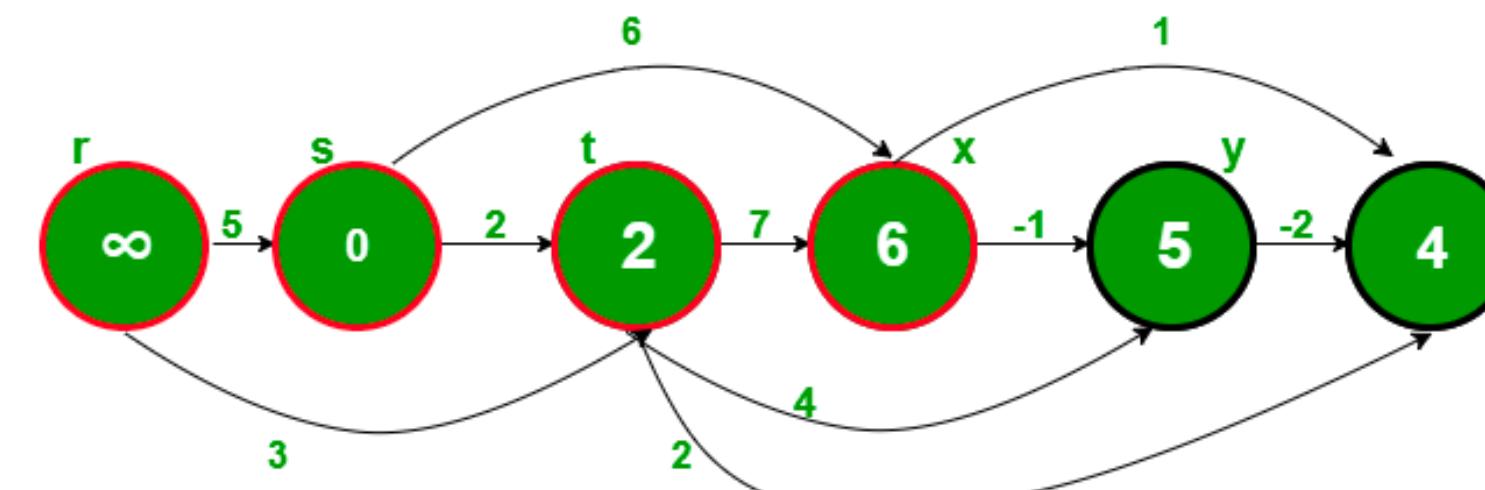
(c)



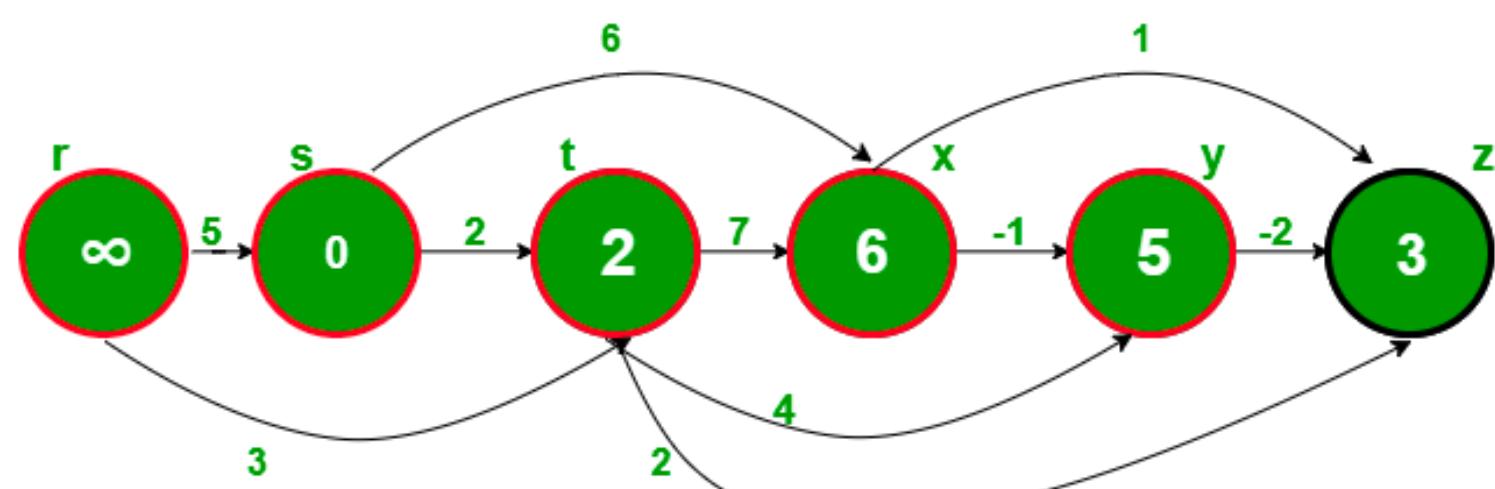
(d)



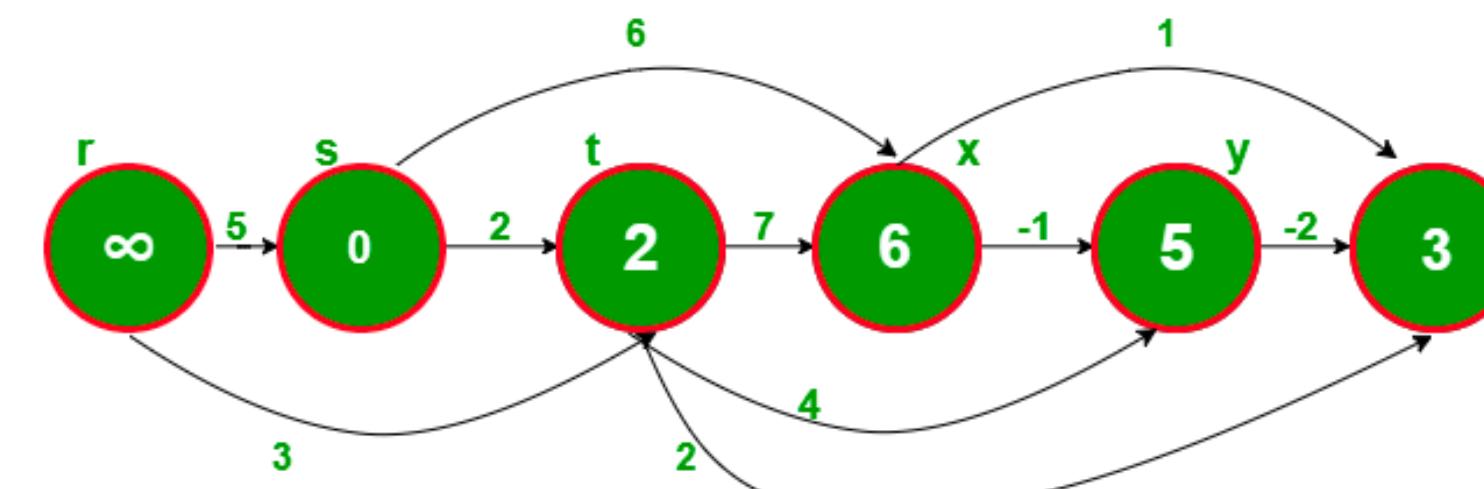
(e)



(f)

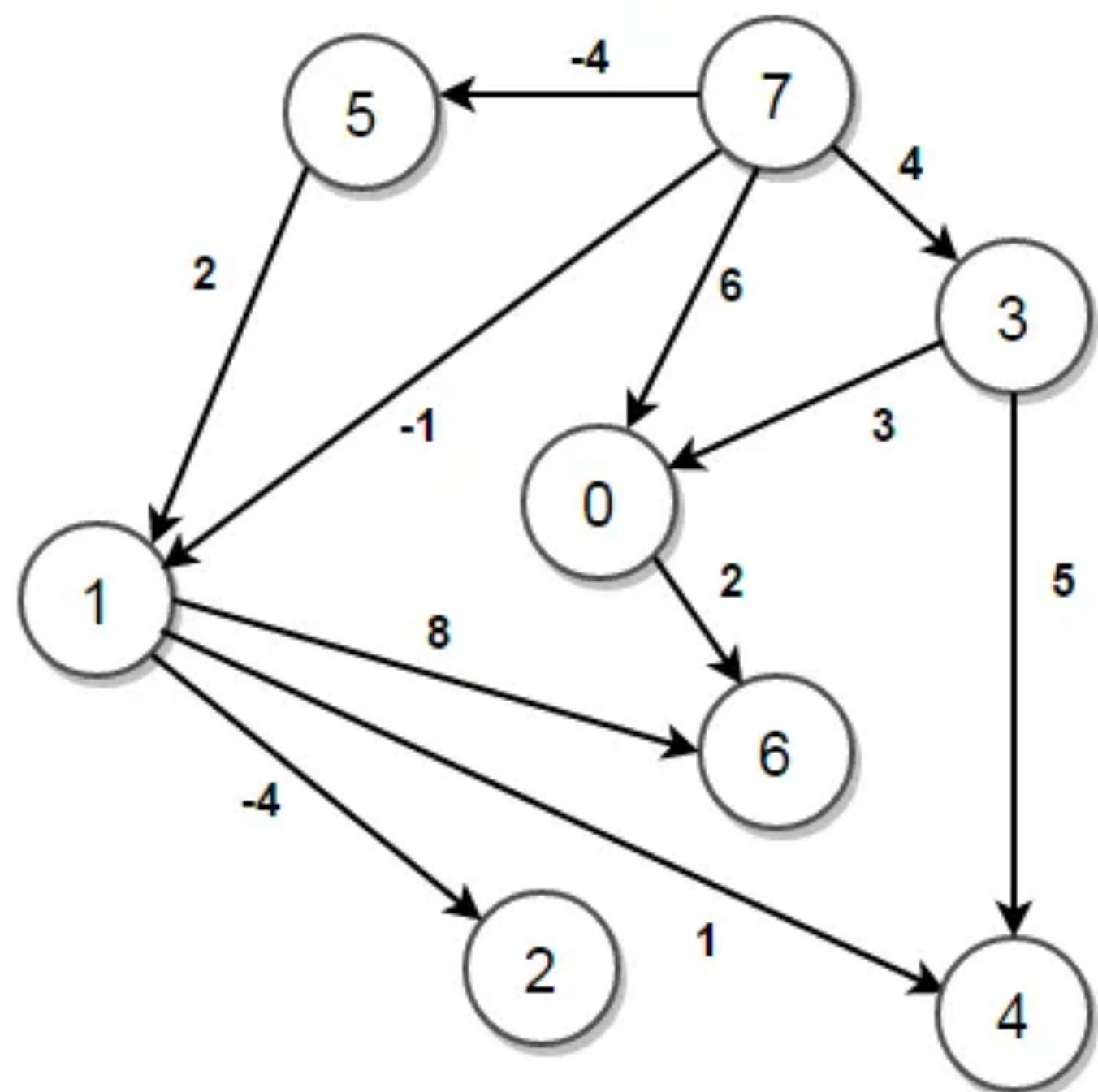


(g)

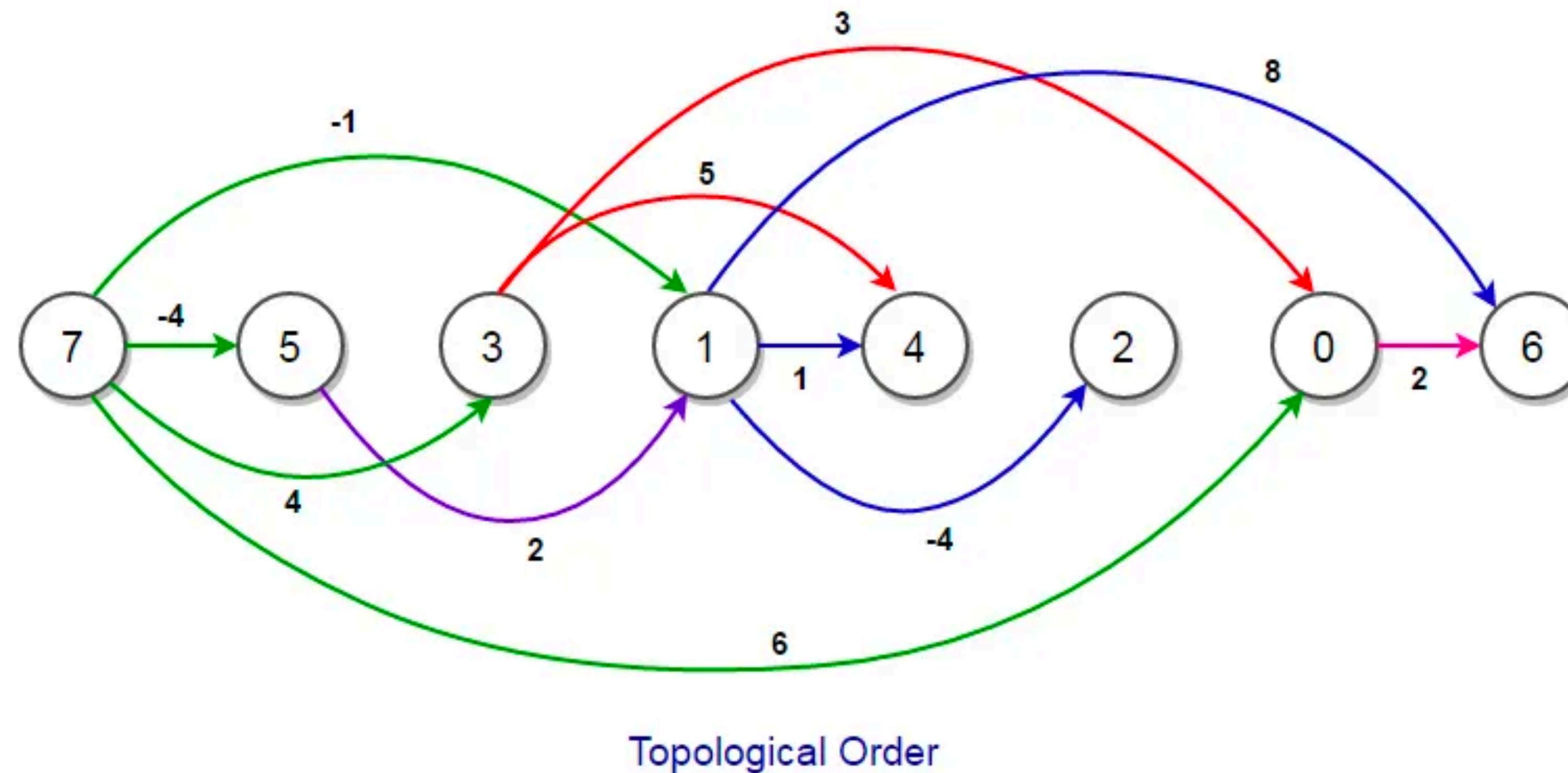


(h)

Start



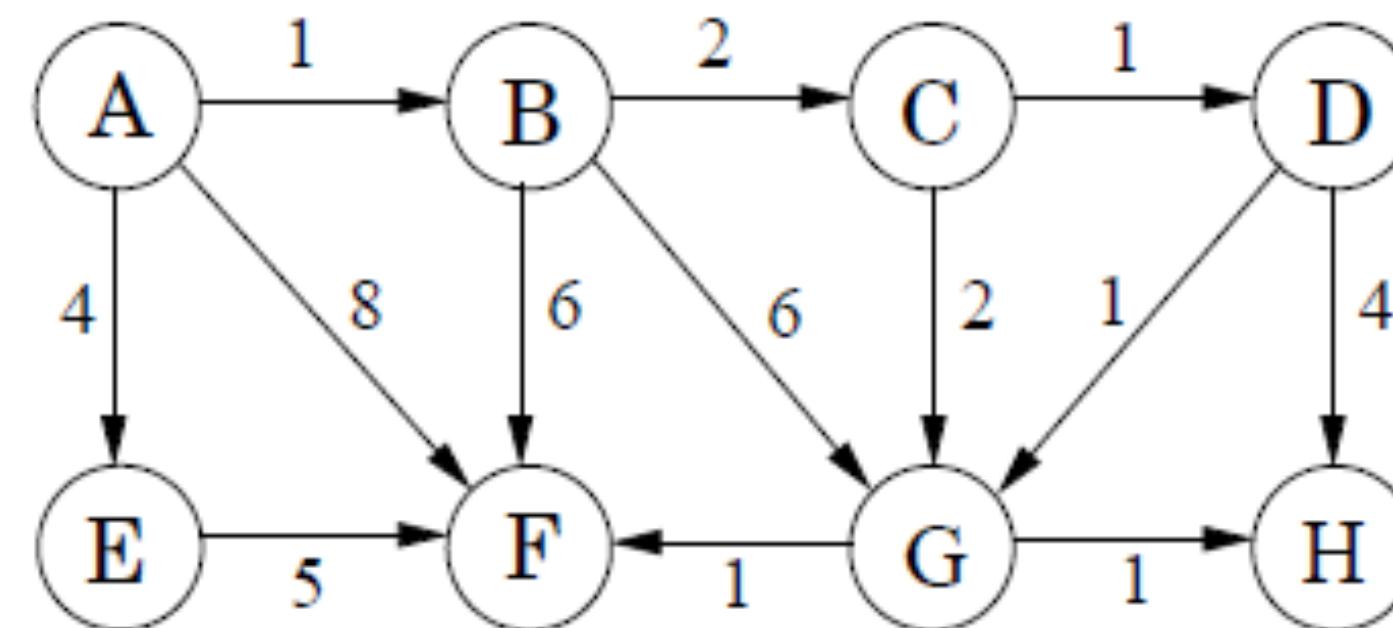
Topological order



Kahn's Algorithm

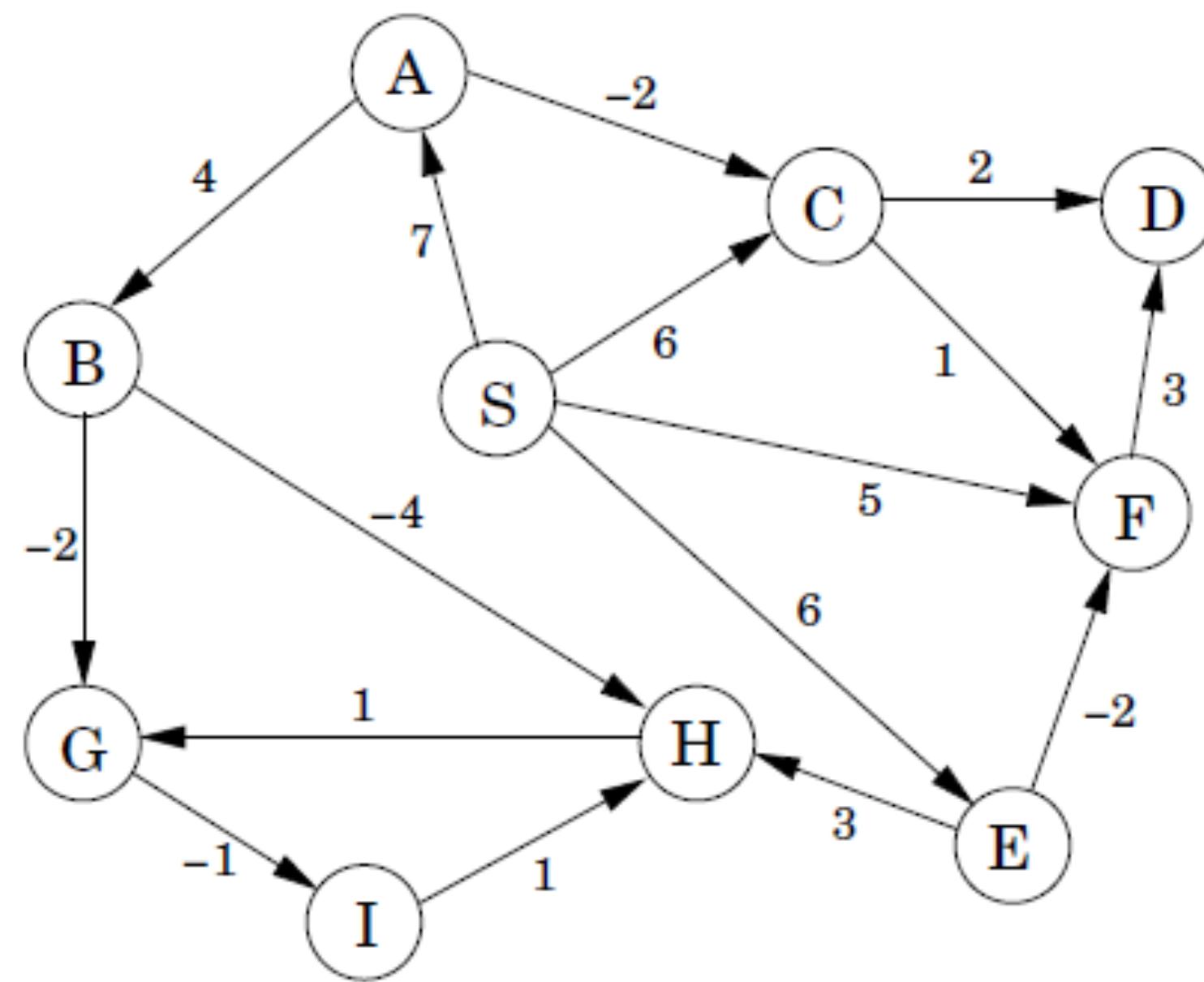
Exercici: Dijkstra

- Començant al node A:
 - dibuixa la taula de distàncies immediates a tots els nodes per a cada iteració del mètode;
 - mostra l'arbre de camins mínims.



Exercici: Bellman-Ford

- Començant al node A: dibuixa la taula de distàncies immediates a tots els nodes per a cada iteració del mètode.



Fins ara hem vist com calcular el camí més curt des d'un node **u** donat.

Com calculem el camí més curts de tots el nodes?