- Representació
- Recorreguts
  - Recorregut en profunditat
  - Recorregut en amplada
- Ordenació topològica
- Problemes

- Recordem que un graf és una col·lecció de nodes (vèrtexos) en què diversos parells de vèrtexos són connectats per un arc.
- Els grafs són més generals que els arbres.
  - En un arbre, només hi ha un camí cap a tot altre node. És per això que no hi pot haver cicles en un arbre.
  - En un graf, no hi ha cap noció de jerarquia entre vèrtexos (tot vèrtex pot estar connectat amb tot altre vèrtex, fins i tot amb sí mateix) mentre que en un arbre, és impossible que un node estigui connectat amb cap altre node que no sigui el seu pare.
  - Un graf consisteix en un o més components connexos (un component connex és una porció d'un graf en què hi ha al menys un camí entre tot parell de vèrtexos) mentre que en un arbre, hi ha exactament un component connex.

- Atès que els grafs són més generals que els arbres, es poden usar per modelitzar més classes de problemes.
  - Ordenació topològica, per a *scheduling* i anàlisi de camí crític.
  - Grafs reduïts a arbres d'expansió mínims, per a problemes d'encaminament de xarxes.
  - Camins mínims, per a problemes de xarxes.
  - Connectivitat, per a estudis de fiabilitat de xarxes.
- Aquesta generalització, però, té un cost.
  - Els algorismes sobre grafs són més complexos que els algorismes sobre arbres. Els algorismes de búsqueda i recorregut han de contemplar la possibilitat que els vèrtexos que estan essent visitats ja hagin estat visitats a través d'un altre camí, mentre que en un arbre, només es pot accedir a un node a través d'un únic camí.

- Els grafs es poden veure com a tipus abstracte de dades.
- Els gèneres són:
  - un conjunt no buit de vèrtexos
  - un conjunt d'arcs (parells ordenats de vèrtexos)
- Les operacions constructores són:
  - crea : $\longrightarrow$ graf
  - insereix_vèrtex : graf vèrtex $\longrightarrow$ graf
  - insereix_arc : graf vèrtex vèrtex $\longrightarrow$ graf
- Algunes de les operacions consultores són:
  - ordre, talla : graf $\longrightarrow$ int
  - grau : graf vèrtex $\longrightarrow$ int
  - origen, destinació : graf arc $\longrightarrow$ vèrtex
  - oposat : graf arc vèrtex $\longrightarrow$ vèrtex
  - buit : graf $\longrightarrow$ bool
  - vèrtexos : graf $\longrightarrow$ list(vèrtex)
  - arcs : graf $\longrightarrow$ list(arc)

- El tipus abstracte de dades dels grafs se sol estendre amb *iteradors* sobre els vèrtexos i els arcs del graf, els vèrtexos adjacents amb un vèrtex del graf, etc.

– Iterador sobre els vèrtexos del graf

for all vèrtex v de G **do**

...

– Iterador sobre els arcs del graf

for all arc (v, w) de G **do**

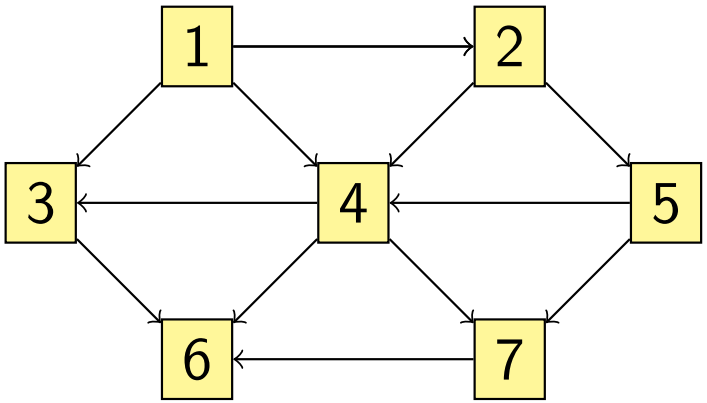
...

– Iterador sobre els vèrtexos adjacents amb un vèrtex del graf

for all vèrtex w adjacent amb el vèrtex v **do**

...

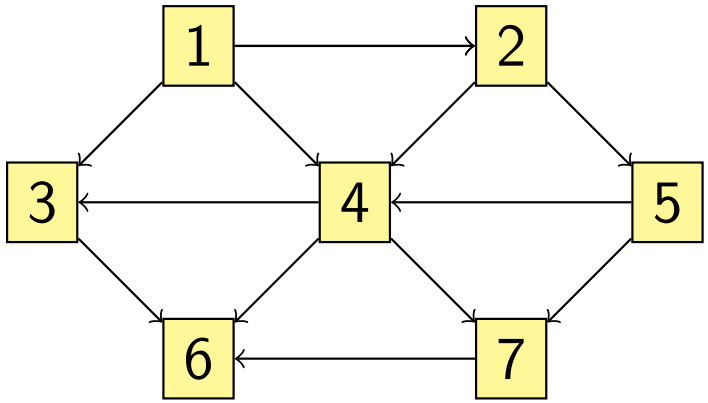
- Estructures de dades més usuals per representar grafs:
  - Matrius d'adjacència



$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

  - $A : n \times n$
  - $A_{ij} \equiv (v_i, v_j)$ és un arc del graf
  - A_{ij} representa el pes de l'arc (v_i, v_j)
  - Cost espacial $\Theta(n^2)$

- Estructures de dades més usuals per representar grafs:
  - Llistes d'adjacència



$$
\begin{array}{lcl}
1 & : & [2,3,4] \\
2 & : & [4,5] \\
3 & : & [6] \\
4 & : & [3,6,7] \\
5 & : & [4,7] \\
6 & : & [\,] \\
7 & : & [6]
\end{array}
$$

  - Cost espacial $\Theta(n+m)$

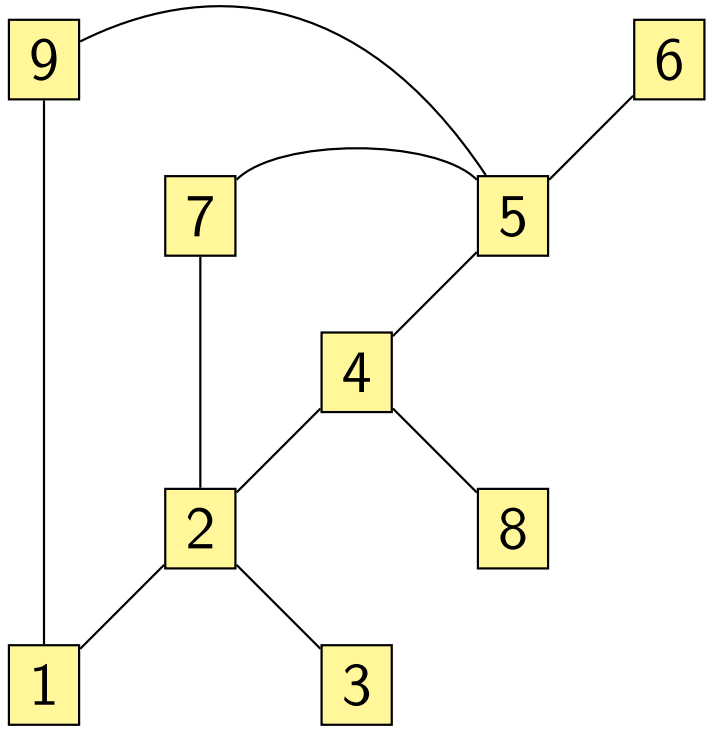
- **function** grau (G: graf; v: vèrtex)
  int deg := 0
  for all vèrtex w adjacent amb el vèrtex v **do**
    deg := deg + 1
  return deg

- ```
  int graph::outdegree( int i ) const
  {
      int deg = 0;
      for ( int j = 0;j < ordre();j++ )
          if ( ( *ar ) ( i, j ) )
              deg++;
      return deg;
  }
  ```
  ```

- A partir d'un vèrtex inicial s en un graf G, enumerar tots els altres vèrtexos de G accessibles des de s a través de camins de G.
- **procedure** recorregut (G: graf; s: vèrtex)
  for all vèrtex v de G **do**
    visitat[v] := false
  visitar el vèrtex s
  visitat[s] := true
  for all vèrtex v de G accessible des de s **do**
    if ¬visitat[v] **then**
      visitar el vèrtex v
      visitat[v] := true

- A partir d'un vèrtex inicial s en un graf G, enumerar tots els altres vèrtexos de G accessibles des de s a través de camins de G.
- **procedure** recorregut (G: graf; s: vèrtex)
  for all vèrtex v de G **do**
  visitat[v] := false
  inserir s en contenidor C
  repeat
  estreure qualque vèrtex v del contenidor C
  if ¬visitat[v] **then**
  visitar el vèrtex v
  visitat[v] := true
  for all vèrtex w adjacent amb el vèrtex v **do**
  if ¬visitat[w] **then**
  inserir w en el contenidor C
  until el contenidor C sigui buit

- Sigui G el graf següent:

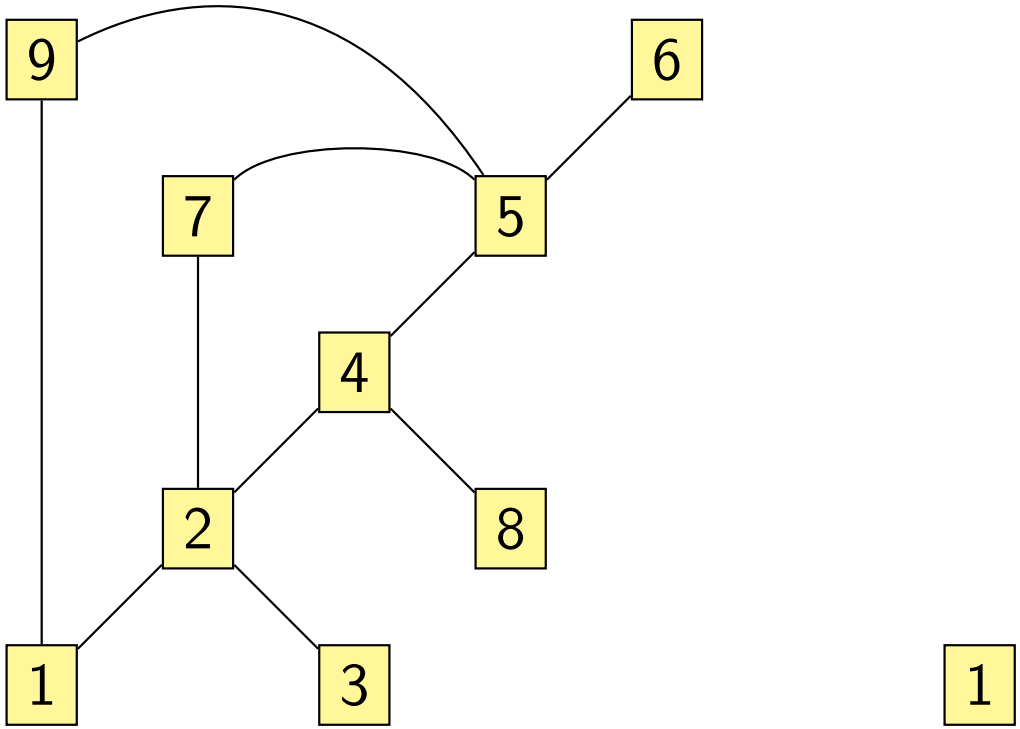


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

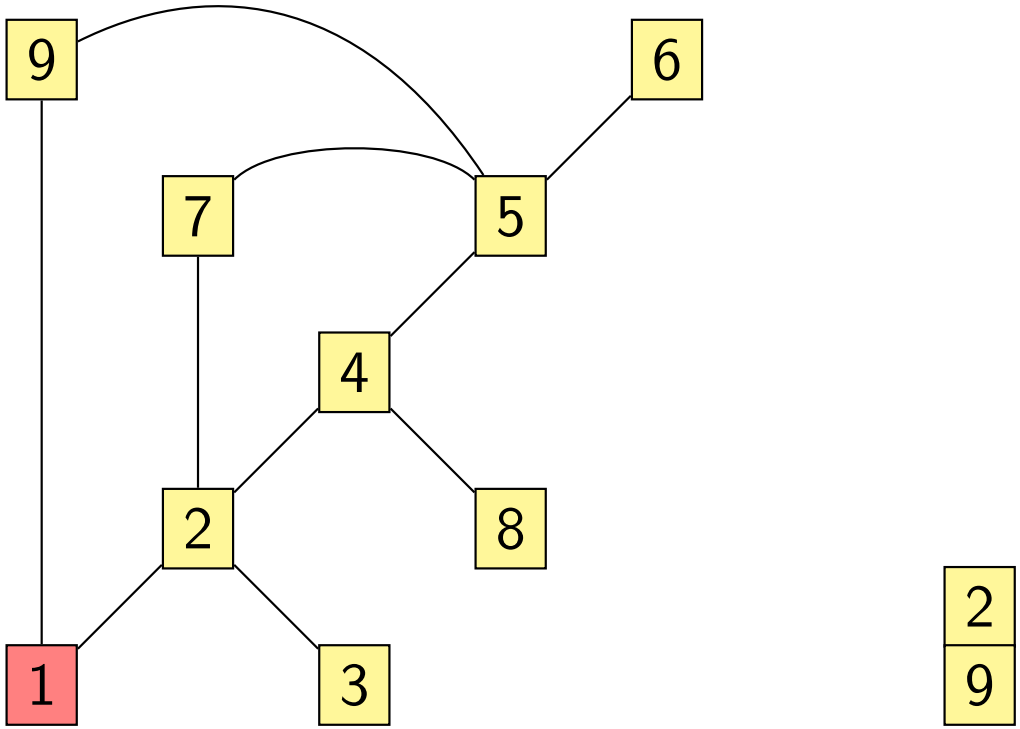


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

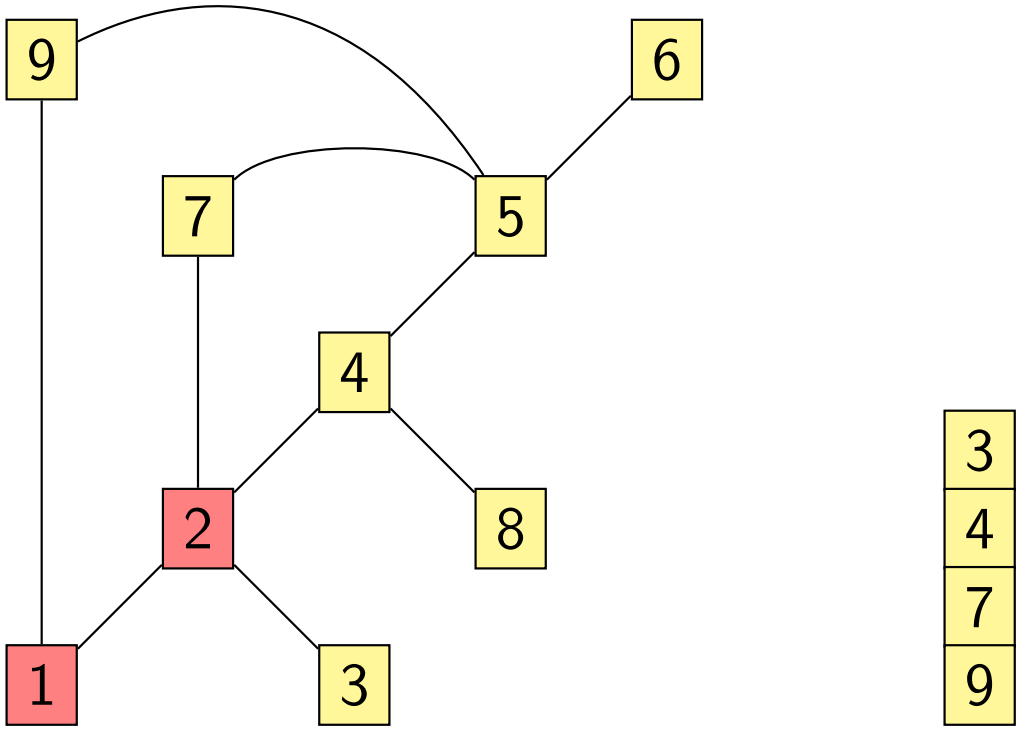


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

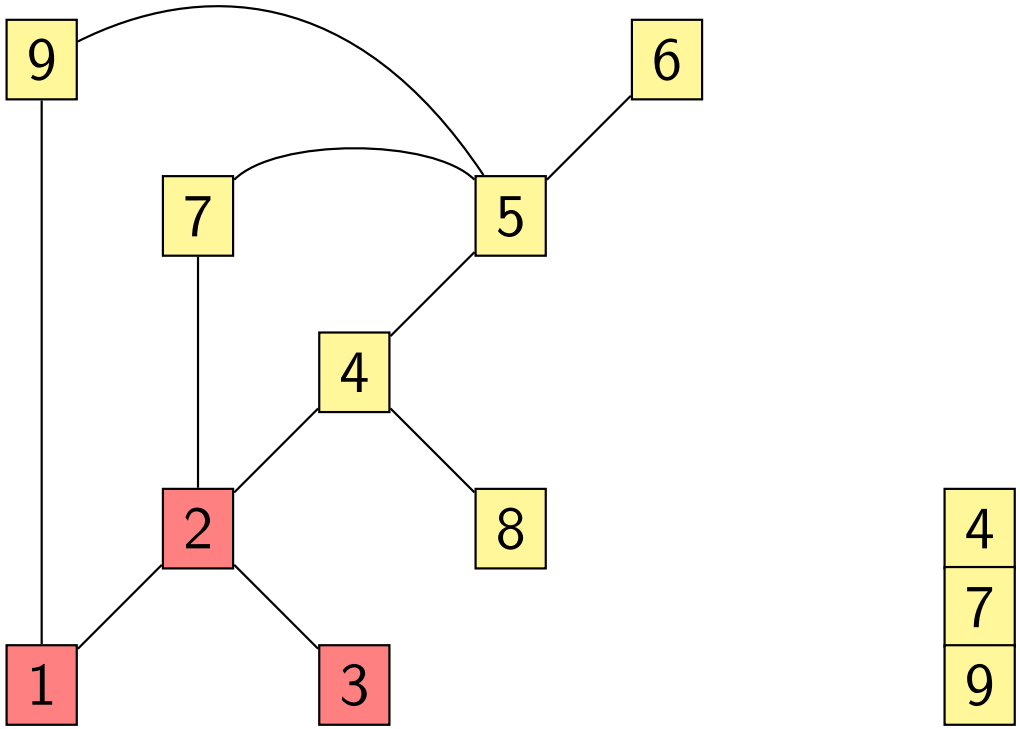


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

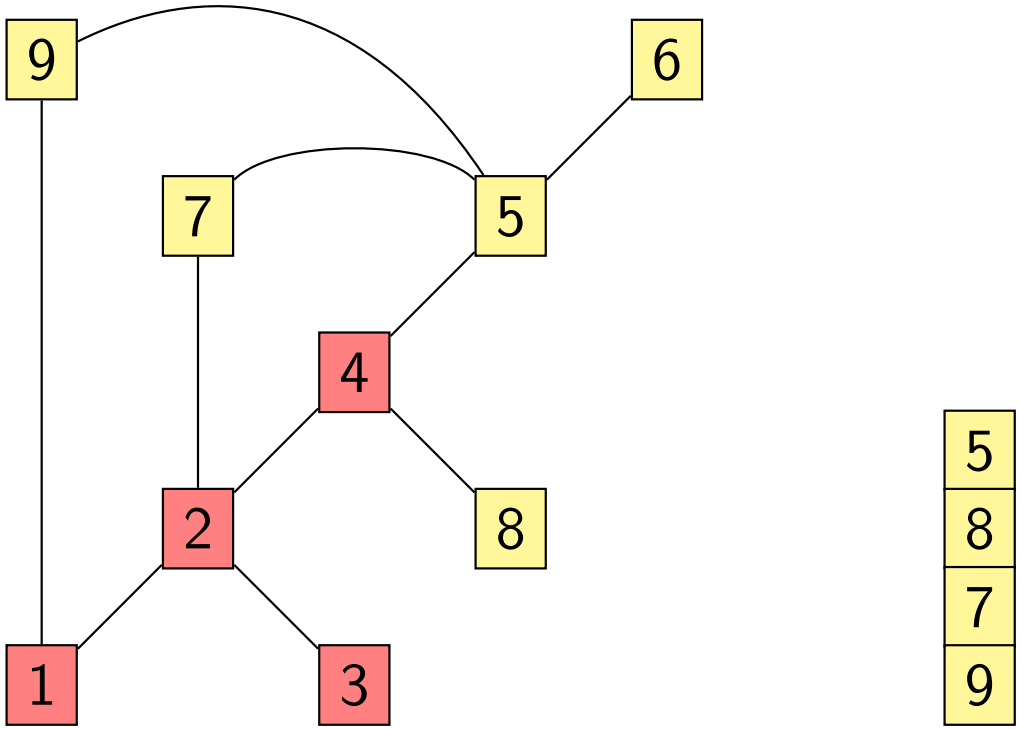


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

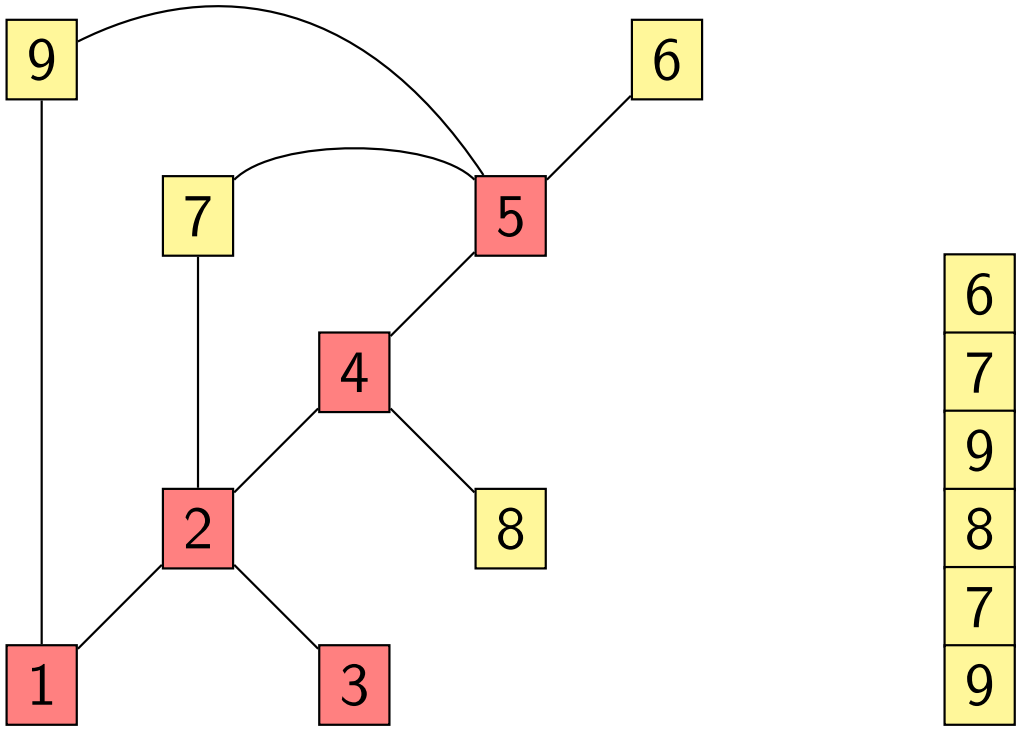


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

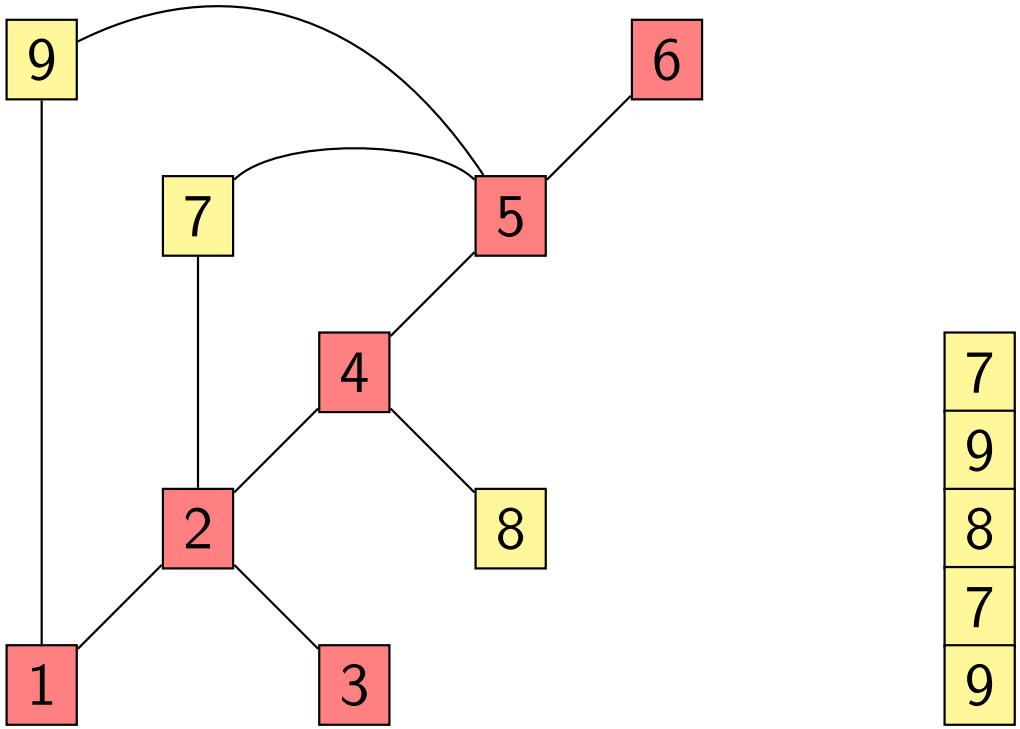


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

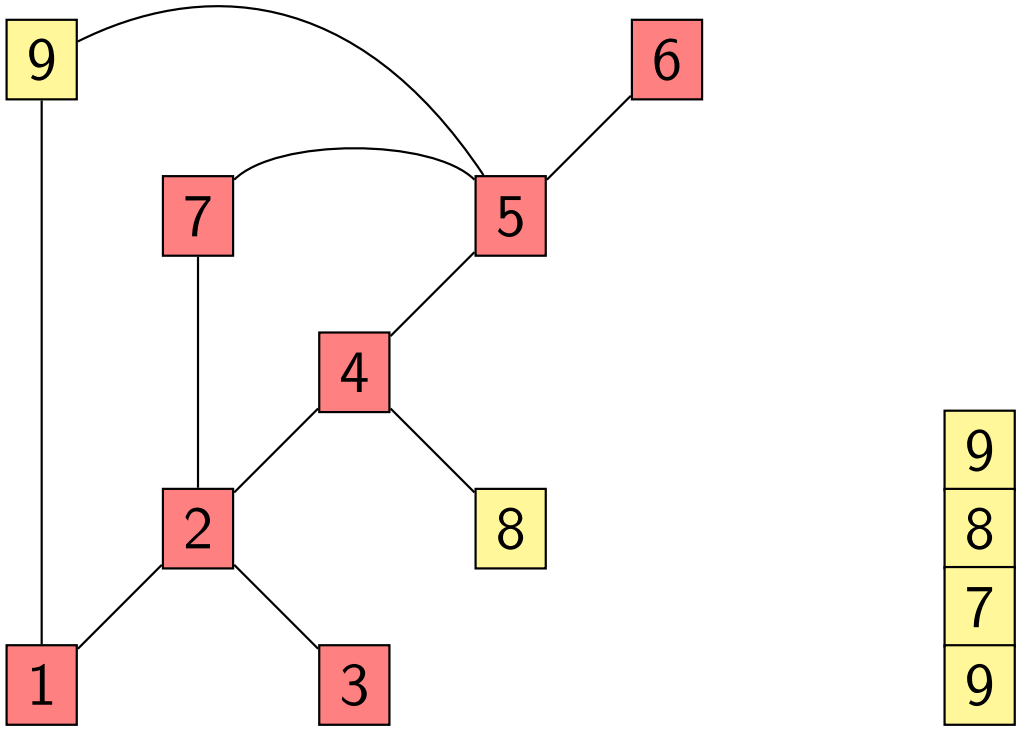


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

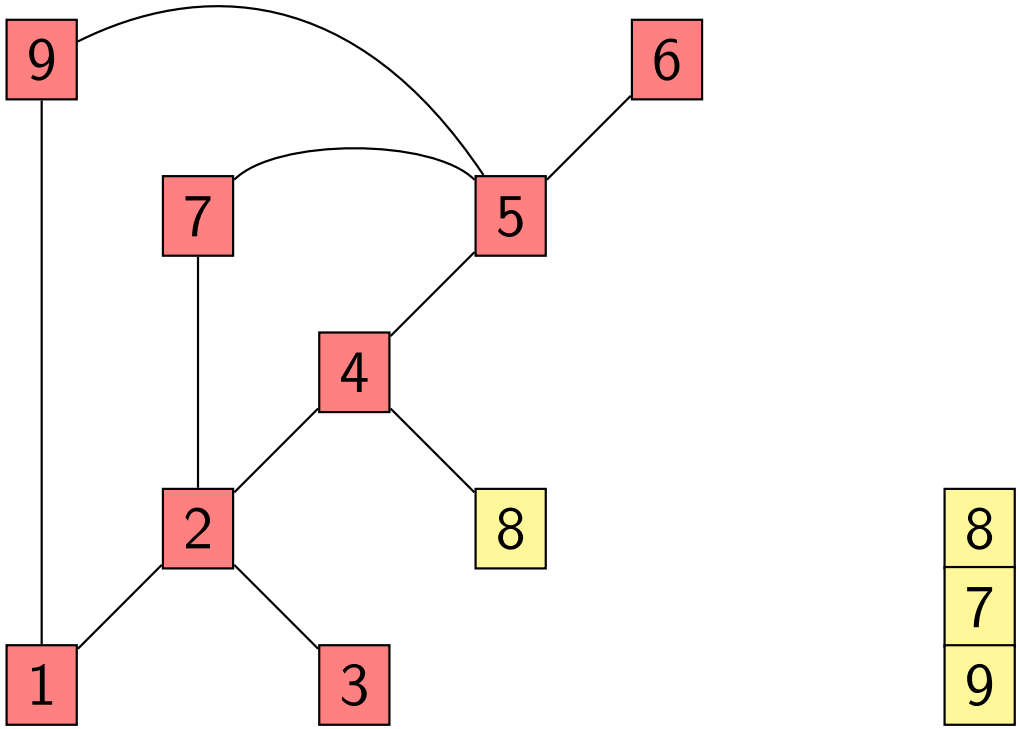


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

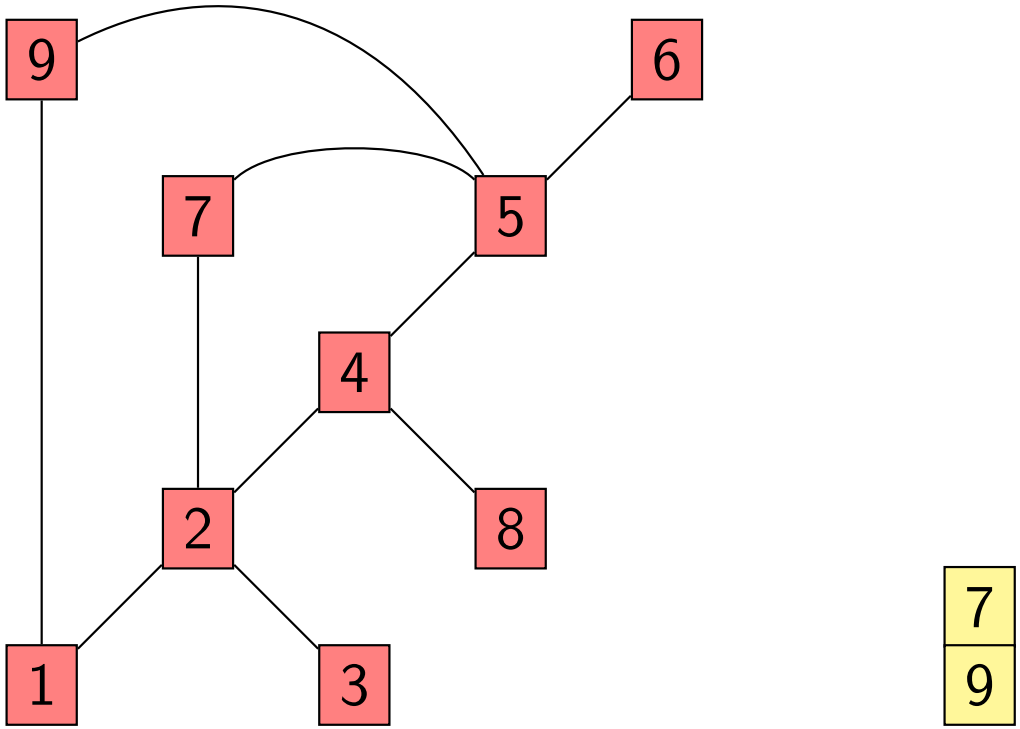


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

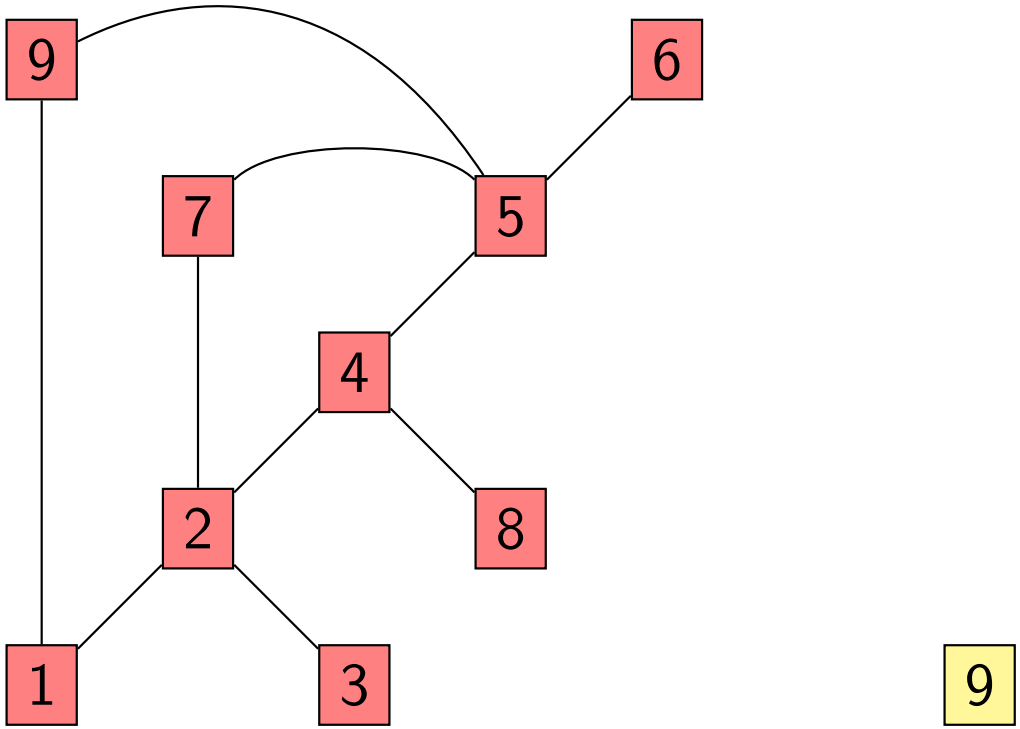


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

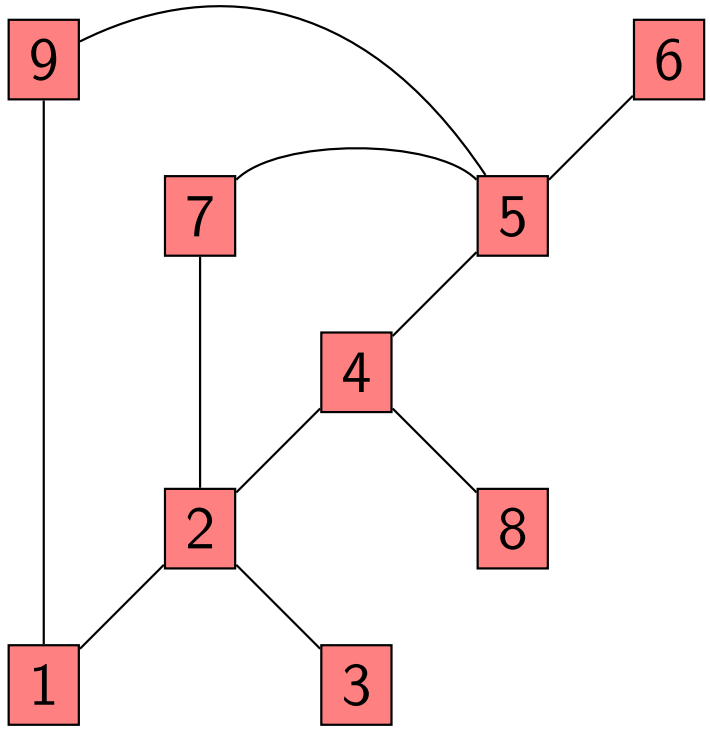


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

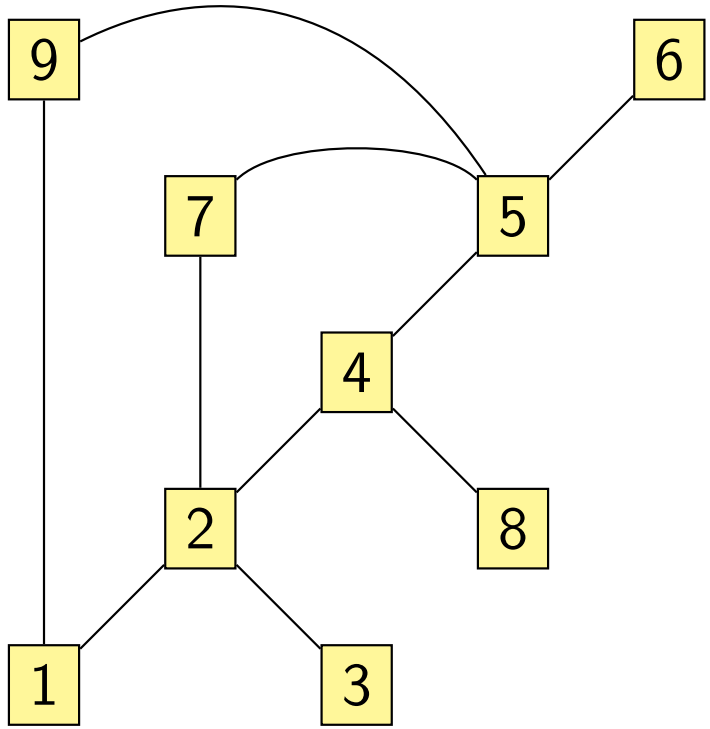


- Els veïns del vèrtex que està essent visitat es posen en una pila.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 3, 4, 5, 6, 7, 9, 8$$

- És una generalització del recorregut d'un arbre en preordre.

- Sigui G el graf següent:

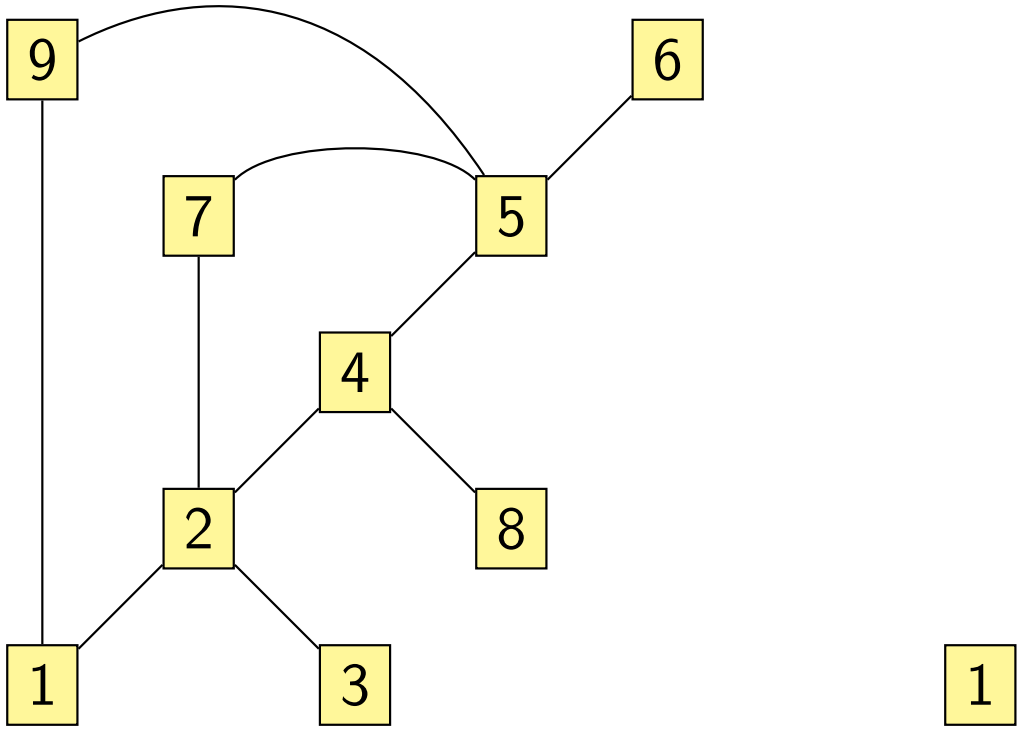


- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:

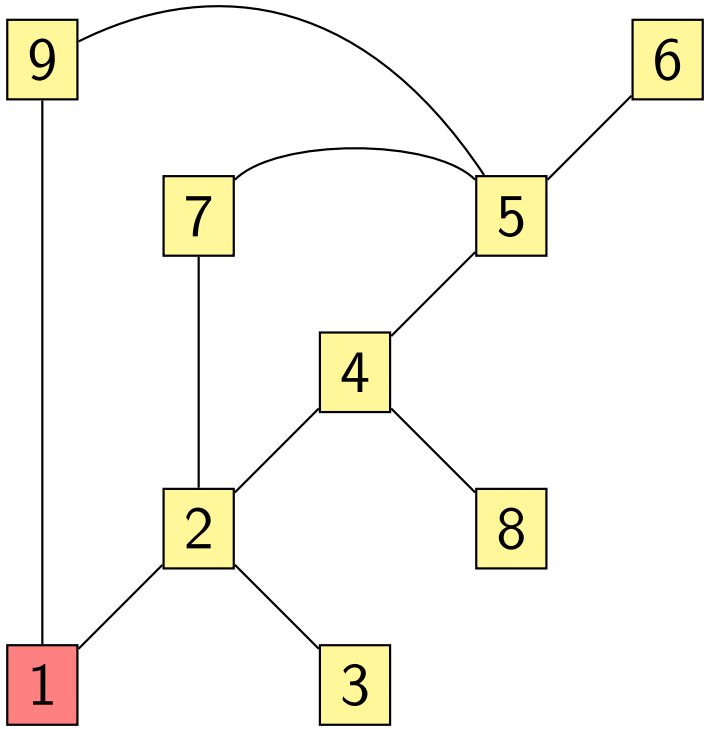


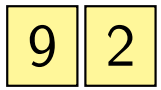
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



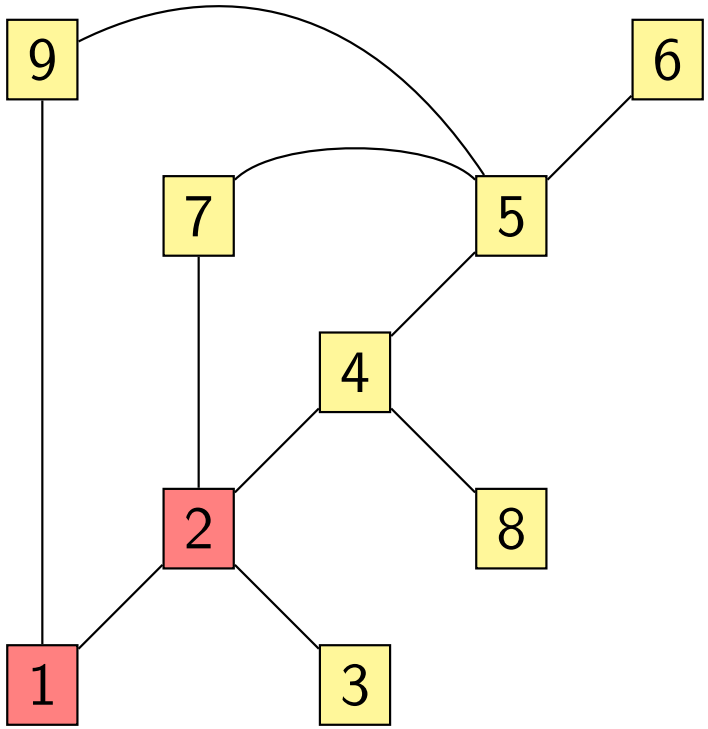


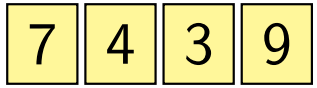
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



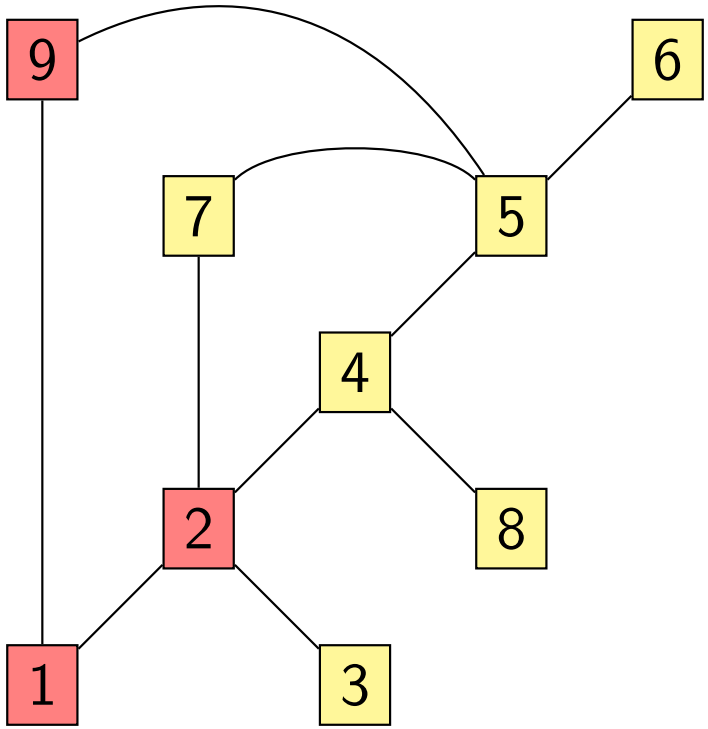


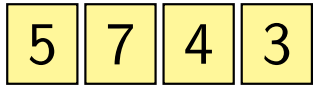
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



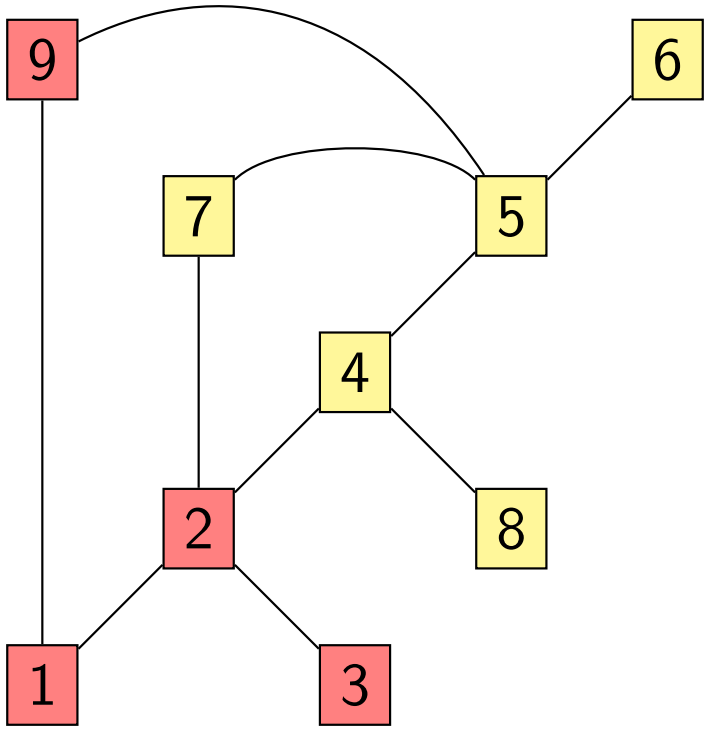


- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:

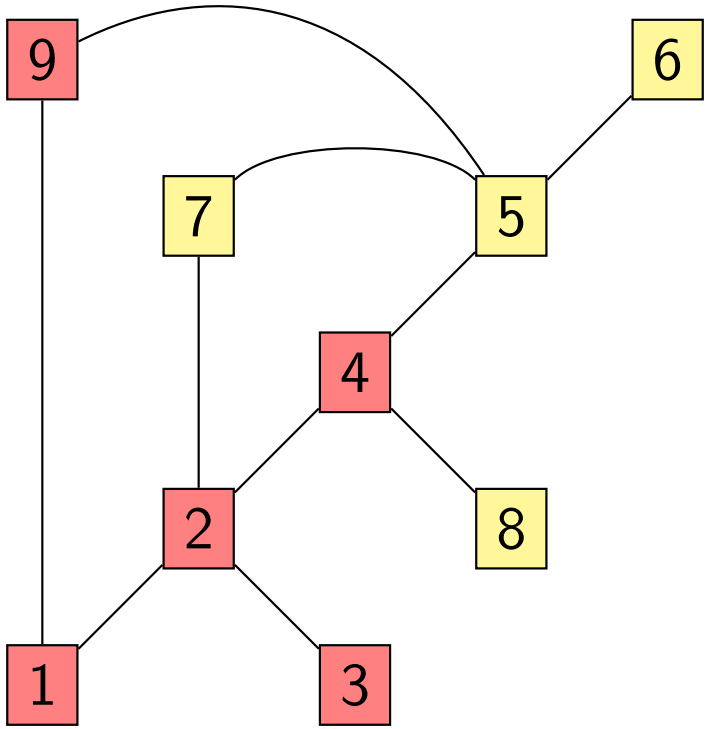


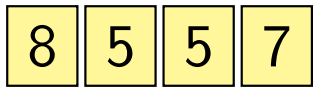
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



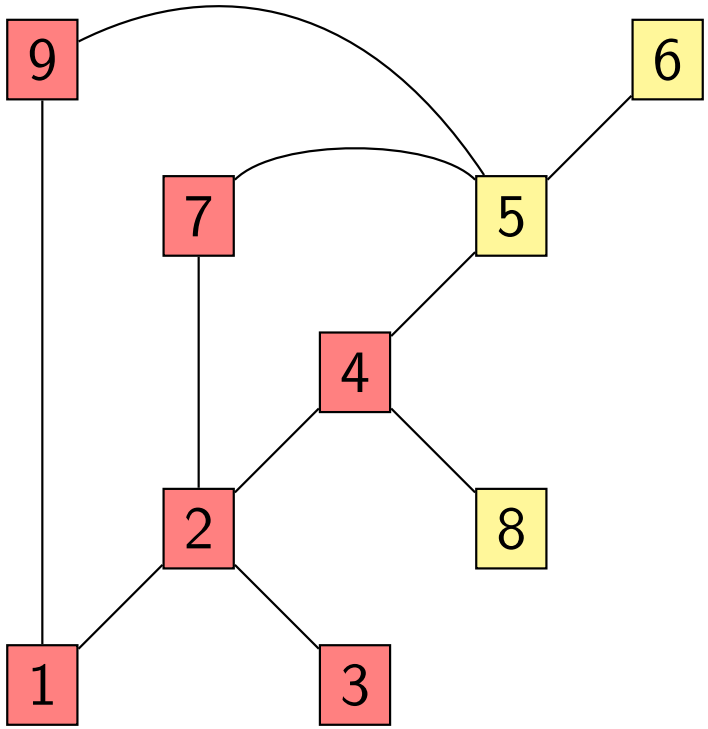


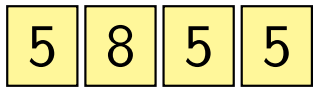
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



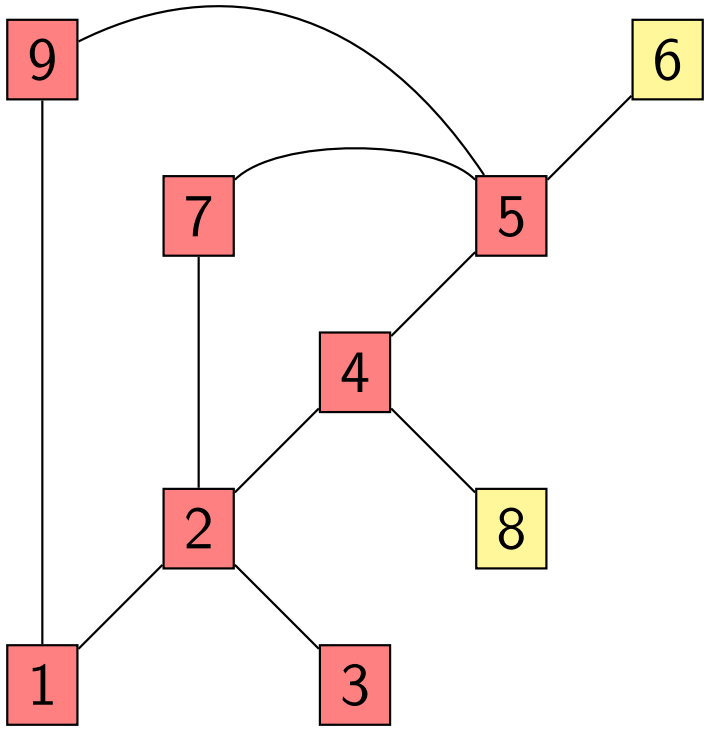


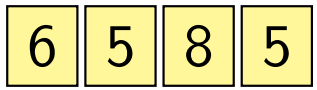
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



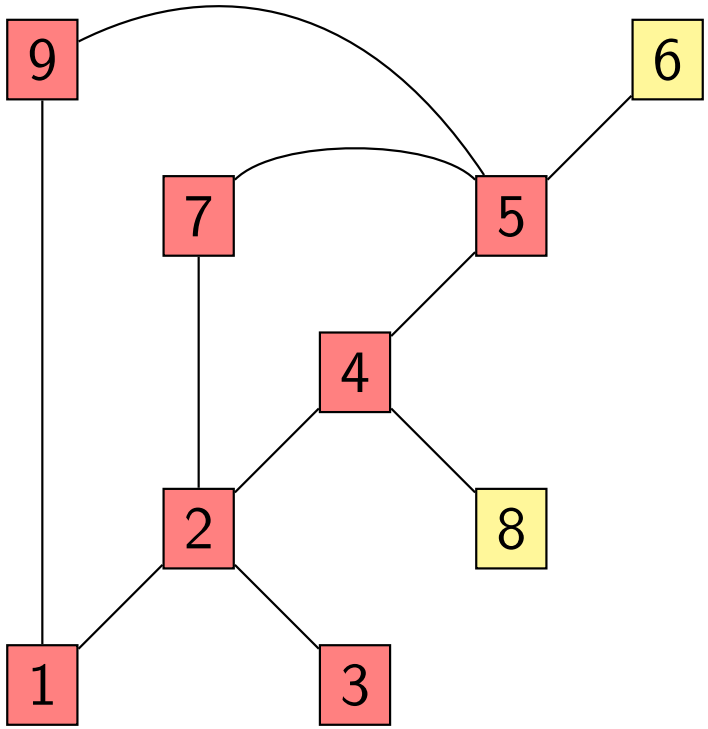


- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:

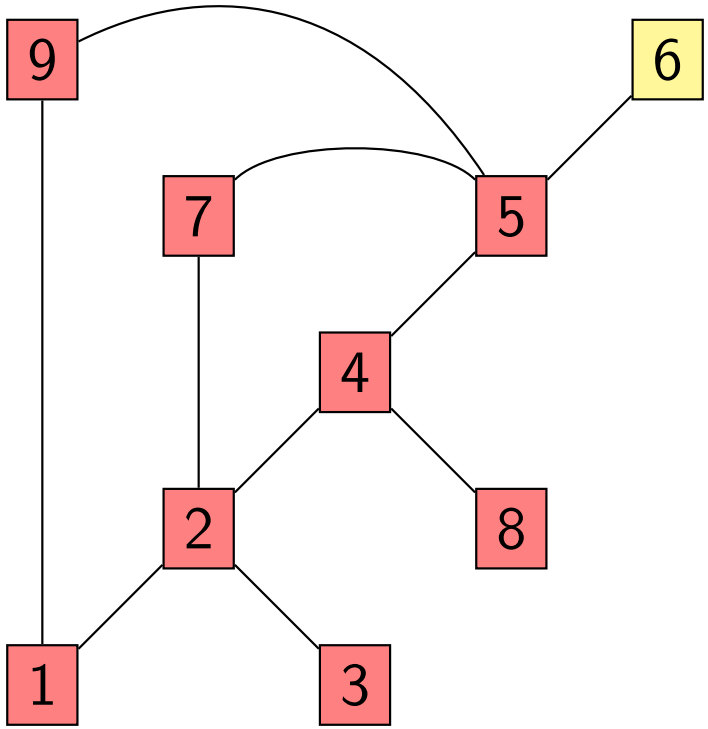


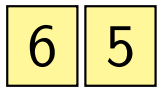
- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:



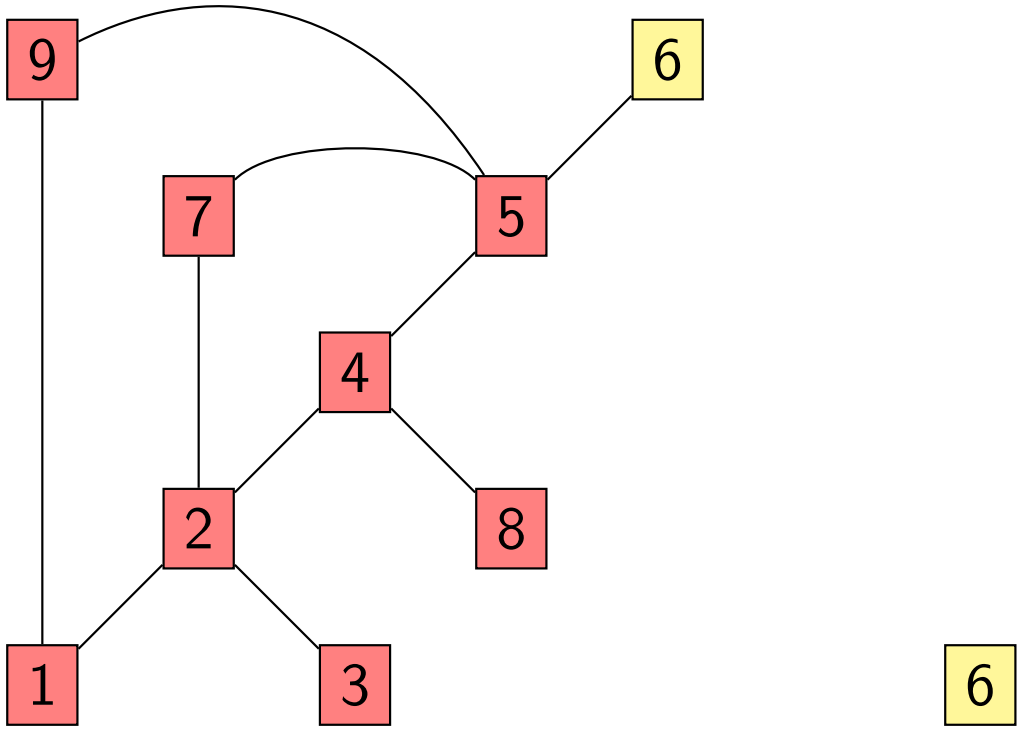


- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:

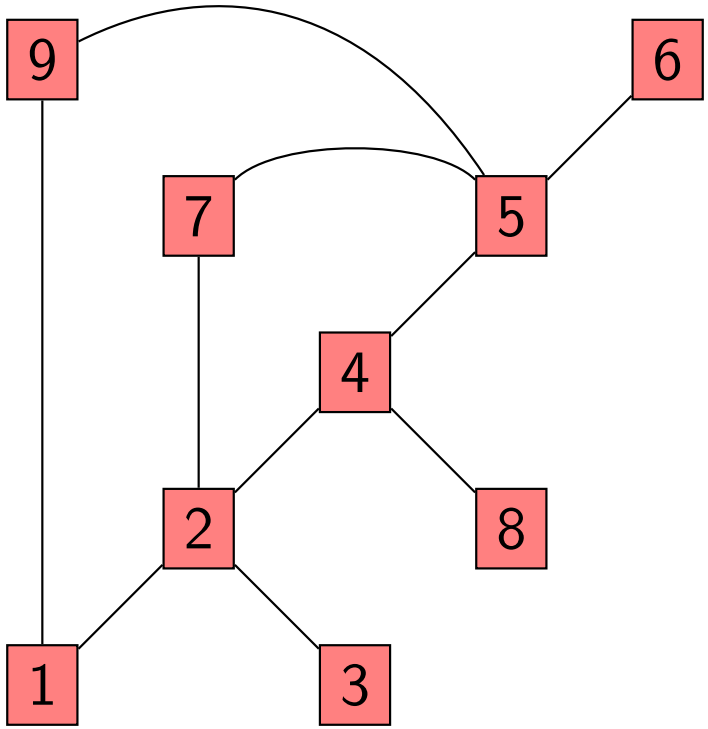


- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

- És una generalització del recorregut d'un arbre per nivells.

- Sigui G el graf següent:

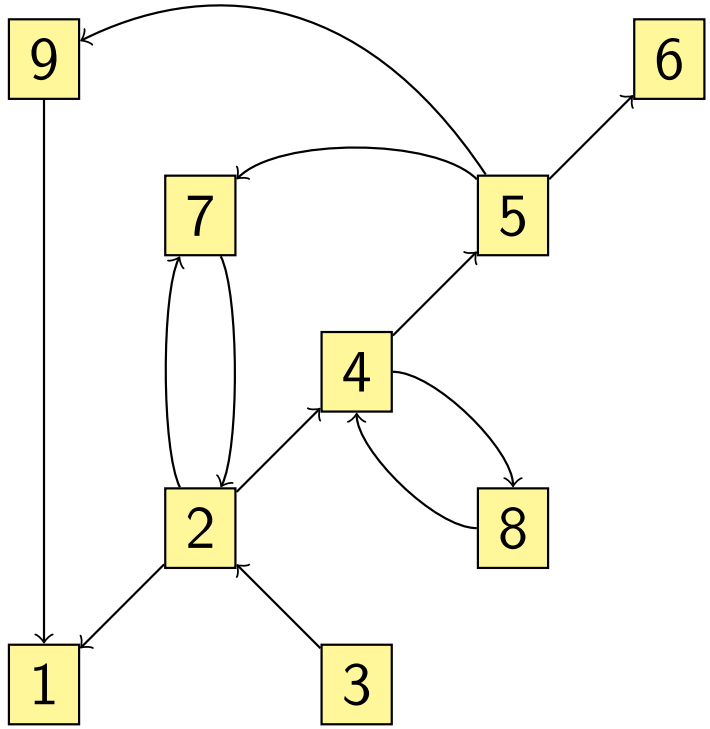


- Els veïns del vèrtex que està essent visitat es posen en una cua.
- recorregut$(G, 1)$ visita els vèrtexos de G en l'ordre següent:

$$1, 2, 9, 3, 4, 7, 5, 8, 6$$

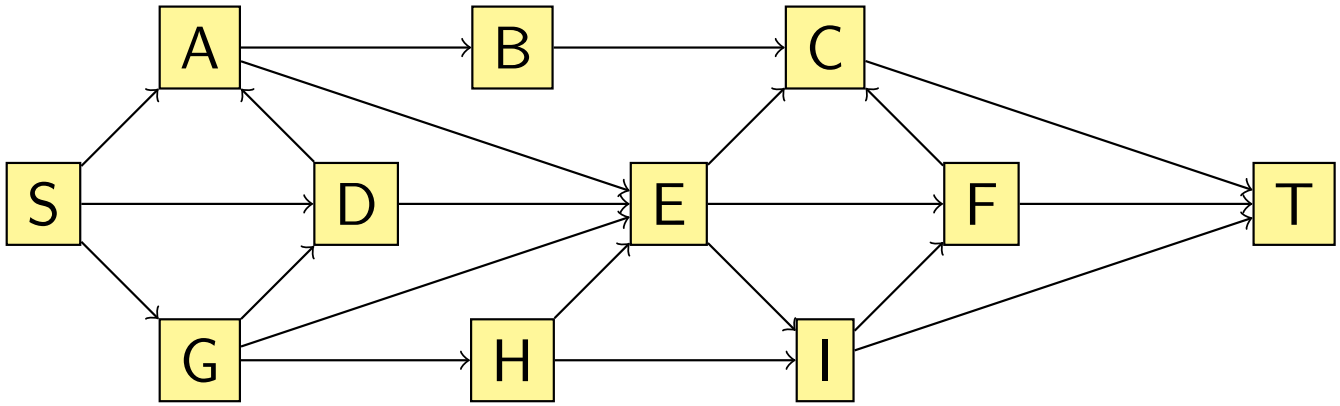
- És una generalització del recorregut d'un arbre per nivells.

- Els recorreguts en profunditat o en amplada visiten tots els vèrtexos d'un graf G, si G és un graf dirigit amb un únic component fortament connex o un graf no dirigit amb un únic component connex, és a dir, si tots els vèrtexos de G són accessibles des del vèrtex inicial v.



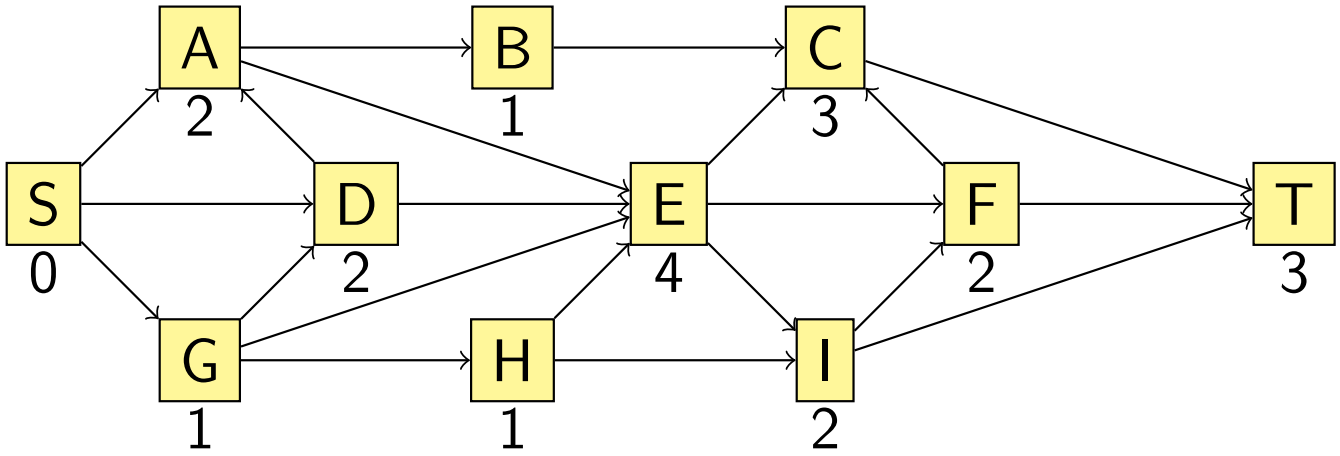
- **procedure** recorregut (G: graf)
  for all vèrtex v de G **do**
    visitat[v] := false
  for all vèrtex v de G **do**
    recorregut(G, v)
- **procedure** recorregut (G: graf; s: vèrtex)
  if ¬visitat[s] **then**
    visitar el vèrtex s
    visitat[s] := true
    for all vèrtex v de G accessible des de s **do**
      if ¬visitat[v] **then**
        visitar el vèrtex v
        visitat[v] := true

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.

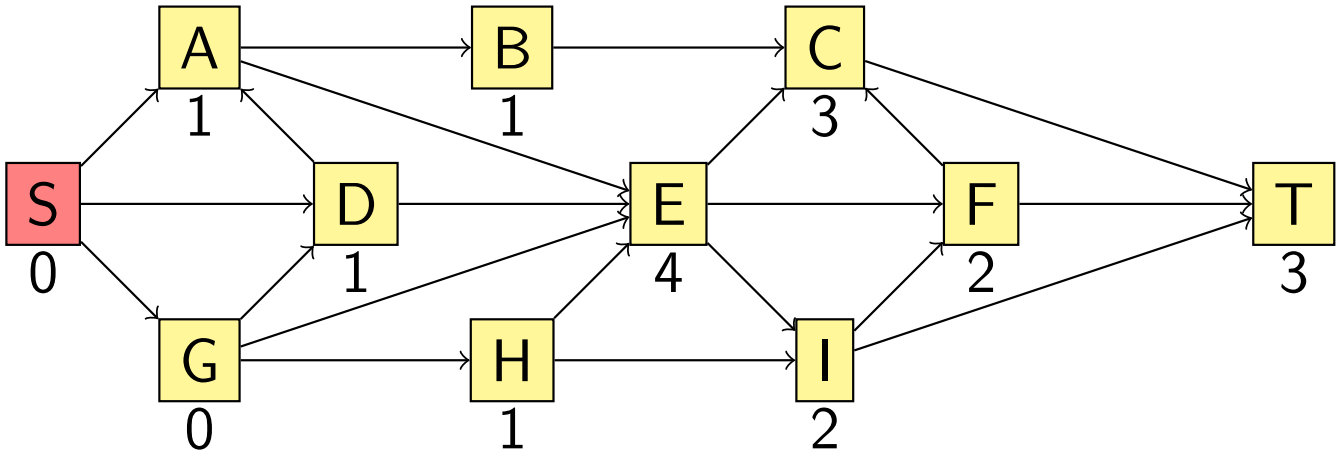


- L'aplicació més usual és l'ordenació causal o temporal d'esdeveniments, com ara l'ordenació del pla d'estudis d'una carrera universitària en funció dels pre-requisits de les assignatures.
- L'interès de l'ordenació topològica rau també en el fet que un graf dirigit admet una ordenació topològica si i només si és acíclic.
- Es posen en una cua inicialment buida tots els vèrtexs de grau d'entrada zero. Mentre la cua no sigui buida, es desencua un vèrtex i es decrementa el grau d'entrada de tots els vèrtexs adjacents amb el vèrtex que d'acaba de desencuar, tot encuant els vèrtexs el grau d'entrada dels quals hagi arribat a zero. L'ordenació topològica ve donada per l'ordre en què es desencuen els vèrtexs.
- Es pot assolir un cost temporal i espacial lineal en el nombre de vèrtexs i arcs del graf amb una representació per llistes d'adjacència, estesa amb el grau d'entrada de cada vèrtex. El resultat d'usar una pila en comptes d'una cua és també una ordenació topològica, si el graf dirigit és acíclic.

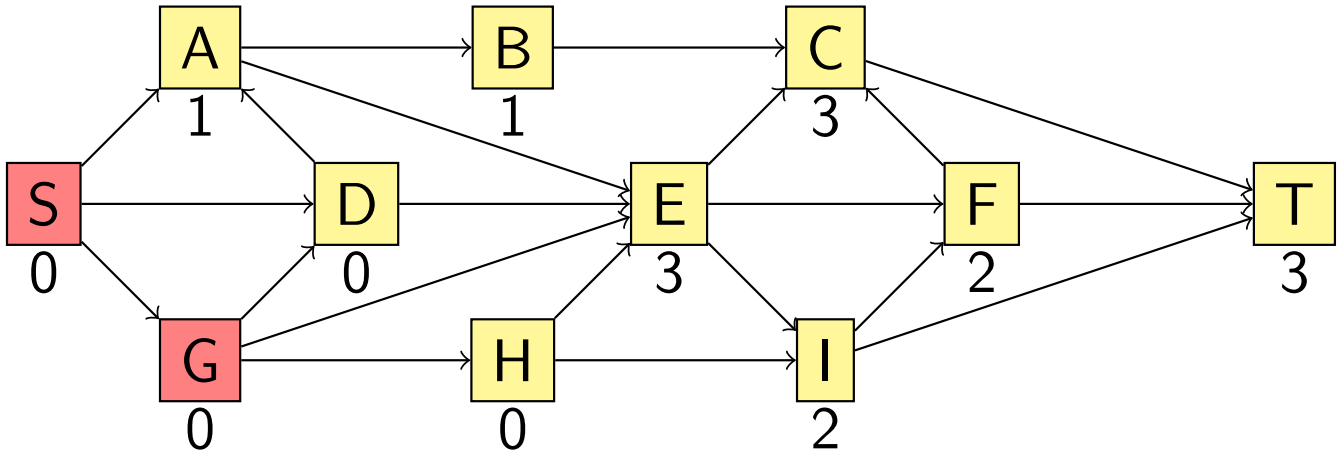
- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.

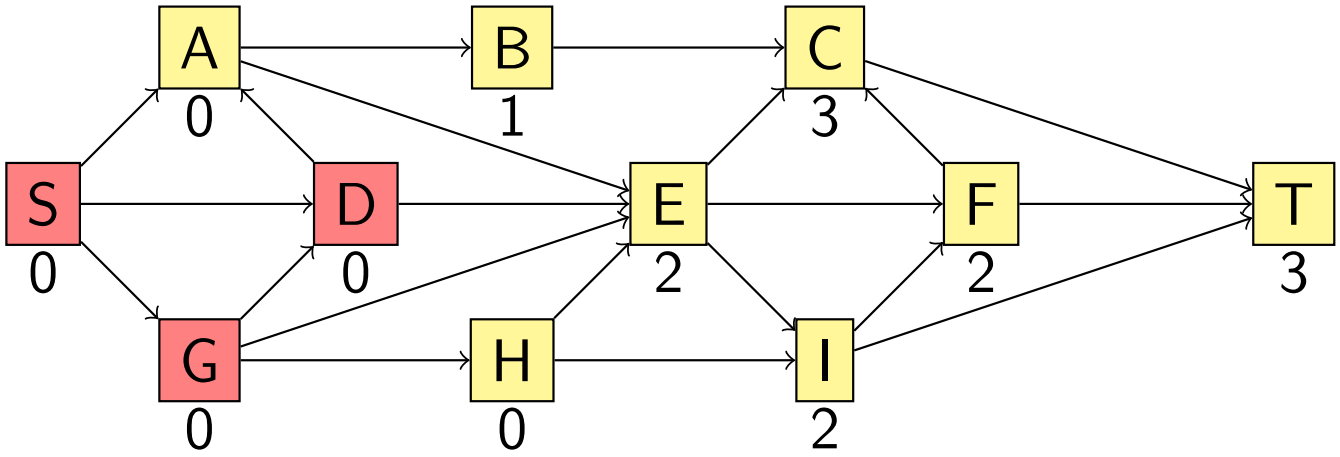


- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



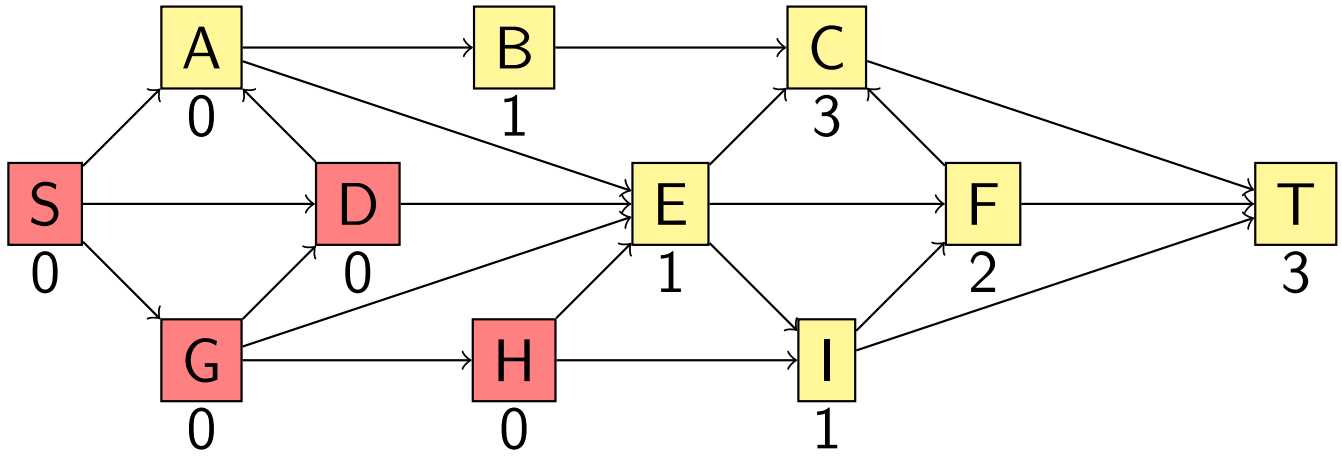
H	D

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.

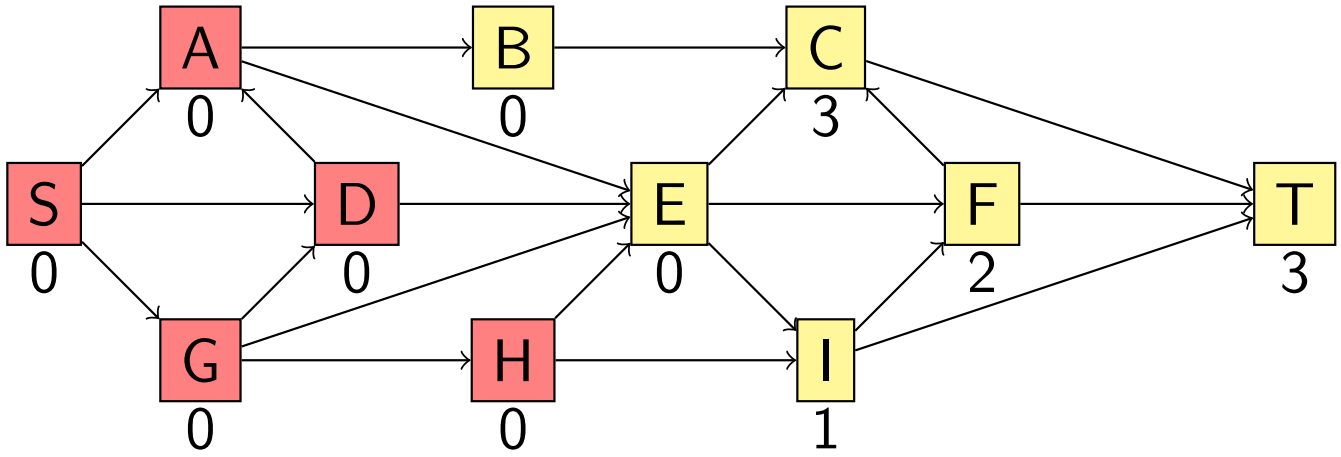


A	H

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.

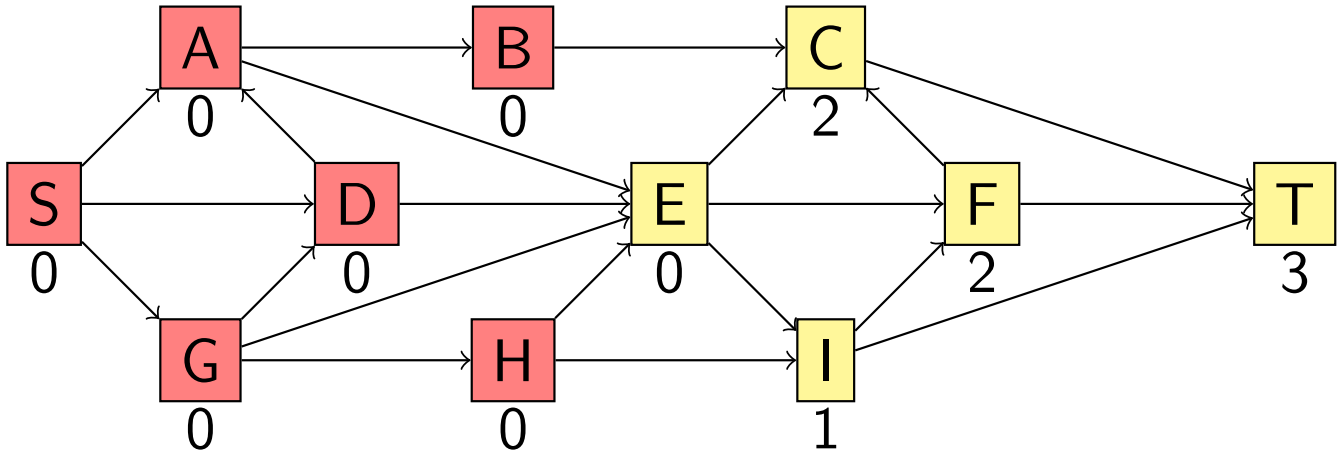


- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



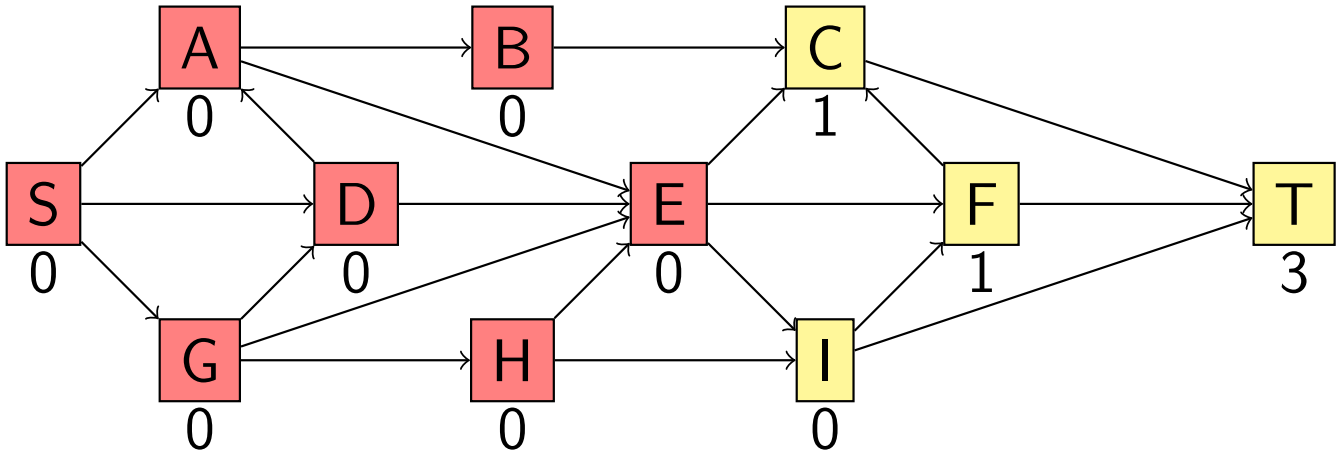
E	B

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



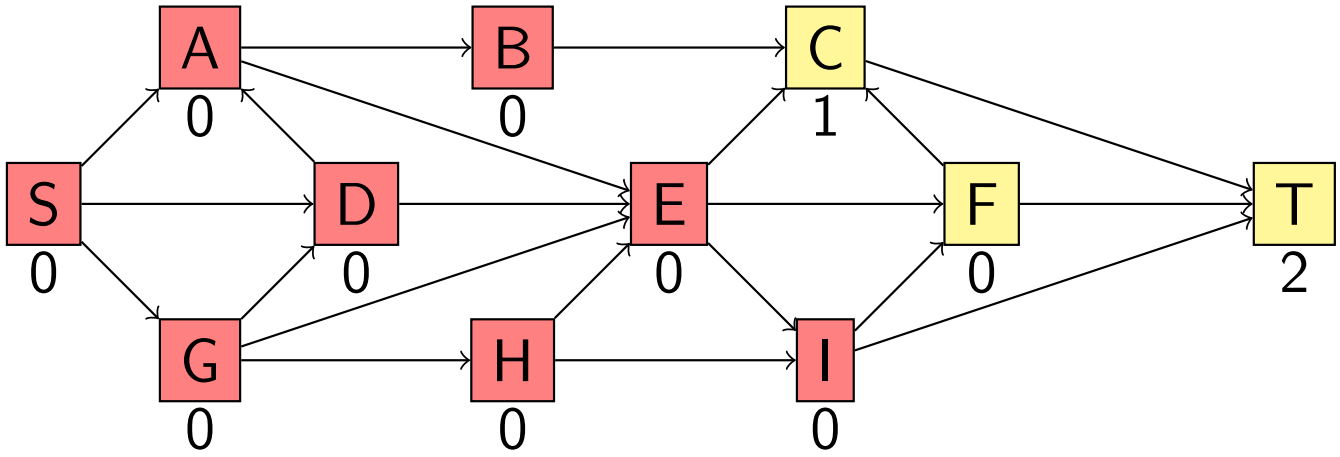
E

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



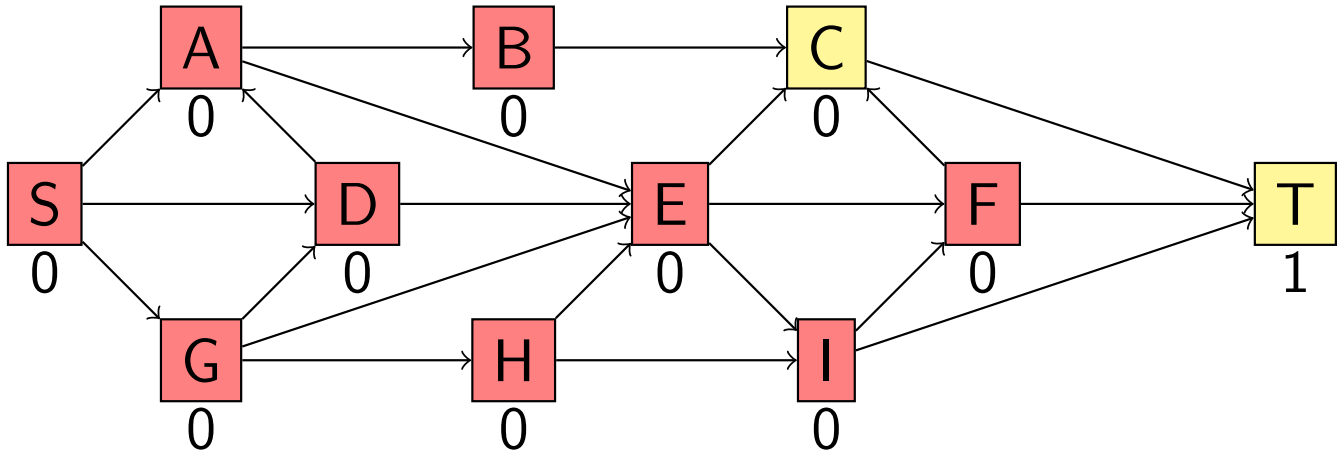
I

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



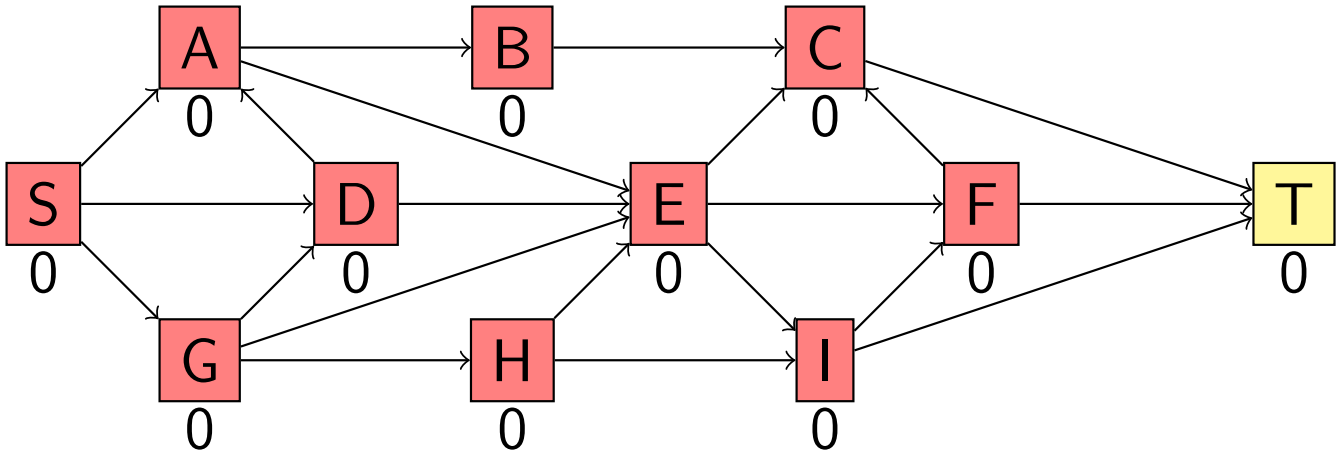
F

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.

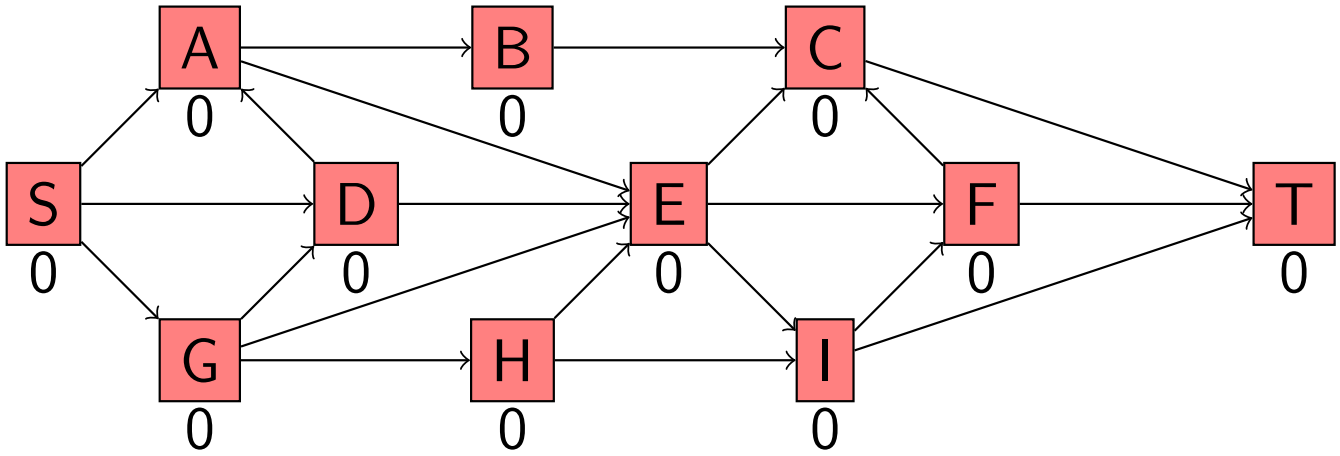


C

- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



- L'ordenació topològica d'un graf dirigit $G = (V, E)$ és una ordenació lineal dels vèrtexos de V tal que per a tot arc $(u, v) \in E$, el vèrtex u apareix abans que el vèrtex v en l'ordenació.
- Per exemple, l'ordenació topològica del graf dirigit següent és: S, G, D, H, A, B, E, I, F, C, T.



- **procedure** topsort (G: graph; **sort** order: **array** $[1..n]$ **of vertex**)

```
var q: queue
var u, v: vertex
var counter: int
counter := 0
create(q)
for all vertex v in G do
    if indeg(v) = 0 then
        enqueue(q, v)
while ¬empty(q) do
    dequeue(q, u)
    counter := counter + 1
    order[counter] := u
    for all vertex v adjacent to u in G do
        indeg(v) := indeg(v) − 1
        if indeg(v) = 0 then
            enqueue(q, v)
```

Els problemes marcats amb una estrelleta ($\star$) són més difícils. Si no es diu el contrari i us cal, considereu que tots els logaritmes són en base 2.

- 5.3
- 5.4
- 5.5
- 5.6
- 5.7
- 5.8
- 5.11
- 5.12
- 5.13 (en teoria)
- 5.14
- 5.15
- 5.16
- $\star$ 5.19
- 5.22
- 5.23
- 5.26
- 5.30

Problema (5.3)

Dibuixeu el graf $G = (V, E)$ *donat per:*

- $V = \{1, 2, 3, 4, 5, 6\}$
- $E = \{\{1, 2\}, \{1, 4\}, \{3, 2\}, \{4, 5\}, \{5, 1\}, \{5, 2\}\}$

És connex? Si no ho és, quants components connexos té?

Problema (5.4)

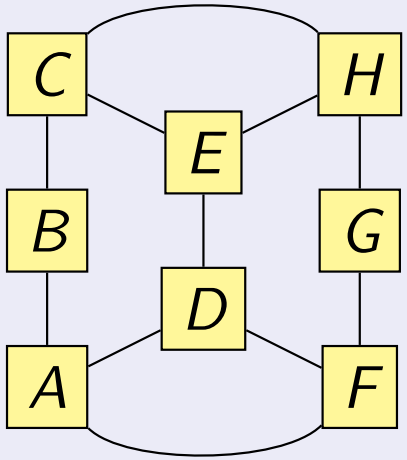
Dibuixeu el graf dirigit $G = (V, E)$ *donat per:*

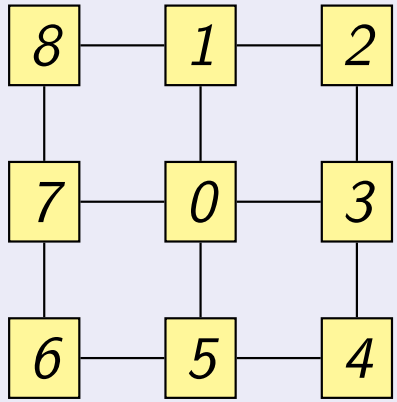
- $V = \{1, 2, 3, 4, 5\}$
- $E = \{(1, 2), (1, 4), (3, 2), (4, 5), (5, 1), (5, 2)\}$

És fortament connex? És feblement connex?

Problema (5.5)

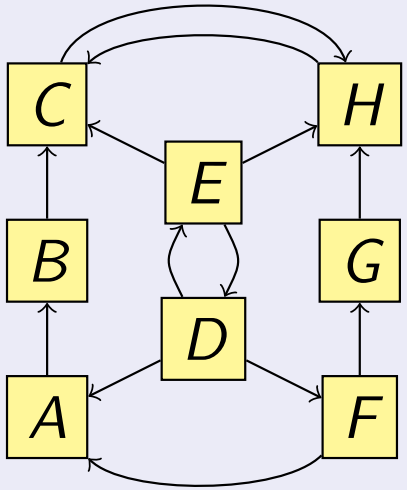
Mostreu com es representen els dos grafs següents utilitzant una matriu d'adjacència i llistes d'adjacència. Compteu el grau de cada vèrtex. Calculeu el diàmetre de cada graf.

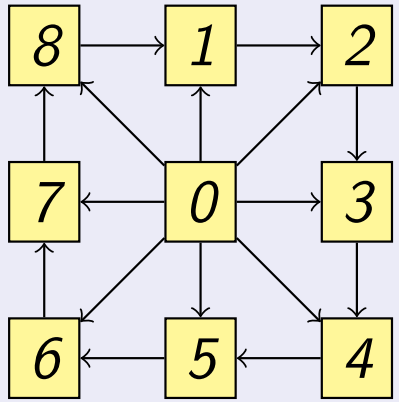




Problema (5.6)

Mostreu com es representen els grafs dirigits següents utilitzant una matriu d'adjacència i llistes d'adjacència. Compteu el grau d'entrada i de sortida de cada vèrtex. Digueu si els grafs són fortament o feblement connexos.





Problema (5.7)

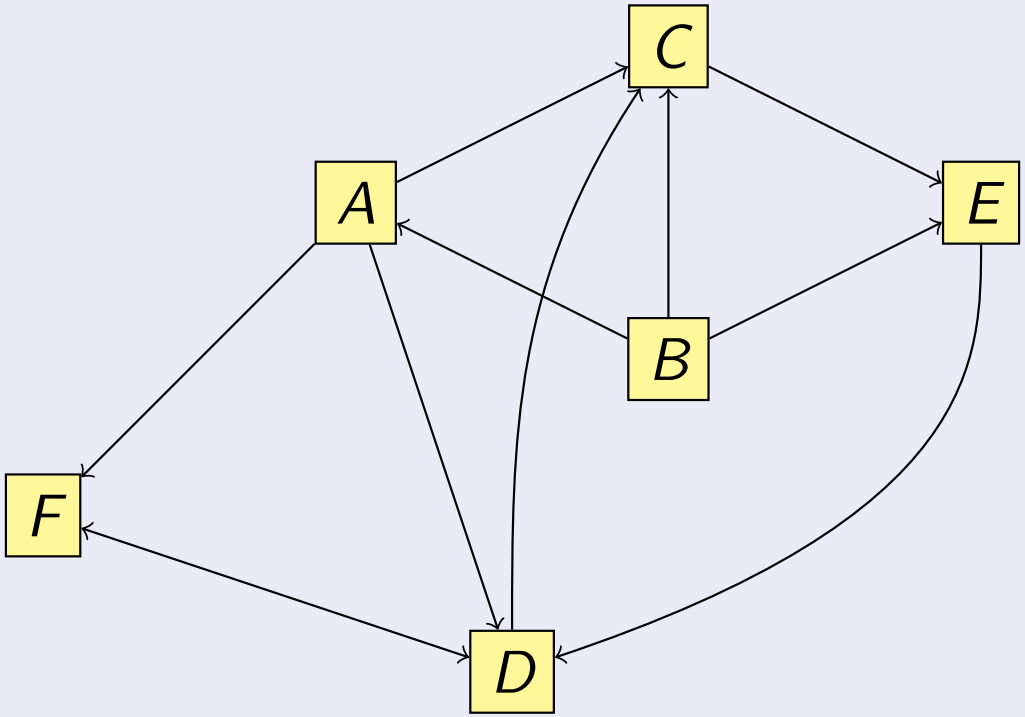
Demostreu que en un graf no dirigit $G = (V, E)$ *es té* $\sum_{u \in V} grau(u) = 2|E|$.

Problema (5.8)

Considereu un graf no dirigit amb cinc vèrtexs, tots de grau tres. Si existeix tal graf, feu-ne un dibuix. Si no n'existeix cap, doneu una explicació enraonada.

Problema (5.11)

Llisteu totes les possibles seqüències de visita dels vèrtexs d'aquest graf dirigit, tot aplicant un recorregut en profunditat que comenci en el vèrtex E.



Problema (5.12)

Repetiu el problema anterior tot aplicant un recorregut en amplada que comenci en el vèrtex B.

Problema (5.13)

Dissenyeu i analitzeu un algorisme per

- *calcular el grau d'un vèrtex donat en un graf no dirigit representat amb una matriu d'adjacència.*
- *calcular el grau d'un vèrtex donat en un graf no dirigit representat amb llistes d'adjacència.*
- *calcular el grau d'entrada i de sortida d'un vèrtex donat en un graf dirigit representat amb una matriu d'adjacència.*
- *calcular el grau d'entrada i de sortida d'un vèrtex donat en un graf no dirigit representat amb llistes d'adjacència.*
- *calcular el grau d'entrada i de sortida de tots els vèrtexs en un graf dirigit representat amb una matriu d'adjacència.*
- *calcular el grau d'entrada i de sortida de tots els vèrtexs en un graf no dirigit representat amb llistes d'adjacència.*

Problema (5.14)

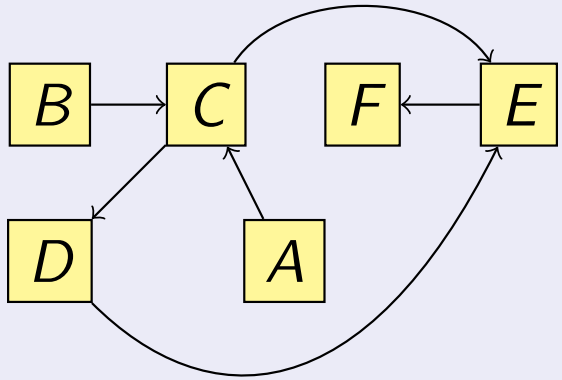
Considereu la funció següent sobre un graf no dirigit $G = (V, E)$ amb $n = |V|$ vèrtexs i $m = |E|$ arestes.

```
function misteri (G: graf)
    compt := 0
    for all vèrtex u ∈ G.V do
        for all vèrtex v ∈ G.veïns(u) do
            ++compt
    return compt
```

- *Expliqueu breument què retorna la funció.*
- *Quin cost té la funció, suposant que el graf G està implementat amb llistes d'adjacència?*

Problema (5.15)

Doneu, en ordre lexicogràfic i una a sota l'altra, totes les possibles ordenacions topològiques del graf dirigit acíclic següent:



Problema (5.16)

Si al graf anterior li afegíssim un vèrtex aïllat, quantes ordenacions topològiques tindria? (No les llisteu, només digueu quantes i expliqueu perquè.)

Problema (5.19)

En un graf no dirigit, un quadre és un cicle de llargada 4. Dissenyeu un algorisme que, donat un graf no dirigit, digui si aquest conté algun quadre. Analitzeu la seva eficiència en diferents representacions.

Problema (5.22)

Dissenyeu i analitzeu un algorisme que en temps $O(|V|)$ determini si un graf no dirigit conté cicles o no.

Problema (5.23)

Dissenyeu i analitzeu un algorisme que determini si un graf dirigit conté cicles o no.

Problema (5.26)

Dissenyeu un algorisme de cost $O(|V|)$ que determini si un graf dirigit $G = (V, E)$ és un arbre arrelat o no.

Problema (5.30)

Demostreu que tot graf no dirigit i connex té, com a mínim, un vèrtex u tal que si eliminem el vèrtex u i totes les seves arestes incidents, el graf resultant continua sent connex. Dissenyeu i analitzeu un algorisme que trobi un vèrtex d'aquestes característiques.