Un grafo direccional acíclico *G* se puede *ordenar topológicamente*

La **ordenación topológica** de *G* es una ordenación lineal de todos los vértices

… tal que si *G* contiene la arista (*u*, *v*), entonces *u* aparece antes que *v* en la ordenación

```
dfs():
    for each u in V:
        u.color = white
        π[u] = null
    time = 0
    for each u in V:
        if u.color == white:
            time = dfsVisit(u, time)
```

```
dfsVisit(u, time):
    u.color = gray
    time = time+1
    u.d = time
    for each v in α[u]:
        if v.color == white:
            π[v] = u
            time = dfsVisit(v, time)
    u.color = black
    time = time+1
    u.f = time
    return time
```

El algoritmo de ordenación topológica

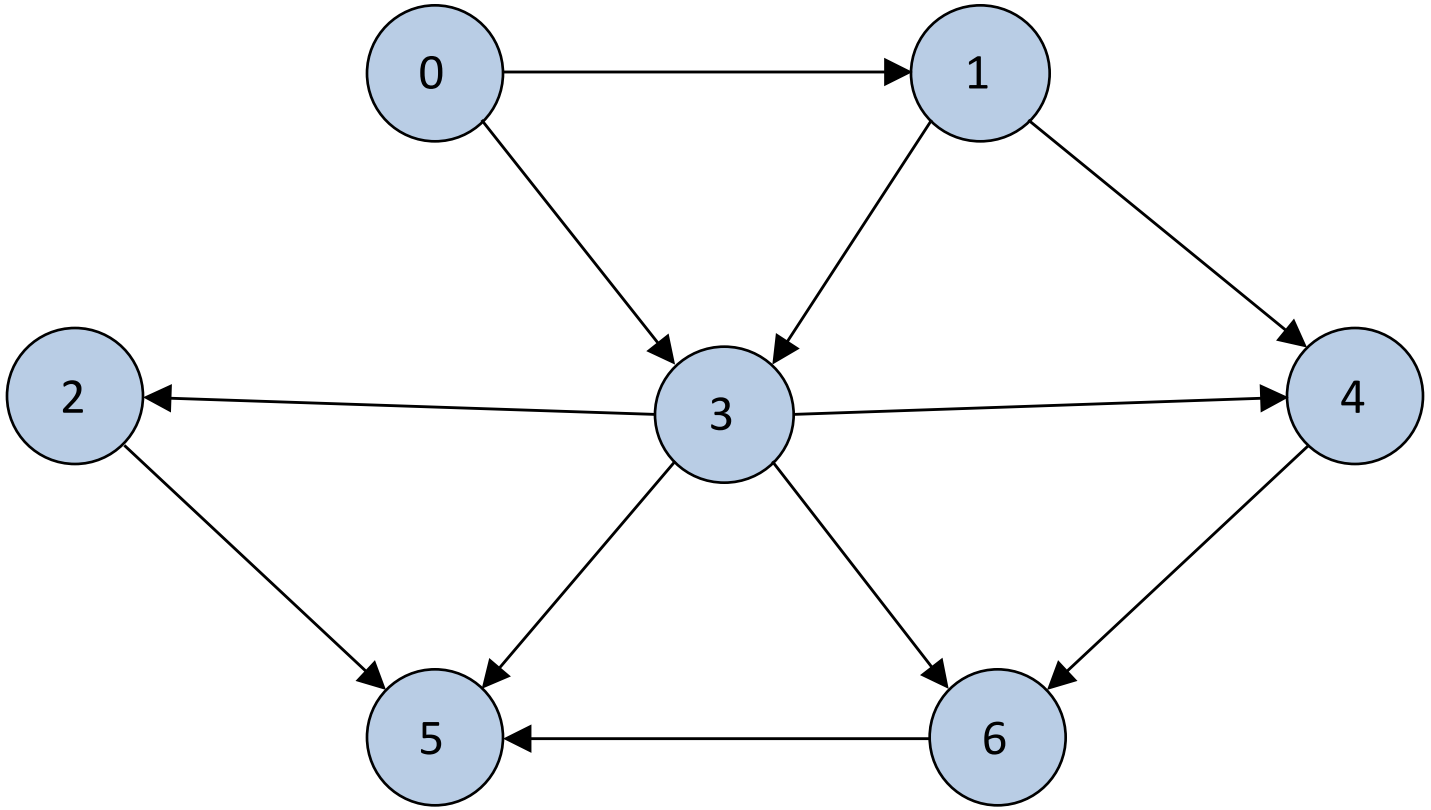
`topSort()`

1) Ejecute `dfs()` para calcular los tiempos *f* para cada vértice

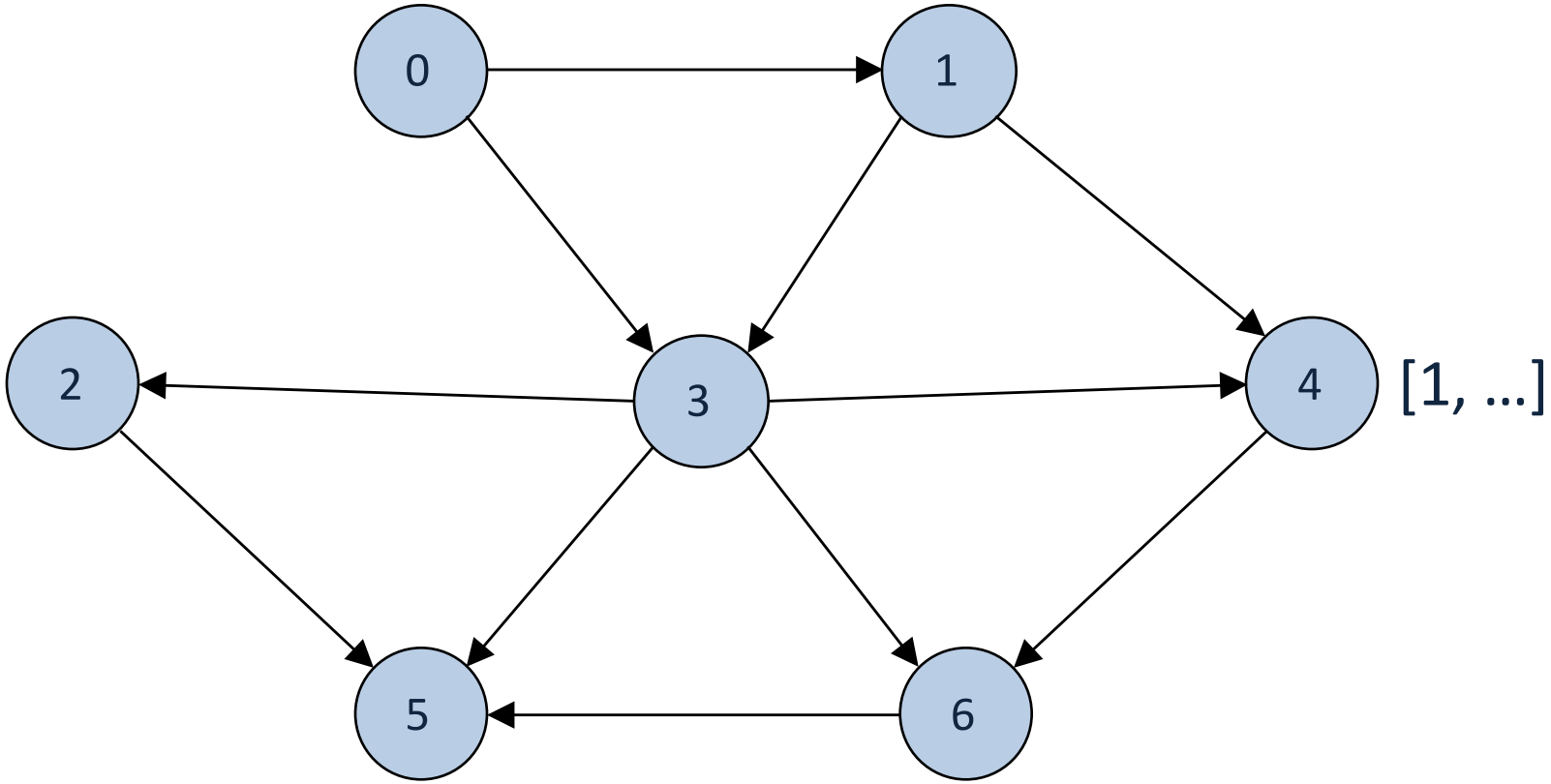
2) Cada vez que calcule el tiempo *f* para un vértice, inserte ese vértice al frente de una lista ligada

3) `return` la lista ligada de vértices

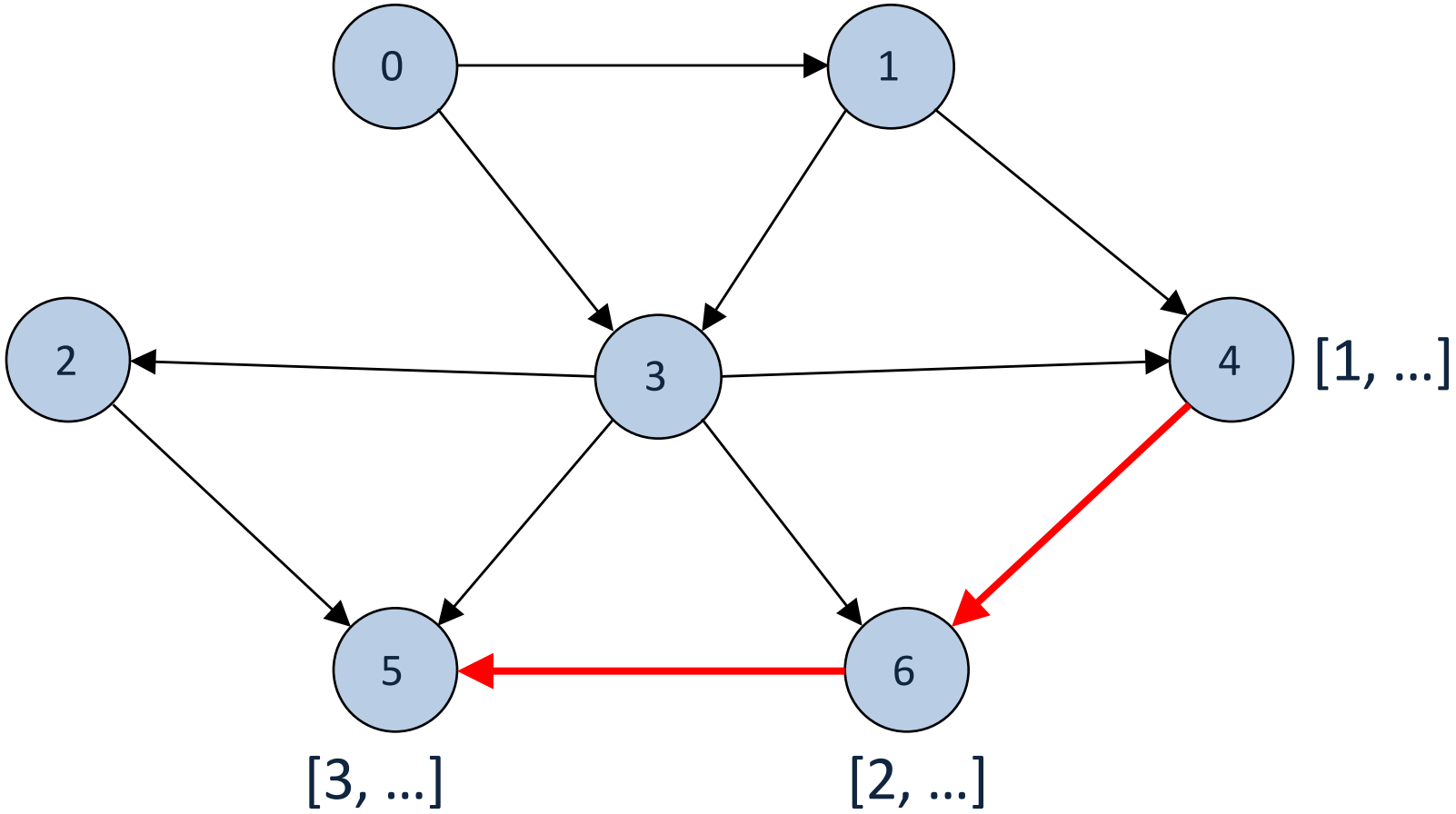
Un grafo, *G, direccional sin costo y acíclico*



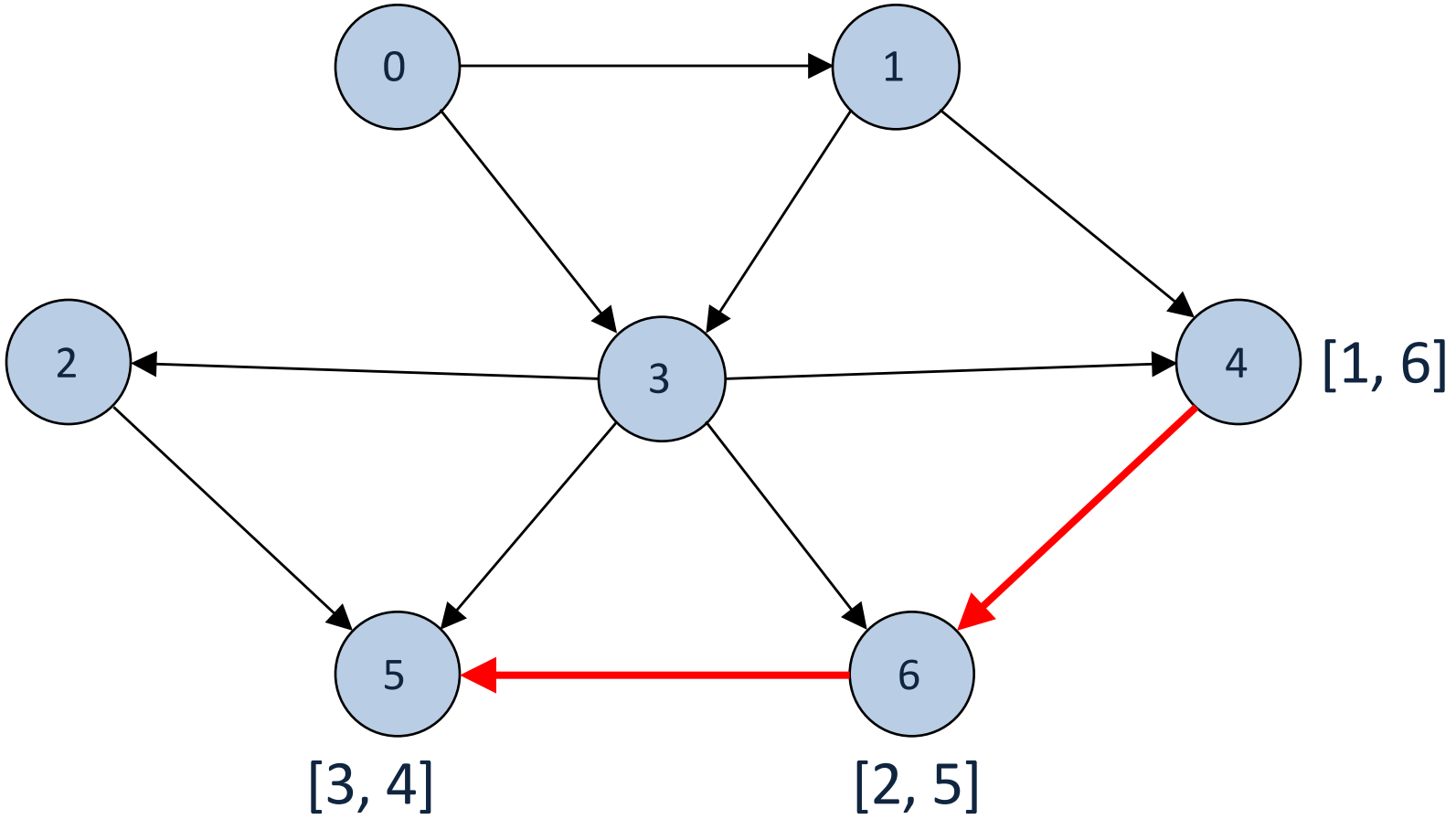
DFS de *G*, a partir del vértice 4 …



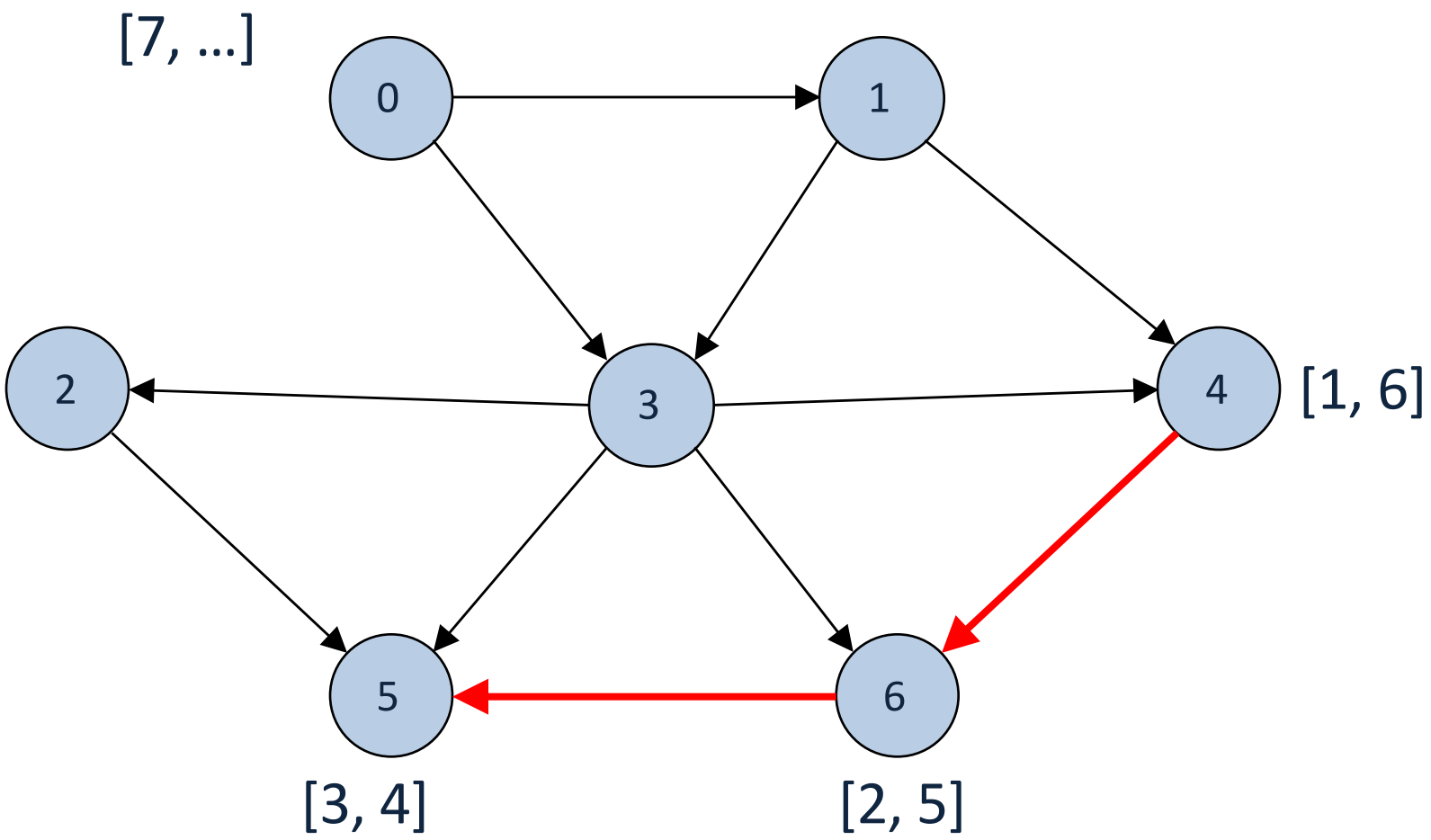
DFS de *G*, a partir del vértice 4 ...



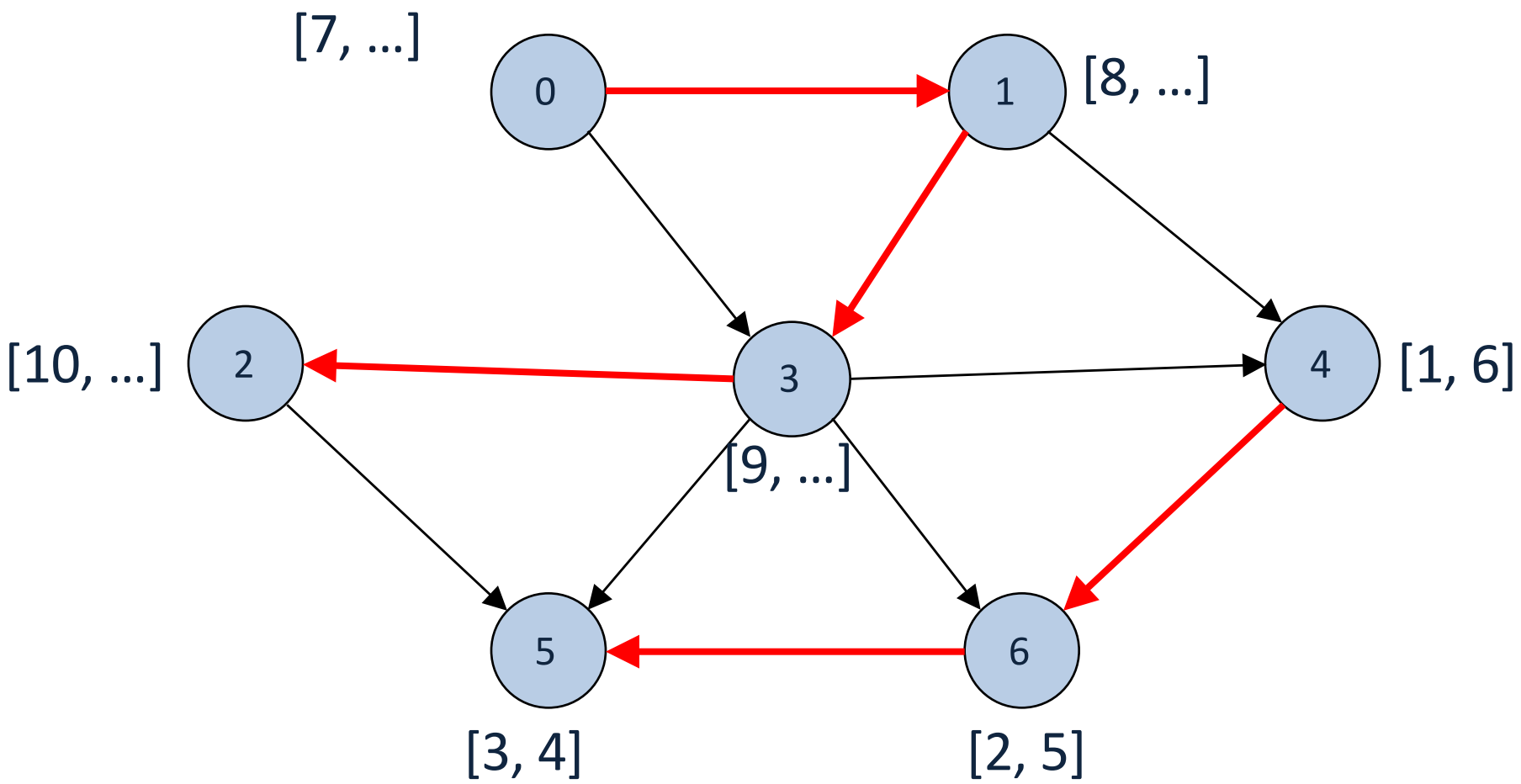
DFS de *G*, a partir del vértice 4 …



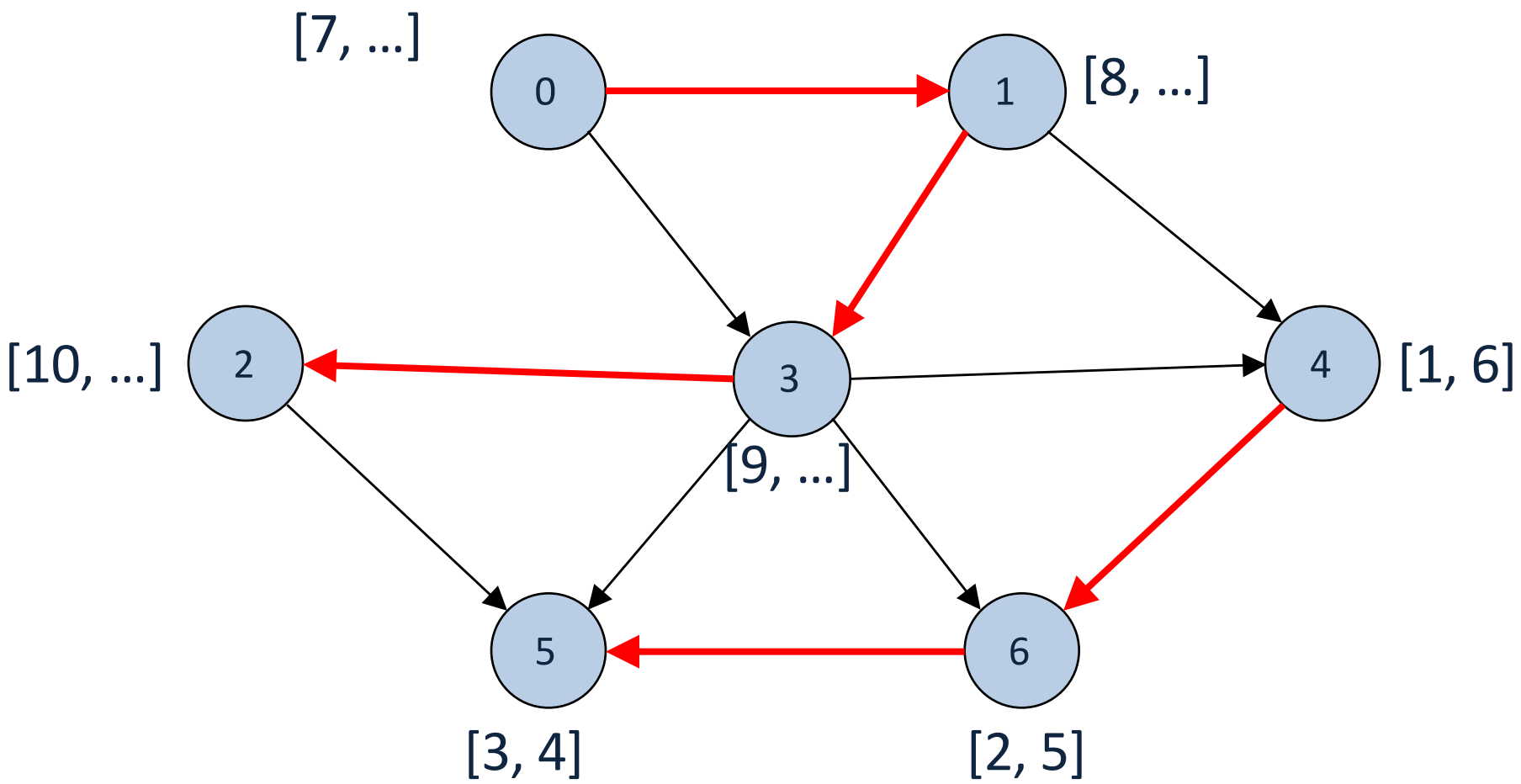
DFS de *G*, a partir del vértice 4 y, luego, del vértice 0



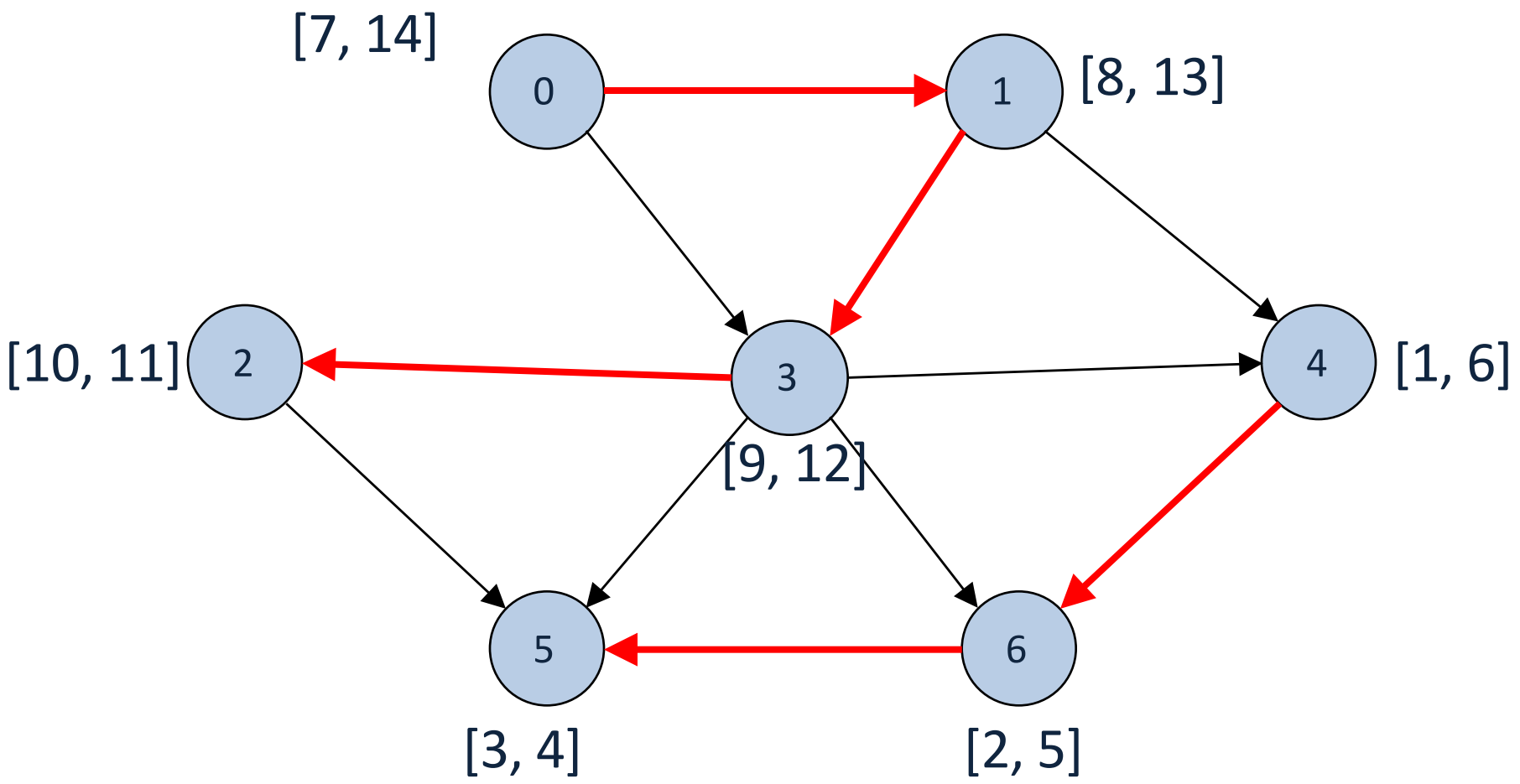
DFS de *G*, a partir del vértice 4 y, luego, del vértice 0



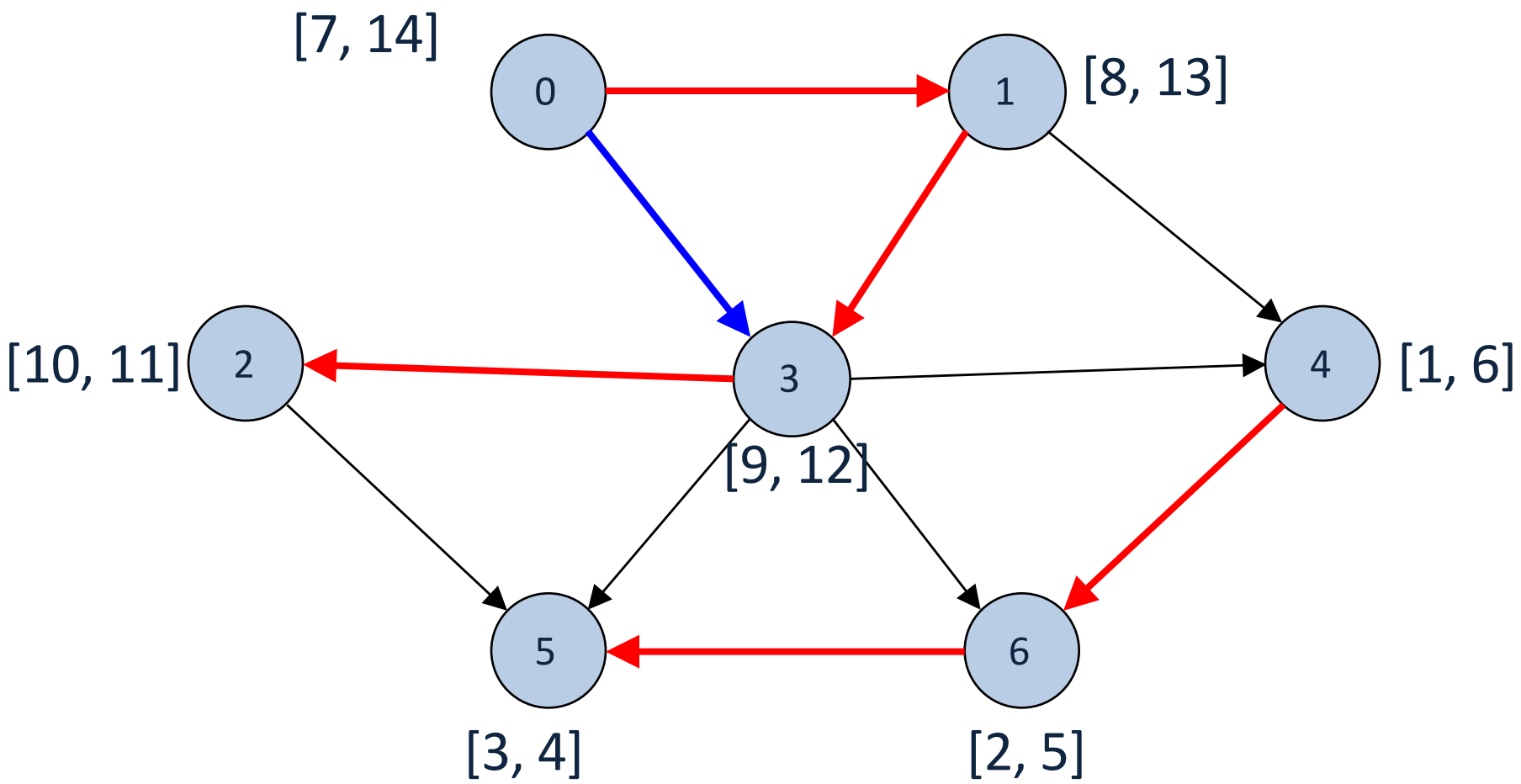
DFS de *G*, a partir del vértice 4 y, luego, del vértice 0



DFS de *G*, a partir del vértice 4 y, luego, del vértice 0



DFS de *G*, a partir del vértice 4 y, luego, del vértice 0



En un grafo que tiene ciclos es *imposible* producir un orden lineal de sus vértices:

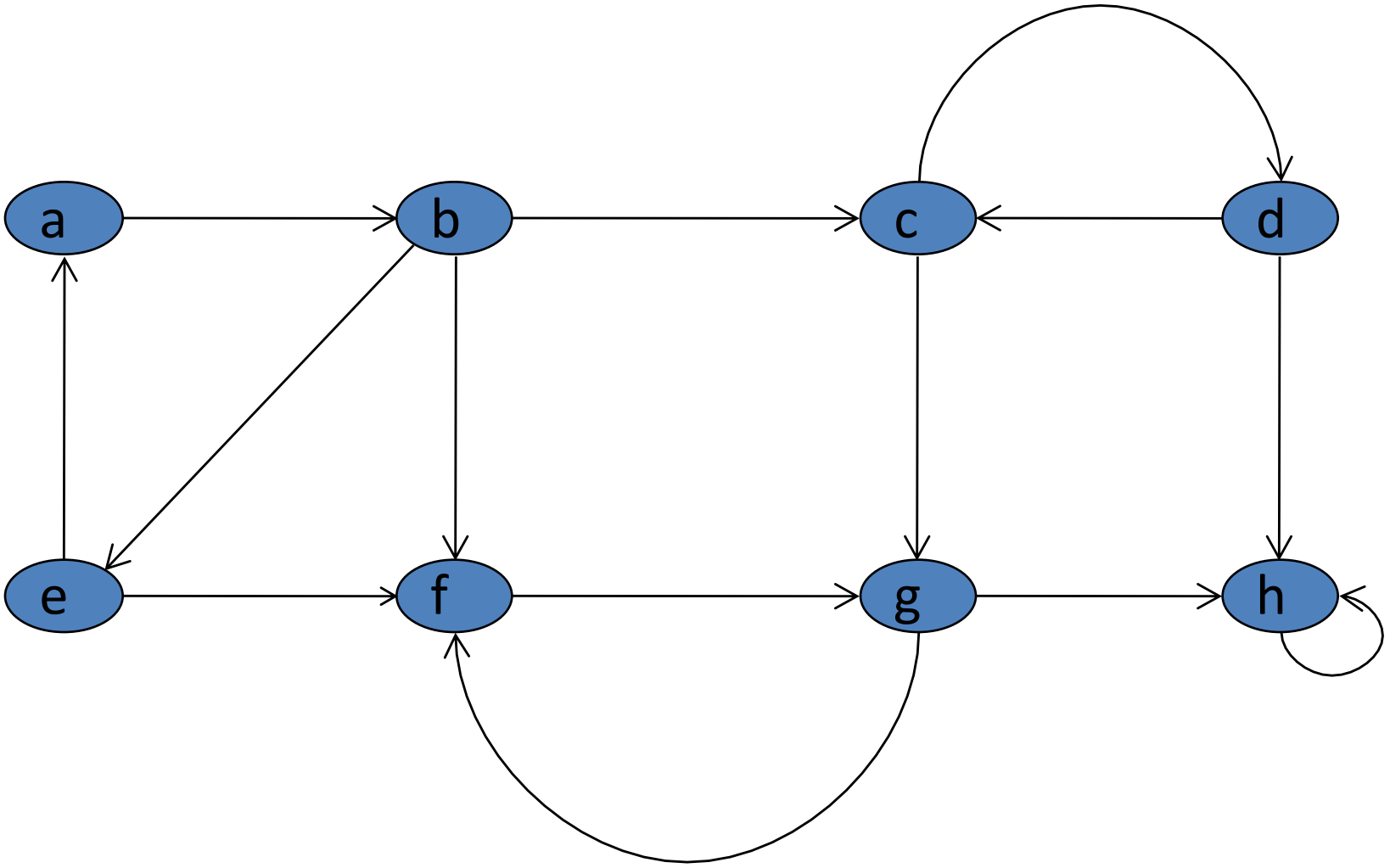
- p.ej., el grafo de la diap. #26

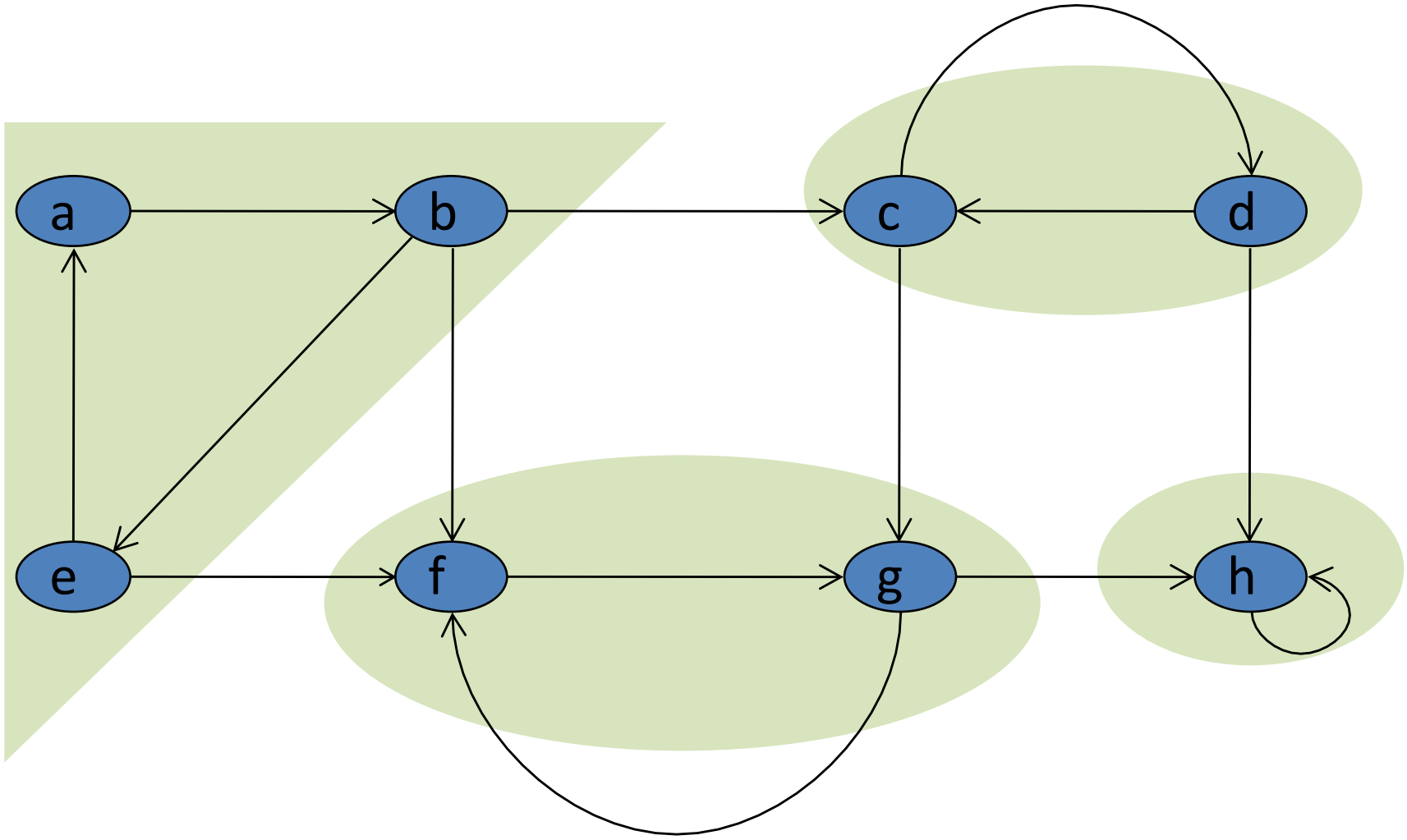
Un grafo direccional *G* es acíclico si y sólo si DFS de *G* no produce aristas hacia atrás

Para demostrar la corrección de `topSort()`, basta demostrar que para cualquier par de vértices *u*, *v*, si hay una arista de *u* a *v*, entonces $v.f < u.f$

¿Qué son las componentes *fuertemente* conectadas de un grafo direccional?

Las **componentes fuertemente conectadas** (SCC's) de un grafo direccional $G = (V, E)$ son conjuntos máximos de vértices $C \subseteq V$ tales que para todo par de vértices u y v en C, u y v son mutuamente alcanzables —se puede llegar a v desde u, y se puede llegar a u desde v


a
b
c
d
e
f
g
h


a
b
c
d
e
f
g
h

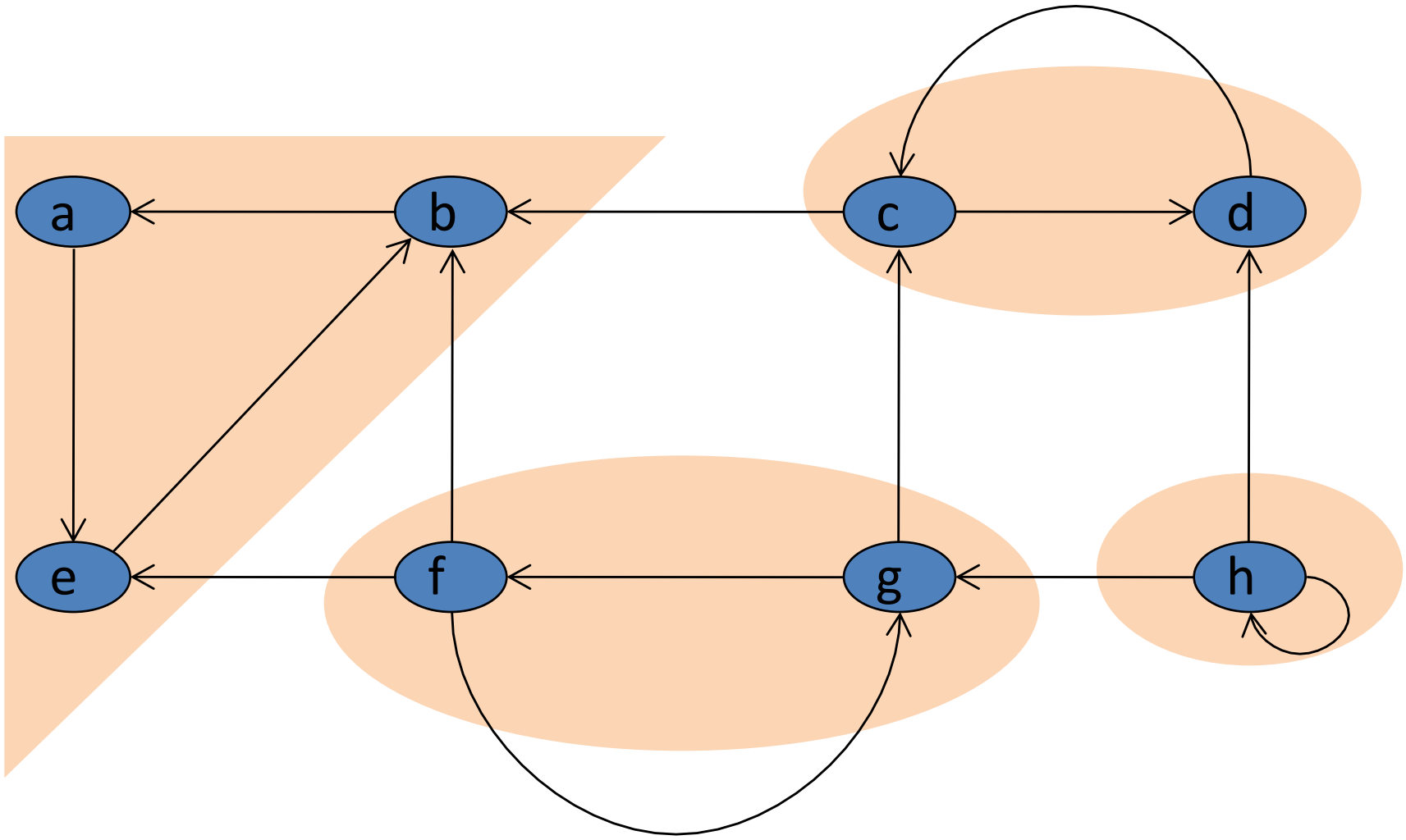
El algoritmo para determinar las SCC's de G usa el *grafo transpuesto* de G

$G^T = (V, E^T)$, en que $E^T = \{ (u, v) : (v, u) \in E \}$

… es decir, E^T consiste en las aristas de G con sus direcciones invertidas

G y G^T tienen exactamente las mismas componentes fuertemente conectadas

u y v son mutuamente alcanzables en G si y sólo si lo son en G^T


a
b
c
d
e
f
g
h

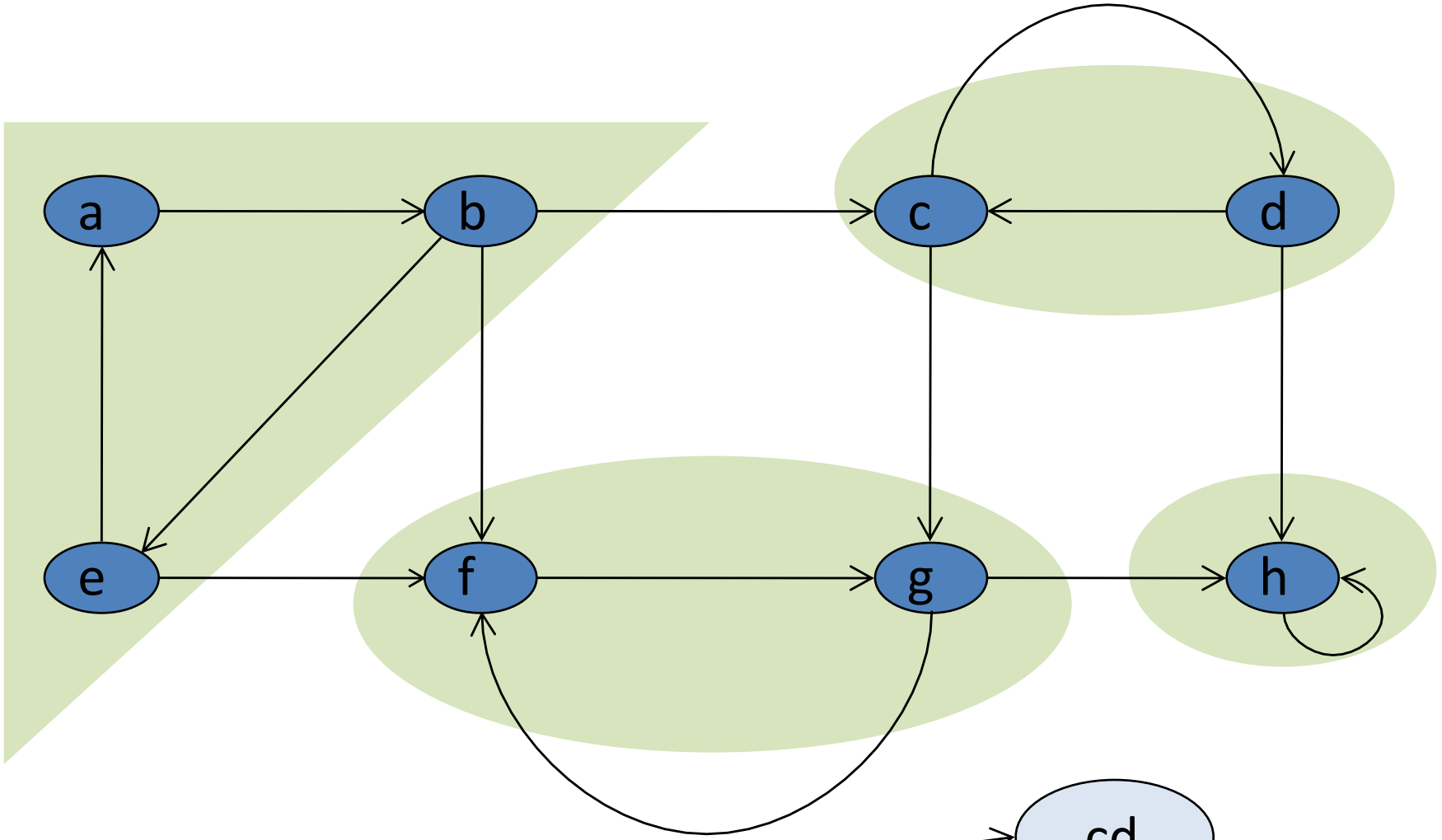
Definamos el *grafo de componentes* de G, $G^{SCC} = (V^{SCC}, E^{SCC})$

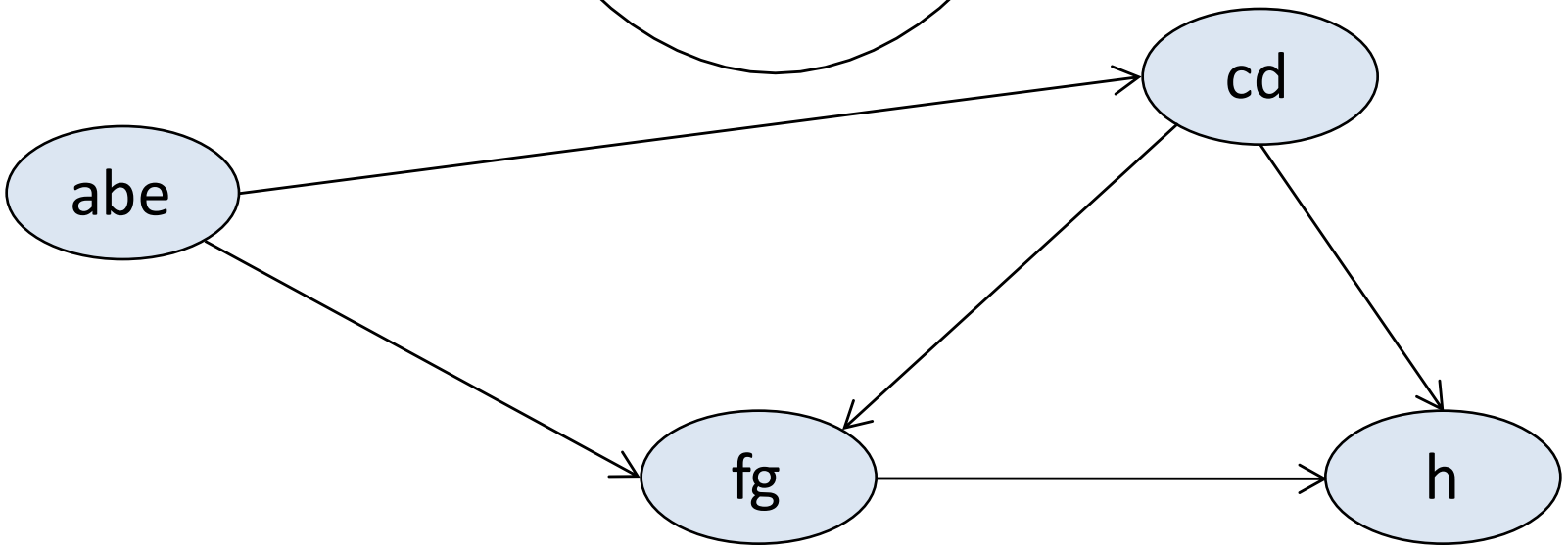
Supongamos que G tiene las componentes fuertemente conectadas $C_1, C_2, \ldots, C_k$

V^{SCC} es $\{v_1, v_2, \ldots, v_k\}$ y contiene un vértice v_i por cada componente fuertemente conectada C_i de G

Hay una arista $(v_i , v_j) \in E^{SCC}$ si G tiene una arista direccional (x, y) para algún $x \in C_i$ y algún $y \in C_j$

La propiedad clave es que G^{SCC} es un **grafo direccional acíclico** (DAG)


a
b
c
d
e
f
g
h

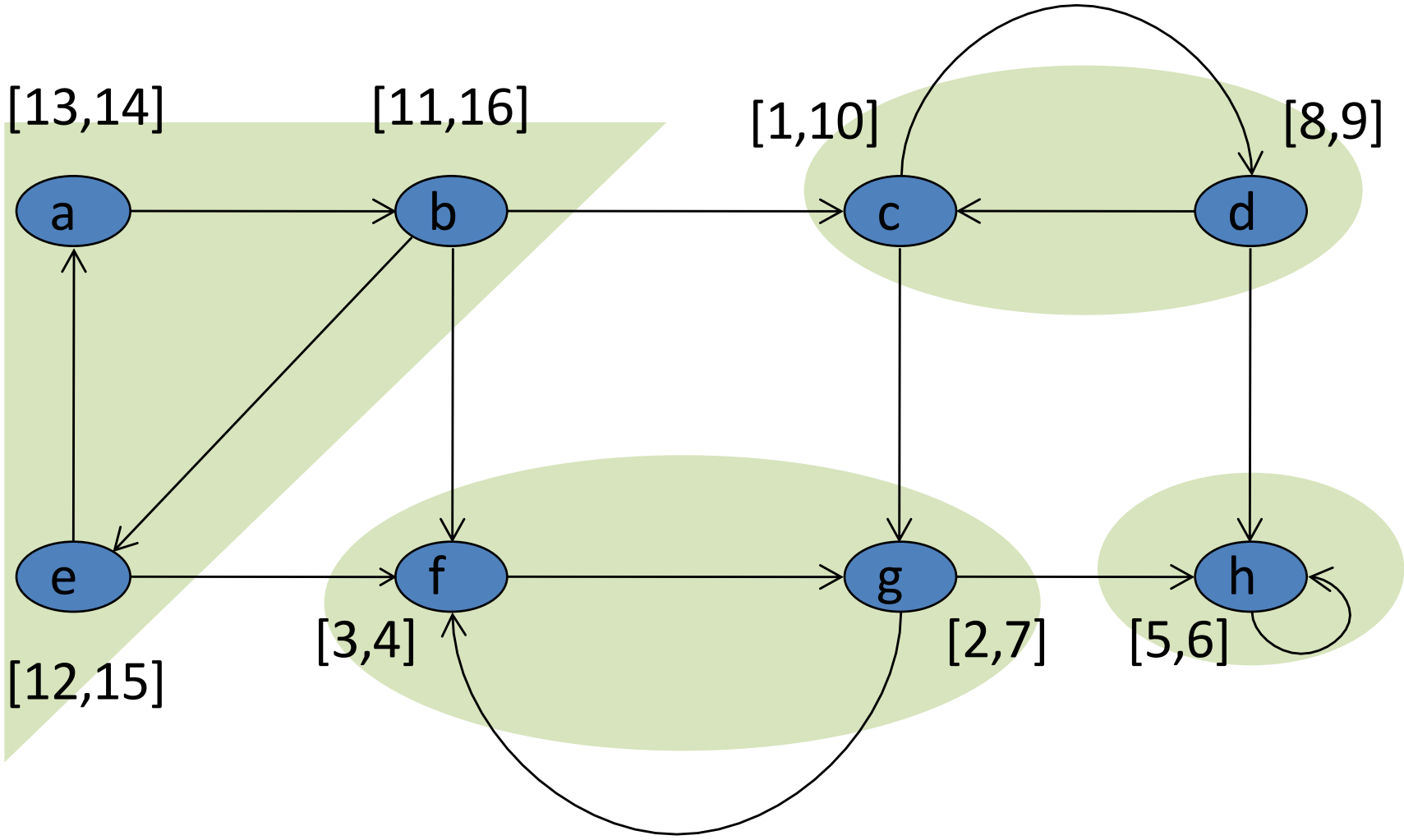

cd
abe
fg
h

Hagamos una exploración DFS de G

Sea $U \subseteq V$

Definimos $d(U) = \min_{u \in U}\{ u.d \}$ —el tiempo de descubrimiento más temprano de cualquier vértice en U

Definimos $f(U) = \max_{u \in U}\{ u.f \}$ —el tiempo de finalización más tardío de cualquier vértice en U

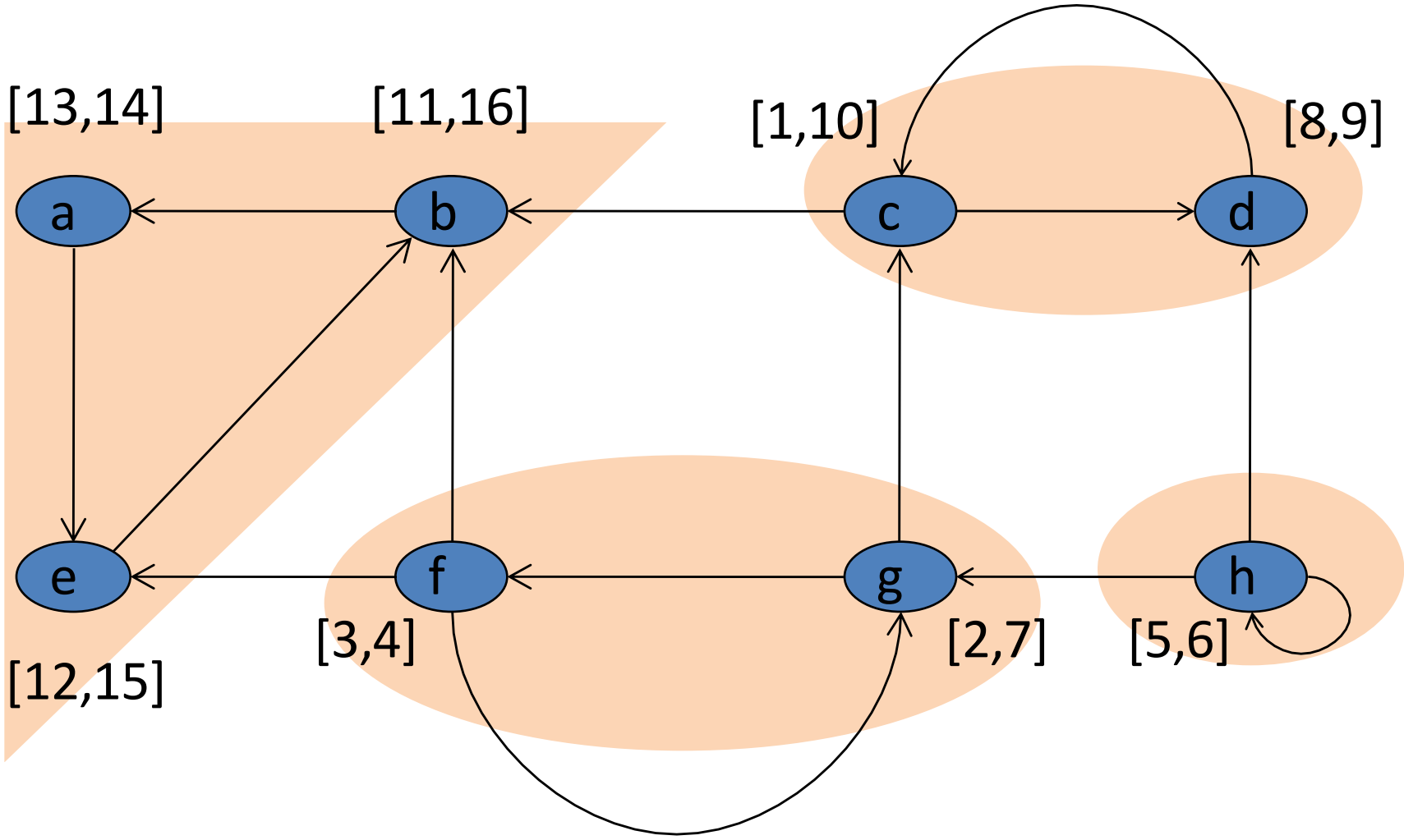

[13,14]
[11,16]
[1,10]
[8,9]
a
b
c
d
e
f
g
h
[12,15]
[3,4]
[2,7]
[5,6]

Una propiedad clave entre SCC's y tiempos de finalización

Sean C y D componentes fuertemente conectadas distintas de $G = (V, E)$:

- si hay una arista $(u, v) \in E$, en que $u \in C$ y $v \in D$, entonces $f(C) > f(D)$
- si hay una arista $(u, v) \in E^T$, en que $u \in C$ y $v \in D$, entonces $f(C) < f(D)$

Cada arista en G^T que va entre SCC's distintas va de una con un tiempo de finalización más temprano a otra con un tiempo de finalización más tardío


[13,14]
[11,16]
[1,10]
[8,9]
a
b
c
d
e
f
g
h
[12,15]
[3,4]
[2,7]
[5,6]

Hagamos ahora una exploración DFS de G^T

En el ciclo principal de `dfs()`, consideremos los vértices en orden decreciente de los $u.f$ determinados en la exploración DFS de G:

- empezamos con la SCC C cuyo tiempo de finalización es máximo
- la exploración empieza en un vértice x de C y visita todos los vértices de C
- no hay aristas en G^T de C a ninguna otra SCC —el árbol con raíz x contiene exactamente los vértices de C

En resumen, el algoritmo para encontrar SCC's de un grafo G es el siguiente

realizamos DFS de G, para calcular los tiempos de finalización de cada vértice

determinamos G^T

realizamos DFS de G^T, pero en el ciclo principal consideramos los vértices en orden decreciente de $u.f$, calculado antes

los vértices de cada árbol en el bosque primero-en-profundidad recién formado son una SCC diferente