Clase anterior: exploración en profundidad

Todos los vértices son inicialmente *blancos*

Un vértice se pinta de *gris* cuando es descubierto

Un vértice se pinta de *negro* cuando su lista de adyacencias ha sido examinada exhaustivamente

Clase anterior: DFS

```
dfs(V, E):
    for each u in V:
        u.color = white
    for each u in V:
        if u.color == white:
            dfsVisit(u)
```

```
dfsVisit(u):
    u.color = gray
    for each v in α[u]:
        if v.color == white:
            dfsVisit(v, time)
    u.color = black
```

Ahora agregamos tiempos de inicio y fin

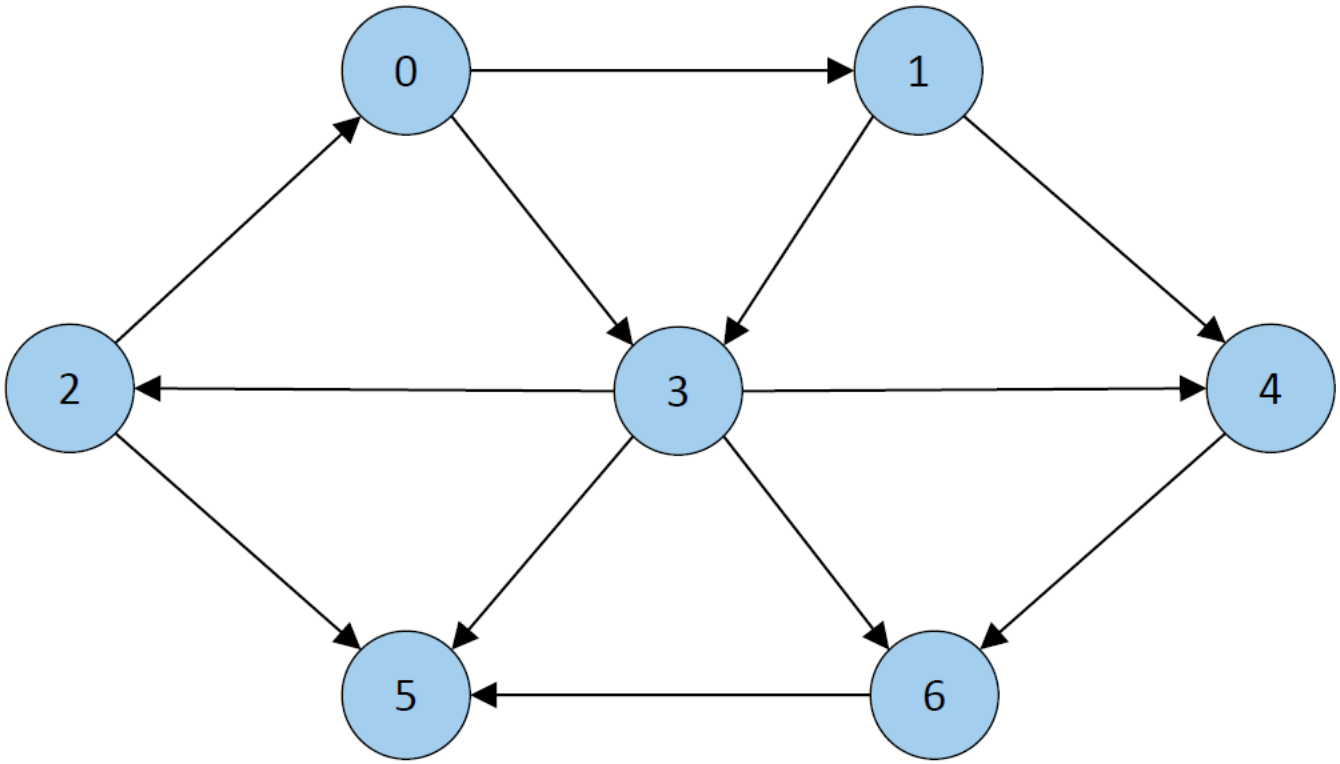
Cuando se visita un nodo blanco no solo se pinta de gris …

… además se marca el tiempo en el que se pinta

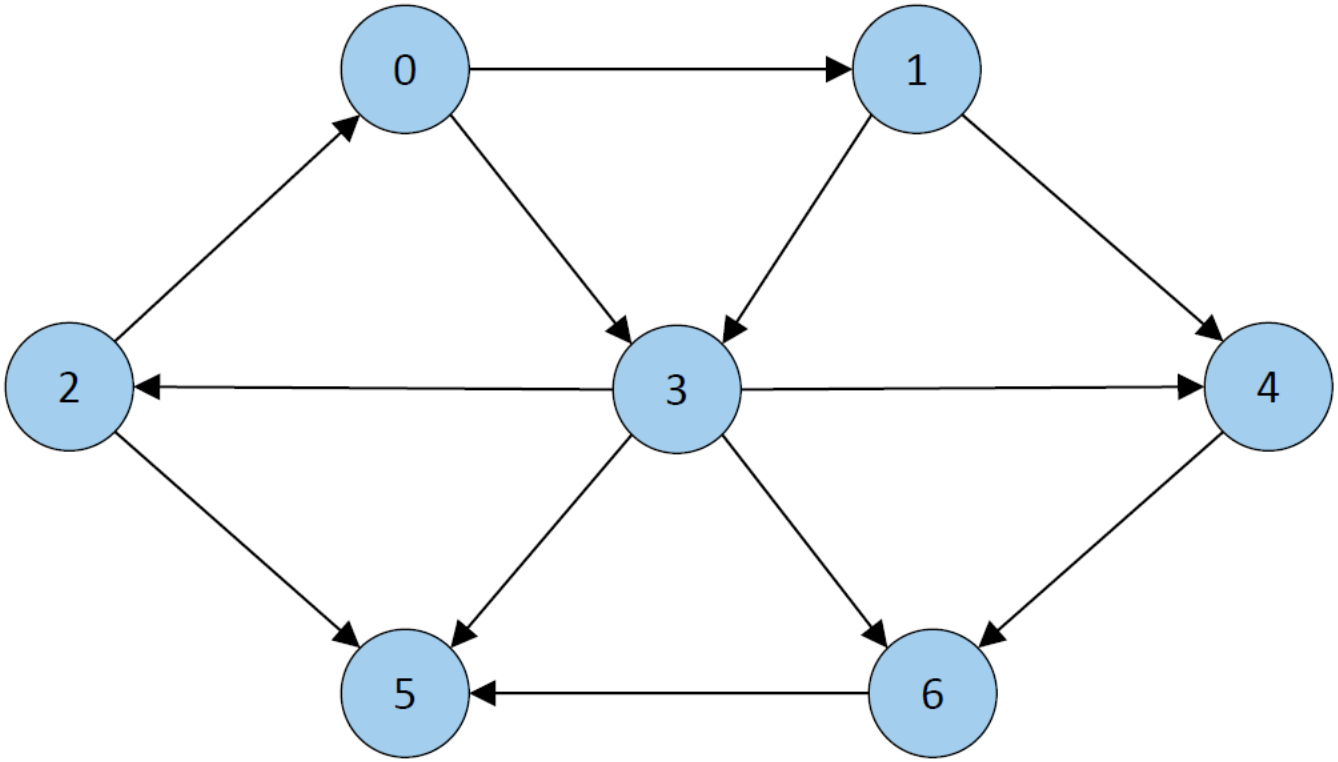
Similarmente, cuando se pinta de negro

Estos son el tiempo de inicio y de fin de un nodo, respectivamente

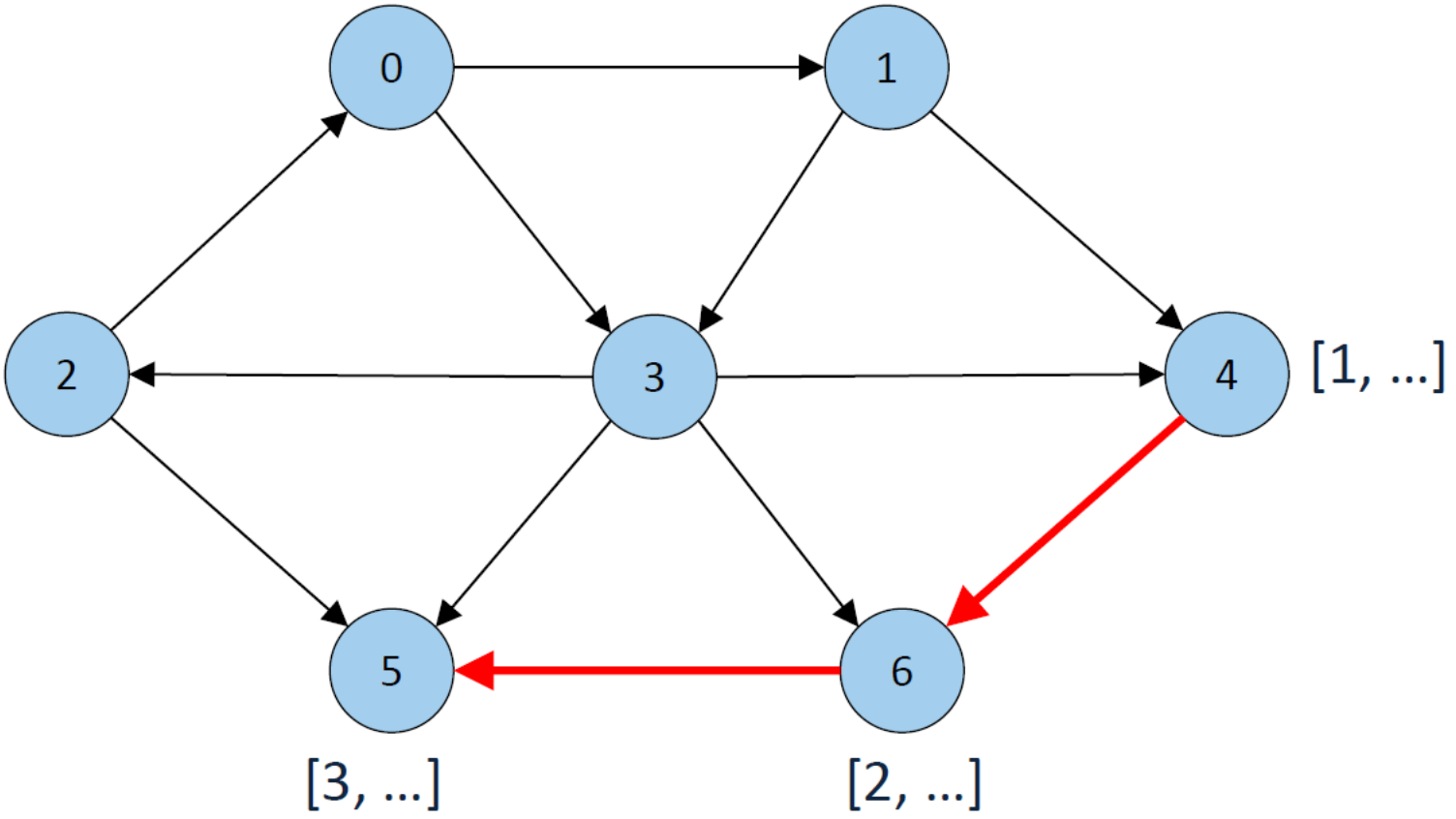
Ejemplo: Un grafo *G* direccional sin costos



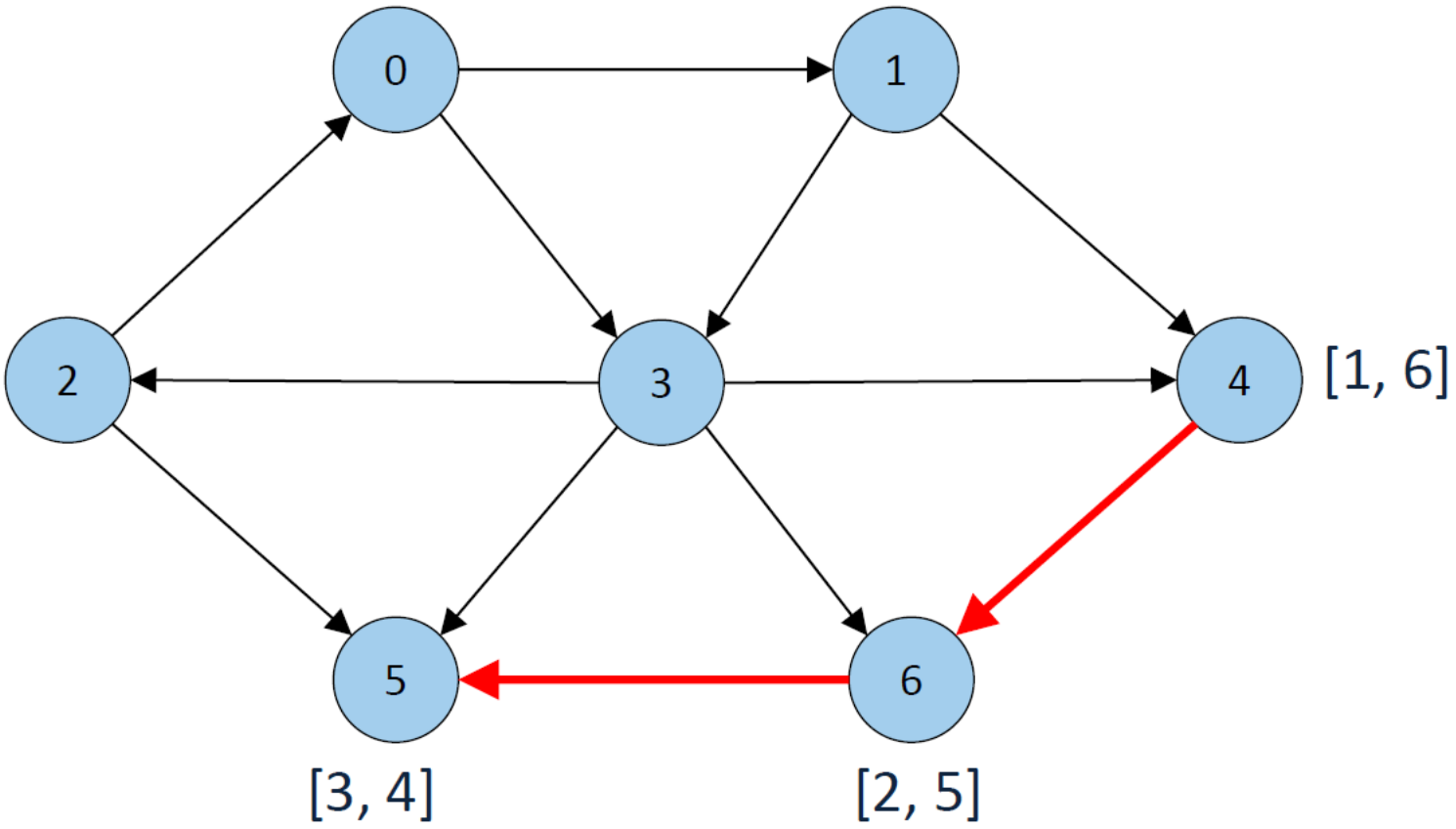
DFS_visit de G a partir del vértice 4



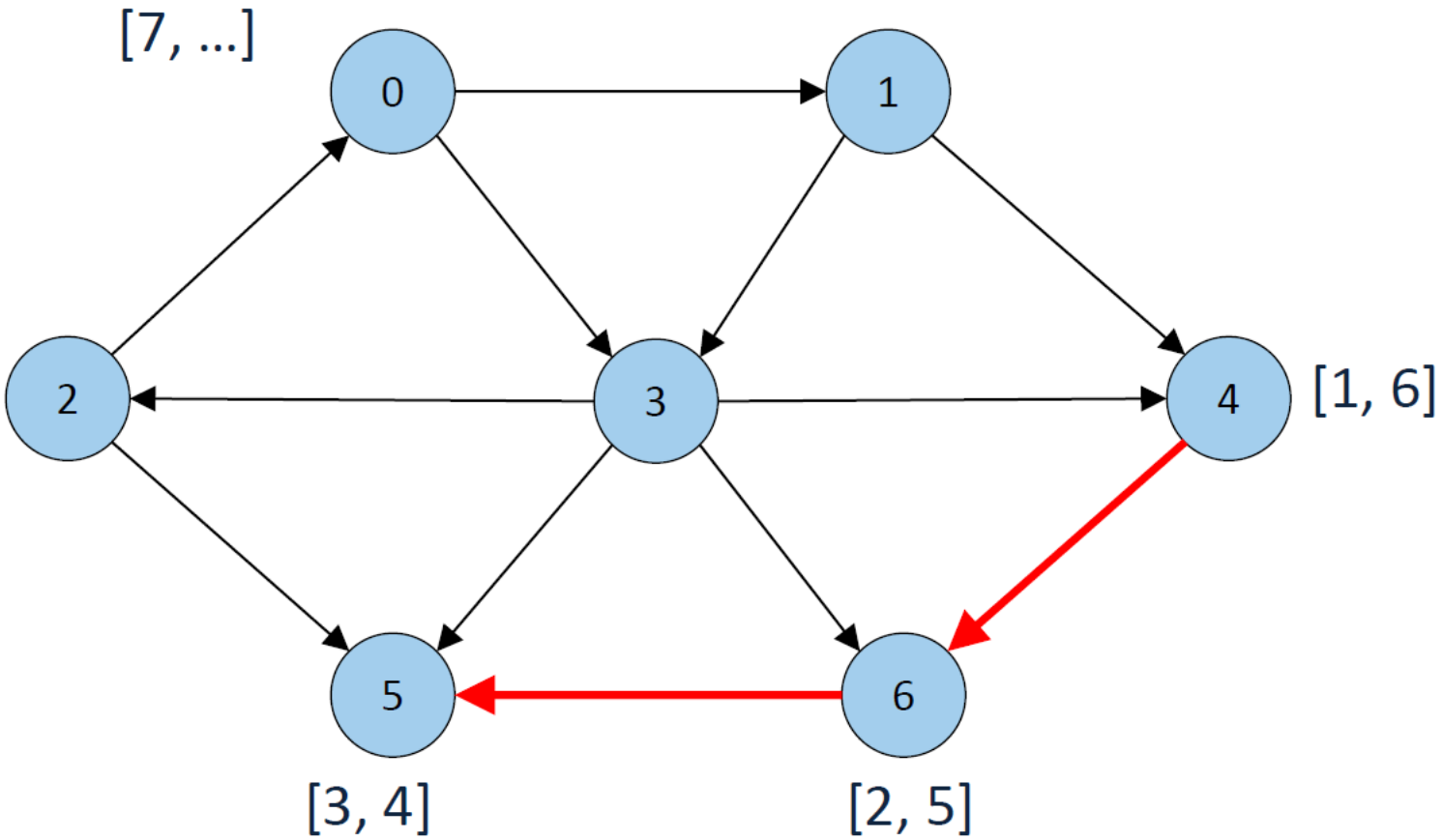
DFS_visit de G a partir del vértice 4



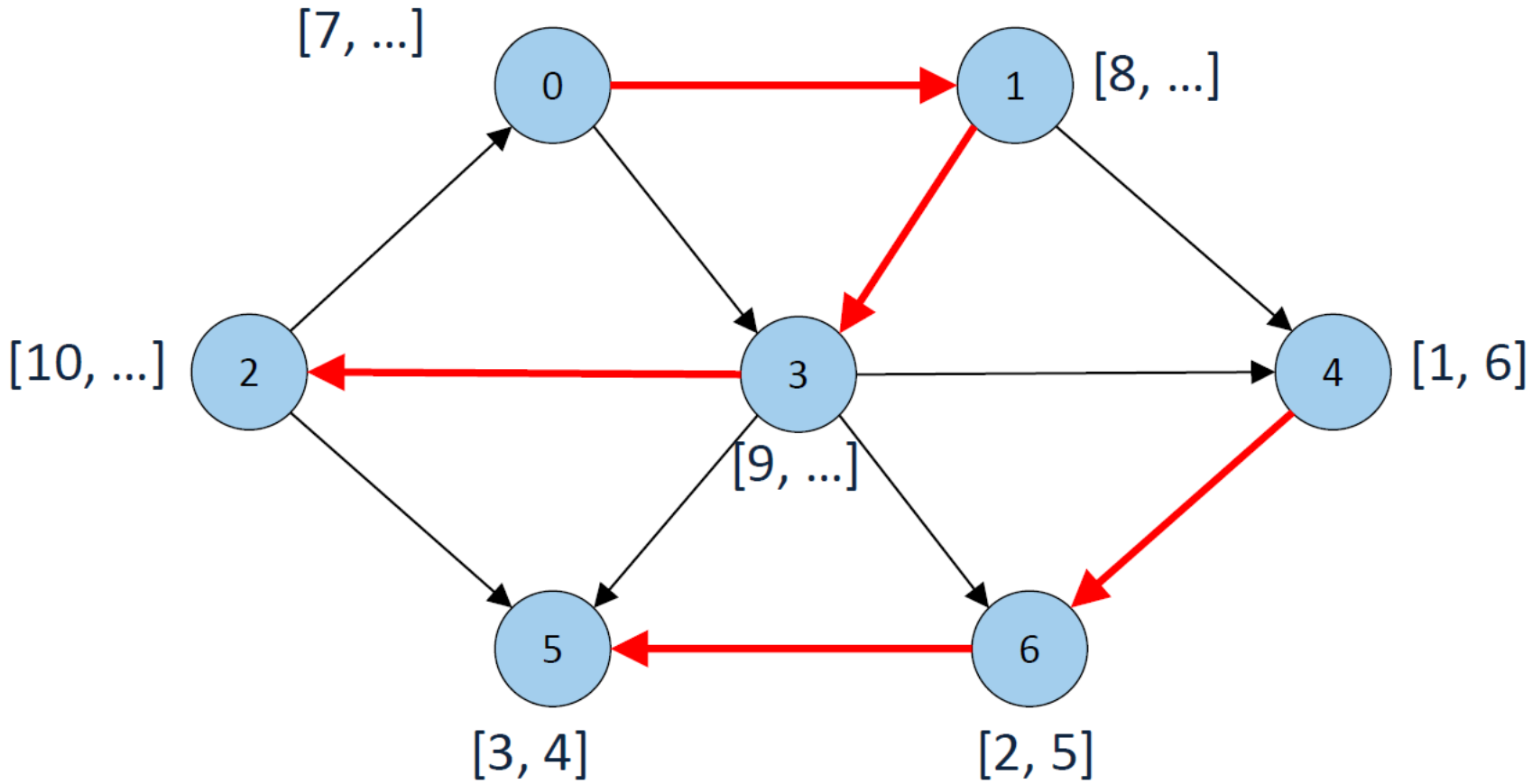
DFS_visit de G a partir del vértice 4



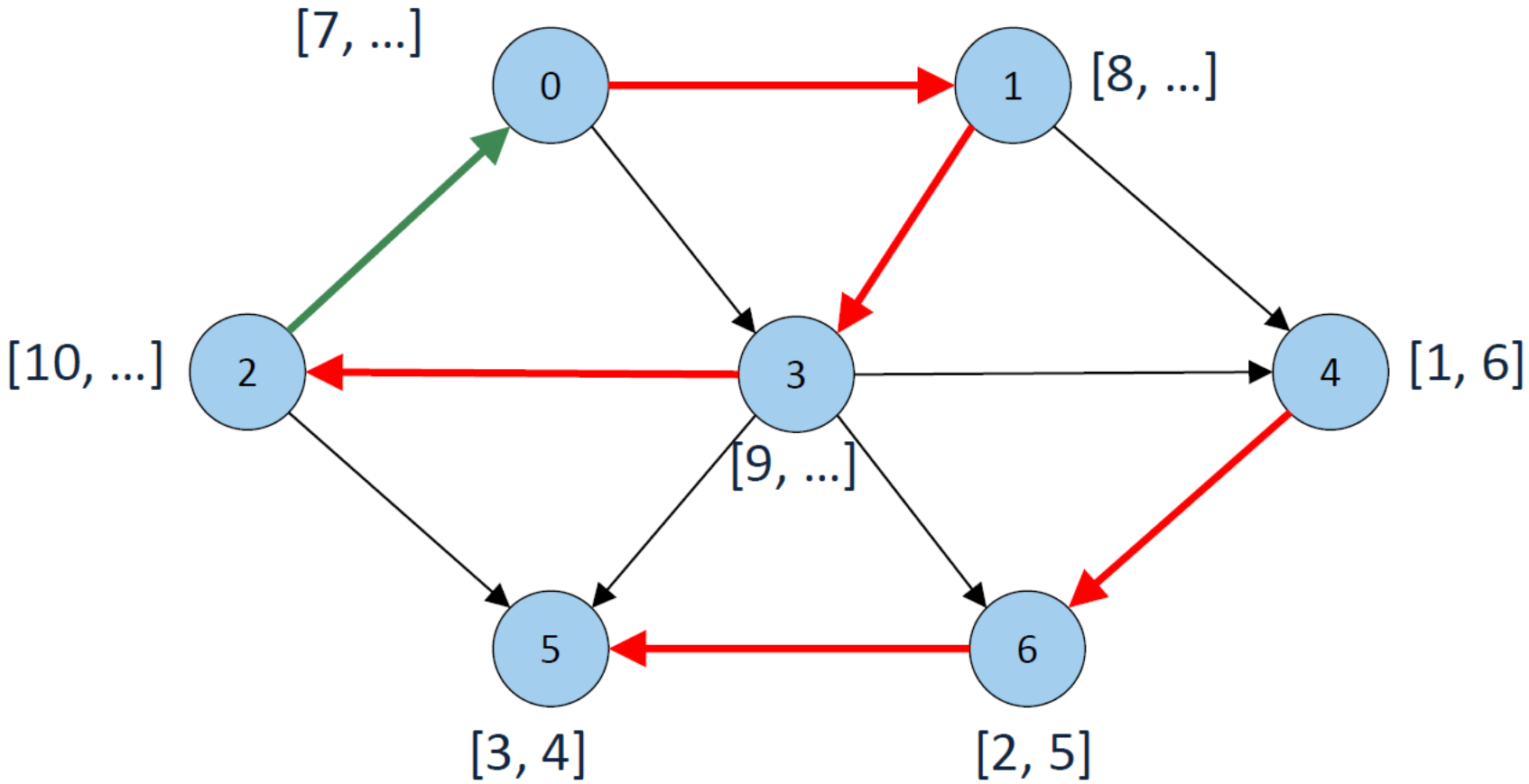
DFS_visit de G a partir del vértice 0



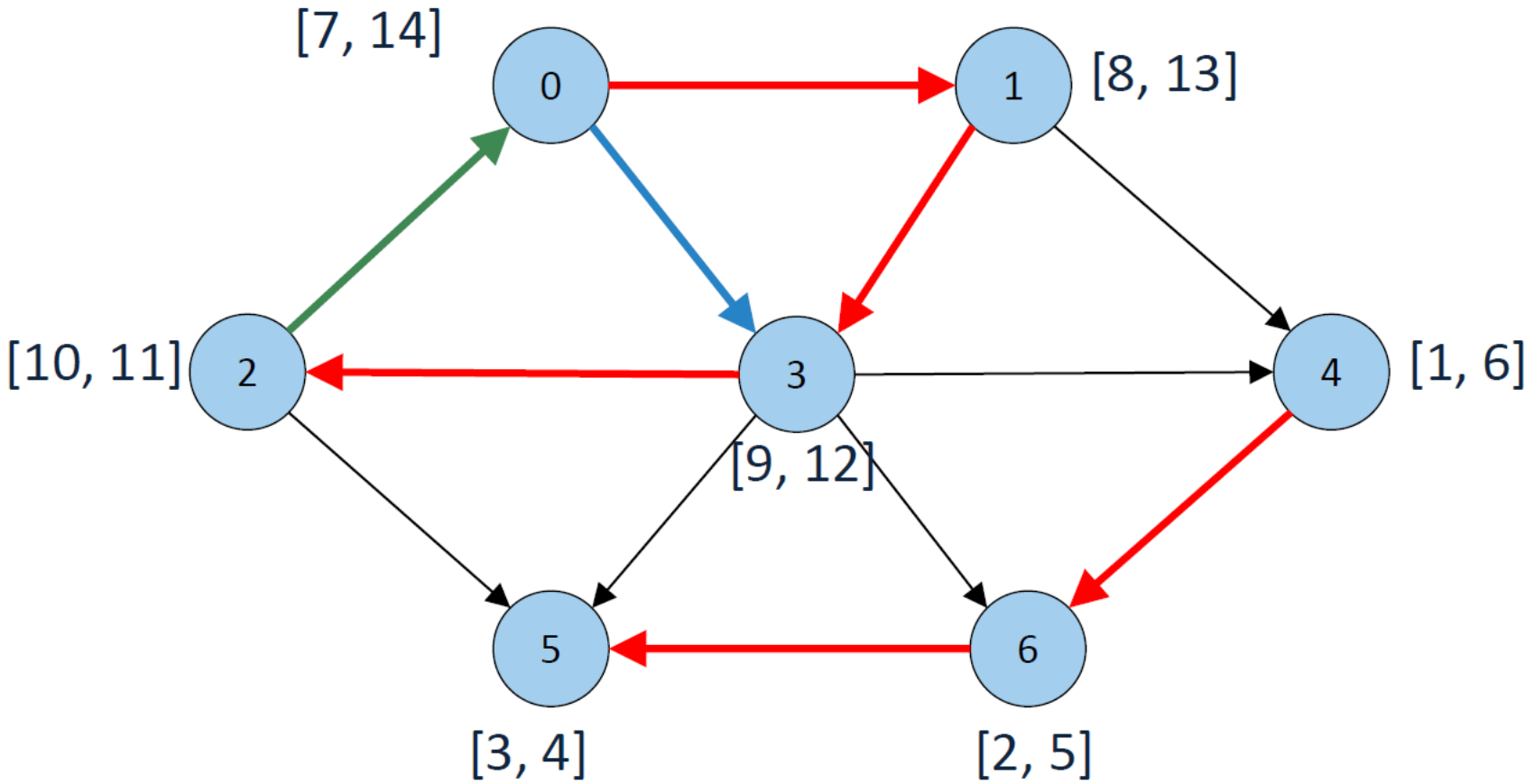
DFS_visit de G a partir del vértice 0



DFS_visit de G a partir del vértice 0



DFS_visit de G a partir del vértice 0



```
dfs(V, E):
    time = 1
    for each u in V:
        u.color = white
    for each u in V:
        if u.color == white:
            time = dfsVisit(u, time)
```

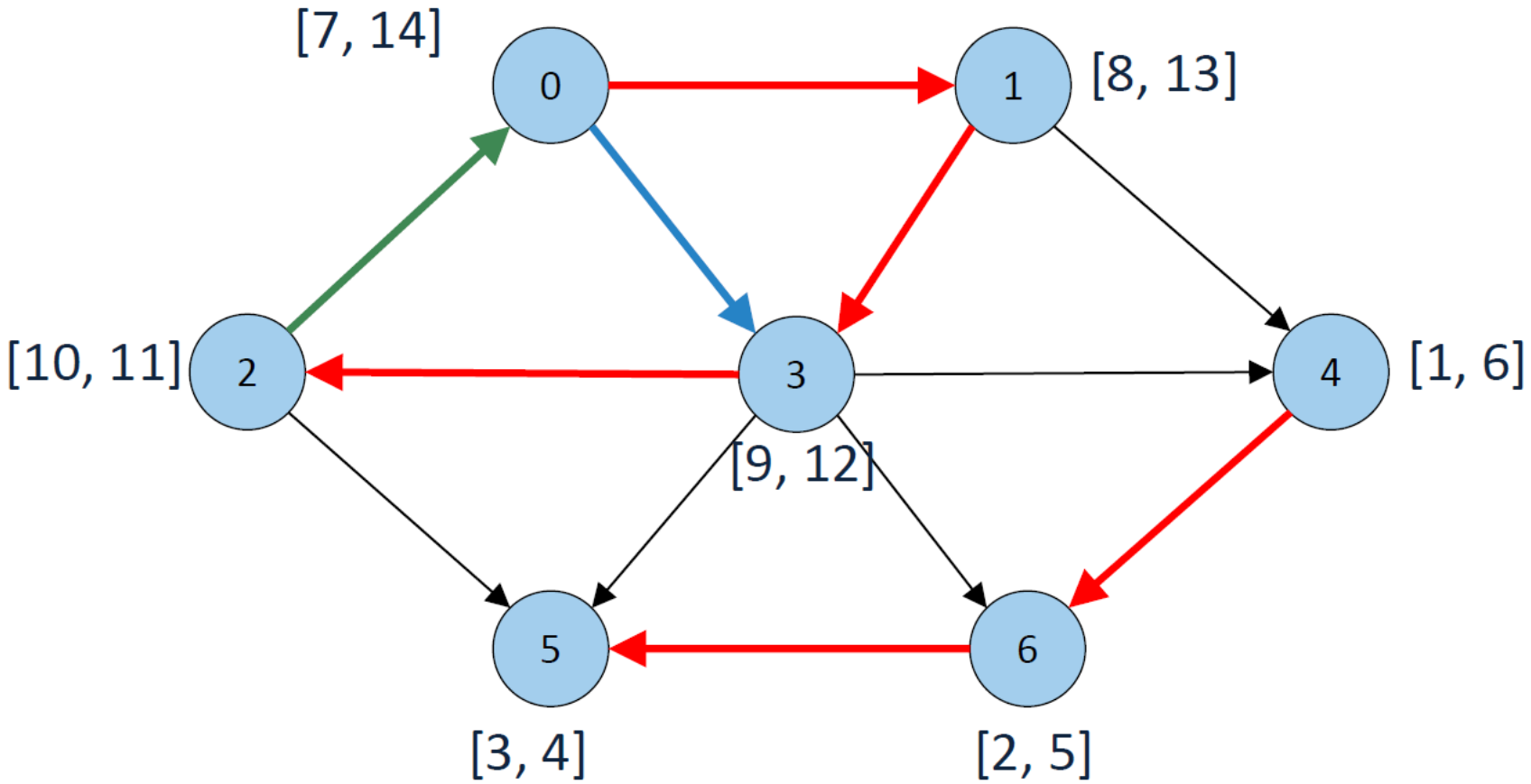
```
dfsVisit(u, time):
    u.color = gray
    u.start = time
    time += 1
    for each v in α[u]:
        if v.color == white:
            time = dfsVisit(v, time)
    u.color = black
    u.end = time
    time += 1
    return time
```

Intervalos de tiempo

Dados dos vértices *u* y *v,* sus intervalos cumplen una de las siguientes relaciones:

- [*u.star*, *u.end*] y [*v.start*, *v.end*] son *disjuntos*, y ni *u* ni *v* es descendiente del otro en el bosque DFS
- [*u.start*, *u.end*] *está contenido* en el intervalo [*v.start*, *v.end*], y *u es descendiente de v* en un árbol DFS
- [*v.start*, *v.end*] *está contenido* en el intervalo [*u.start*, *u.end*], y *v es descendiente de u* en un árbol DFS

DFS_visit de G a partir del vértice 0



Tipos de aristas luego de DFS

Aristas de árbol: la arista (u, v) es una arista de árbol si v fue descubierto por primera vez al explorar (u, v)

Aristas hacia atrás: aristas (u, v) que conectan un nodo u a un ancestro v en un árbol DFS:

- **el grafo es acíclico si y solo si DFS no produce aristas hacia atrás**

Aristas hacia adelante: aristas (u, v) que no son de árbol y conectan un nodo u a un descendiente v en un árbol DFS; *no aparecen en grafos no direccionales*

Aristas cruzadas: todas las otras aristas; *no aparecen en grafos no direccionales*

¿Qué usos le podemos dar a DFS + los tiempos de (inicio y) fin?

En grafos acíclicos: ordenación topológica

En grafos con ciclos: componentes fuertemente conectadas

Orden topológico

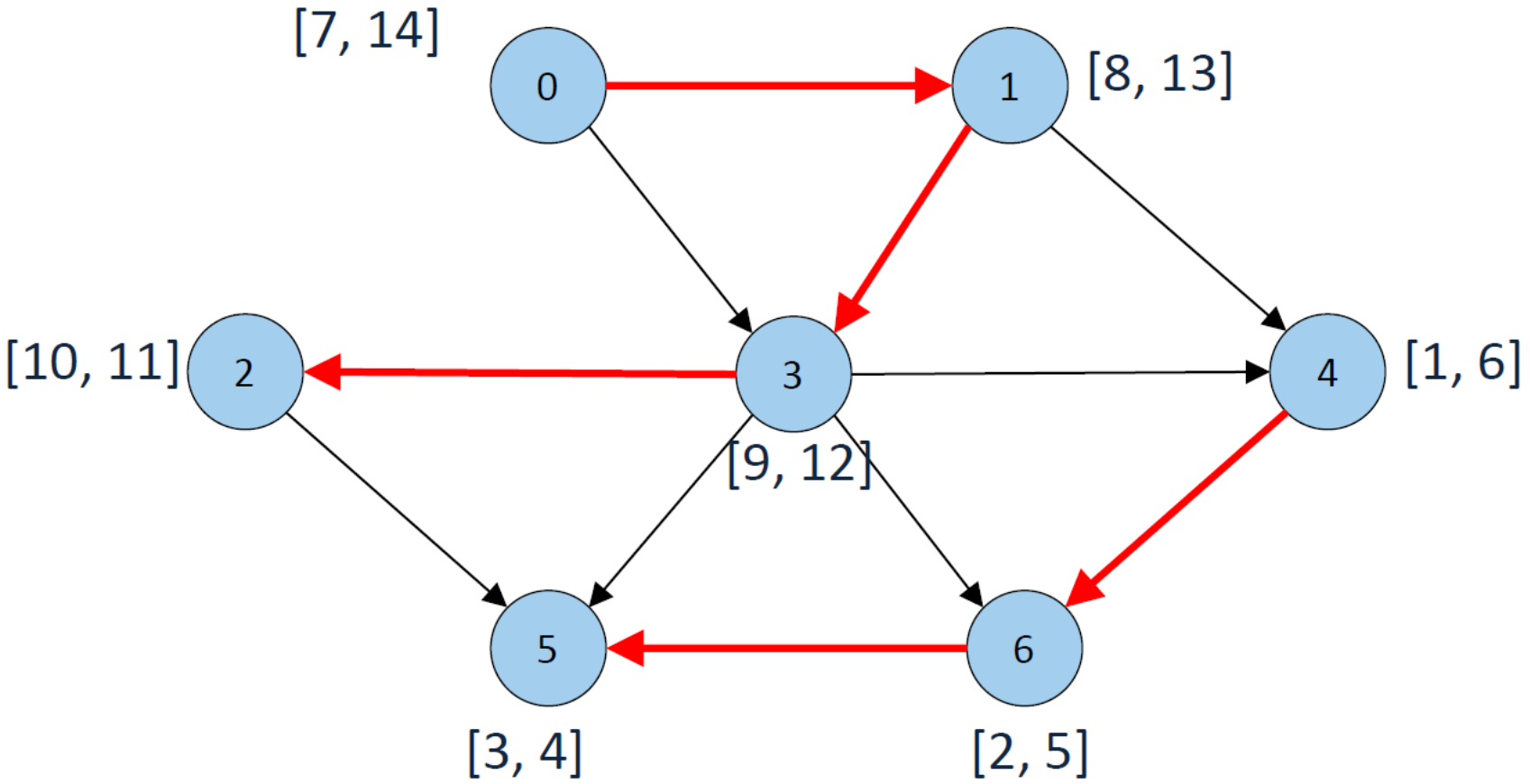
Un grafo acíclico se puede ordenar topológicamente

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

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

Si el grafo tiene ciclos, entonces no existe un orden topológico válido

Ejemplo: grafo después de ejecutar DFS



Lista *L*: 0 1 3 2 4 6 5

El algoritmo de ordenación topológica

$\boldsymbol{topoSort(G)}$

Crear lista $\boldsymbol{L}$ vacía

Ejecutar $\boldsymbol{dfs(G)}$ con tiempos

Insertar nodos en $\boldsymbol{L}$ en orden descendiente de tiempos $\boldsymbol{end}$

$\boldsymbol{return\ L}$

El algoritmo de ordenación topológica

$topoSort(G)$

Crear lista L vacía

Ejecutar $dfs(G)$ con tiempos:

- cada vez que calculamos el tiempo end para un nodo, insertamos ese nodo al frente de L

$return\ L$

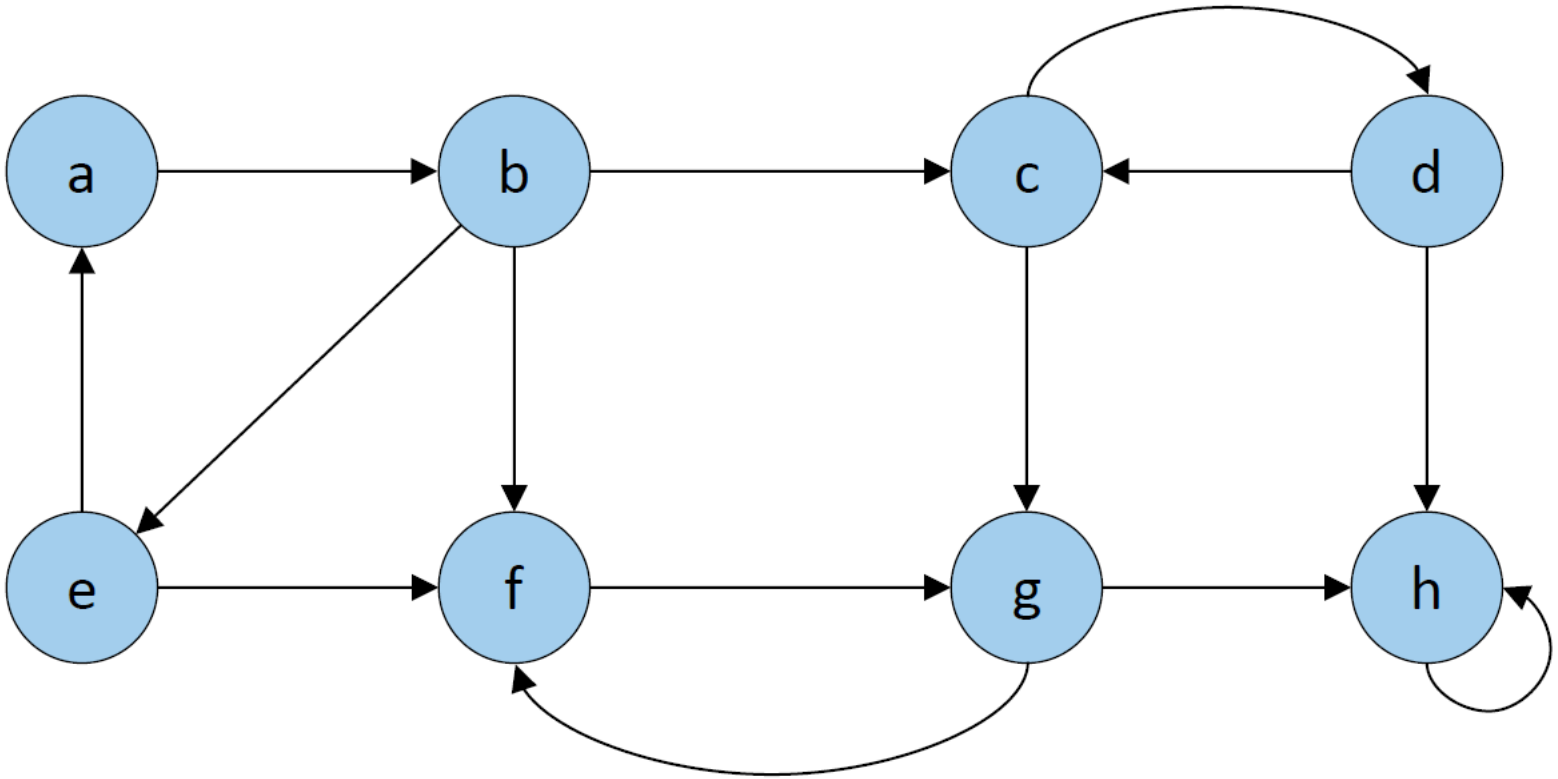
Grafo con ciclos y **CFCs**

En un grafo con ciclos no es posible encontrar un orden topológico ya que dos nodos de un ciclo pueden alcanzarse mutuamente

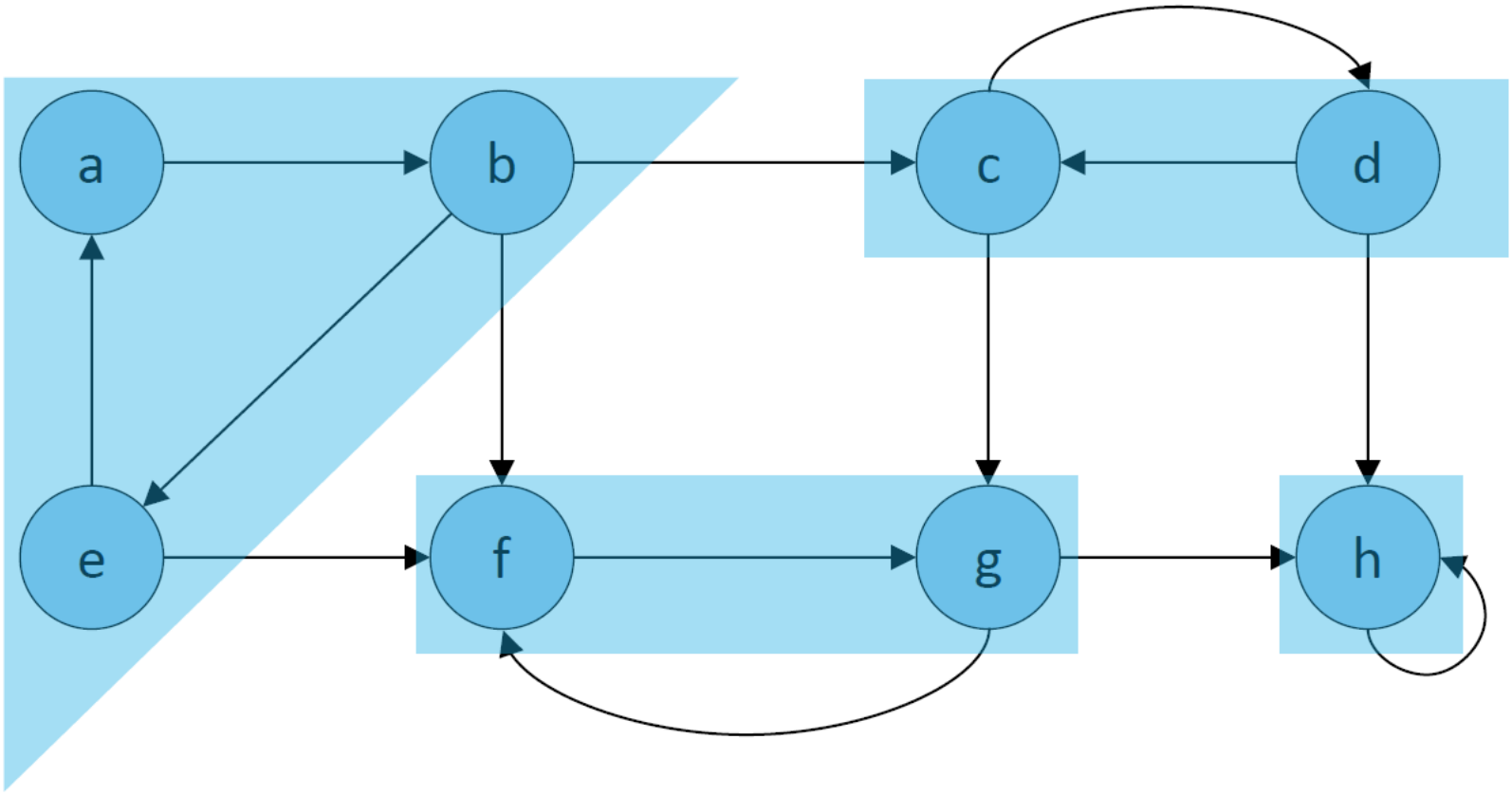
Los nodos de un grafo que se pueden alcanzar mutuamente son miembros de una misma *componente fuertemente conectada* (CFC) del grafo

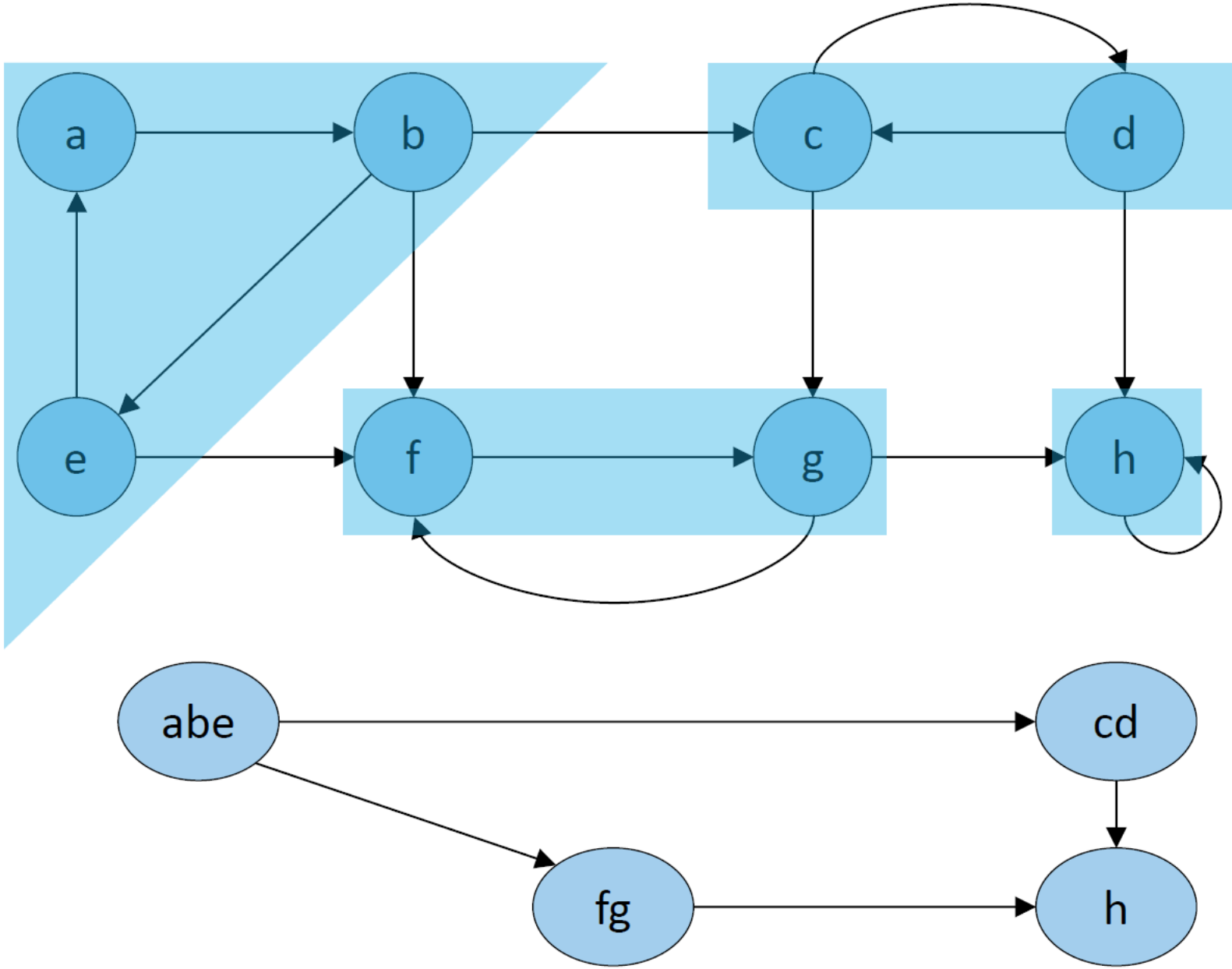
Las CFCs de un grafo direccional G son conjuntos máximos de nodos $C \subseteq V$ tales que para todo par de nodos u y v en C, u y v son mutuamente alcanzables

Ejemplo de de grado con ciclos



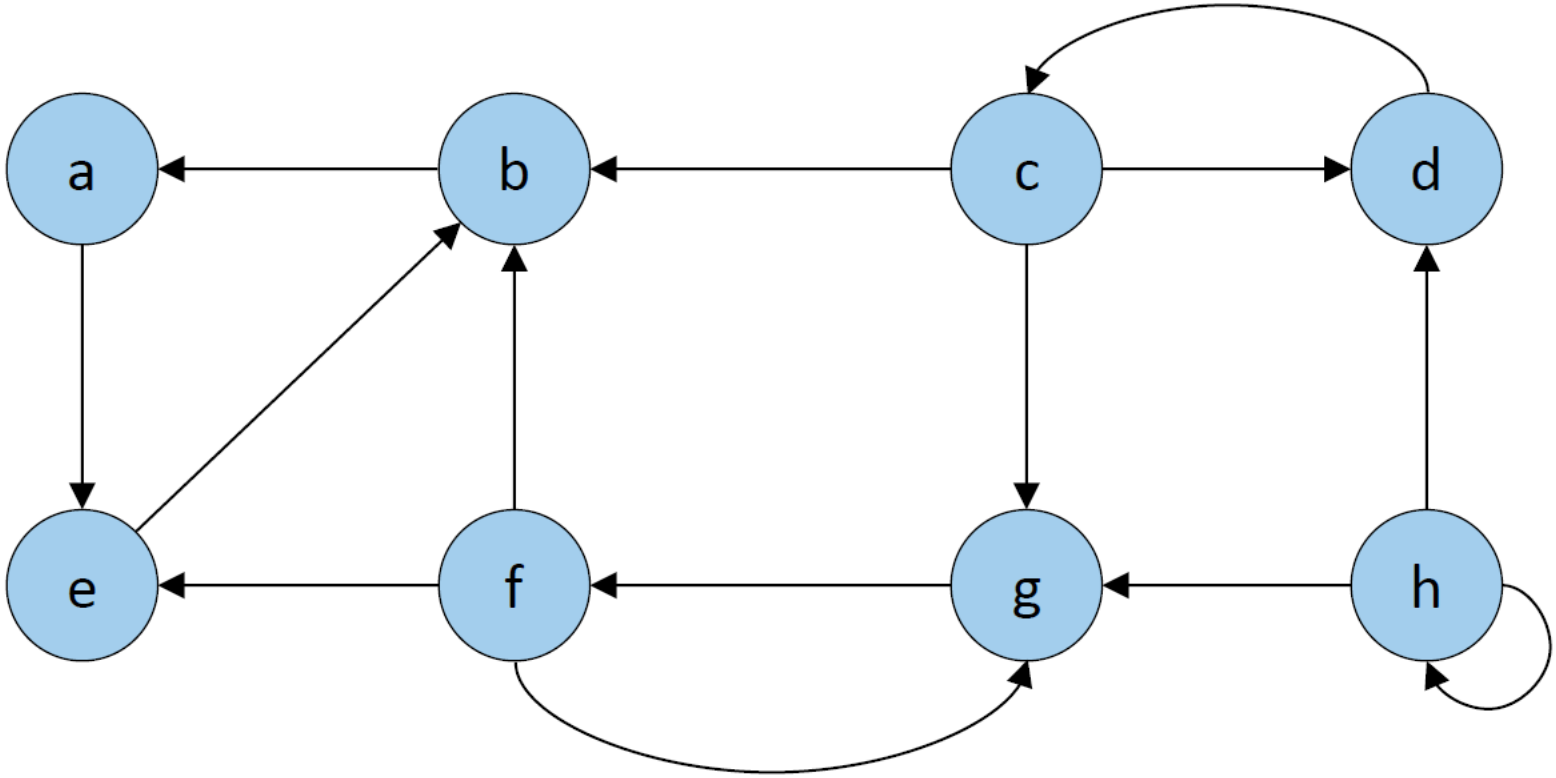
CFCs del grado anterior




a
b
c
d
e
f
g
h
abe
cd
fg
h

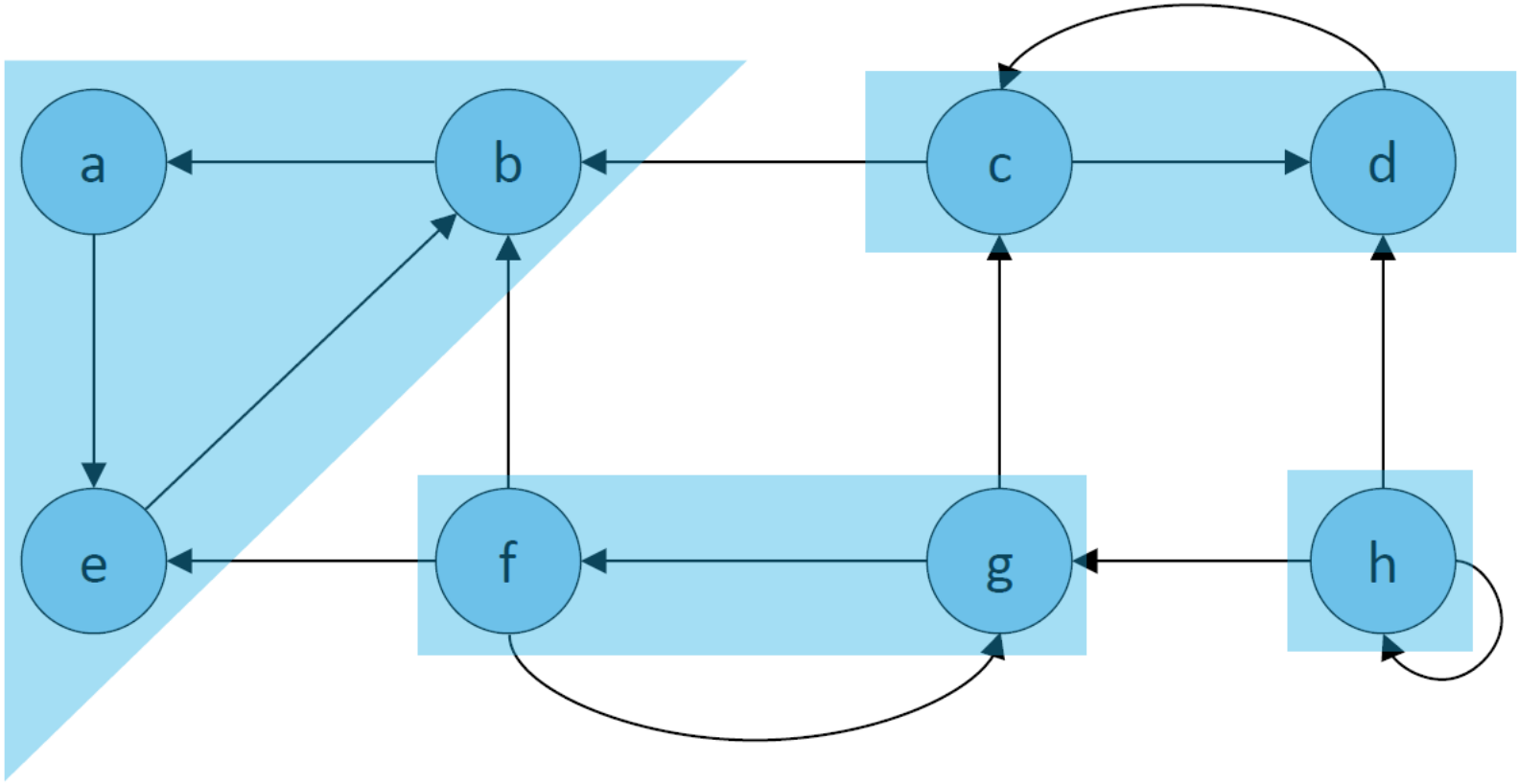
Grafo transpuesto, *G'*

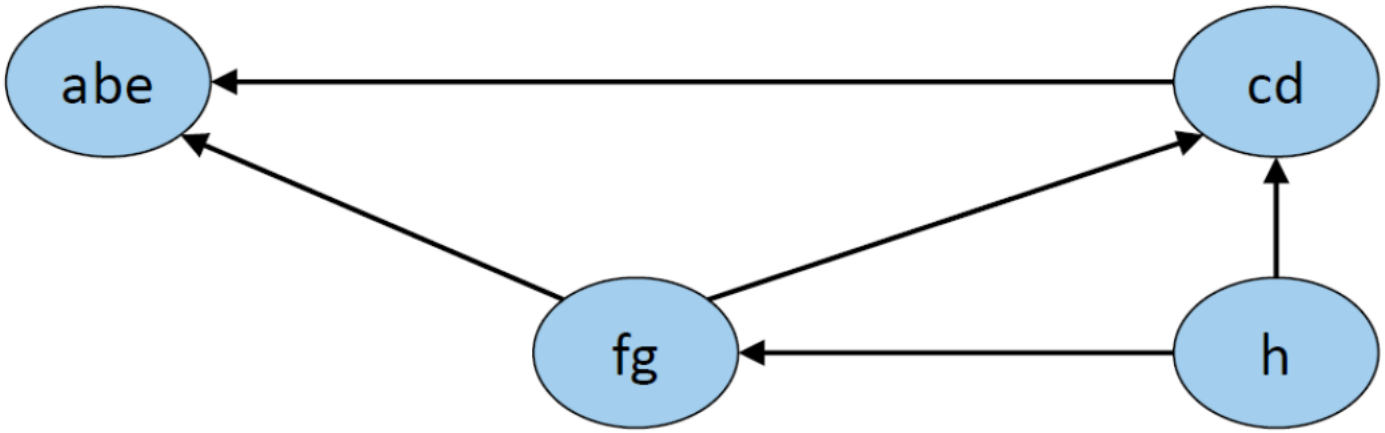
Para este algoritmo necesitaremos el grafo transpuesto *G'* de *G* en el cual damos vuelta las aristas: sea $\alpha'[\boldsymbol{u}]$ la lista de aristas de $\boldsymbol{u}$ en *G'*



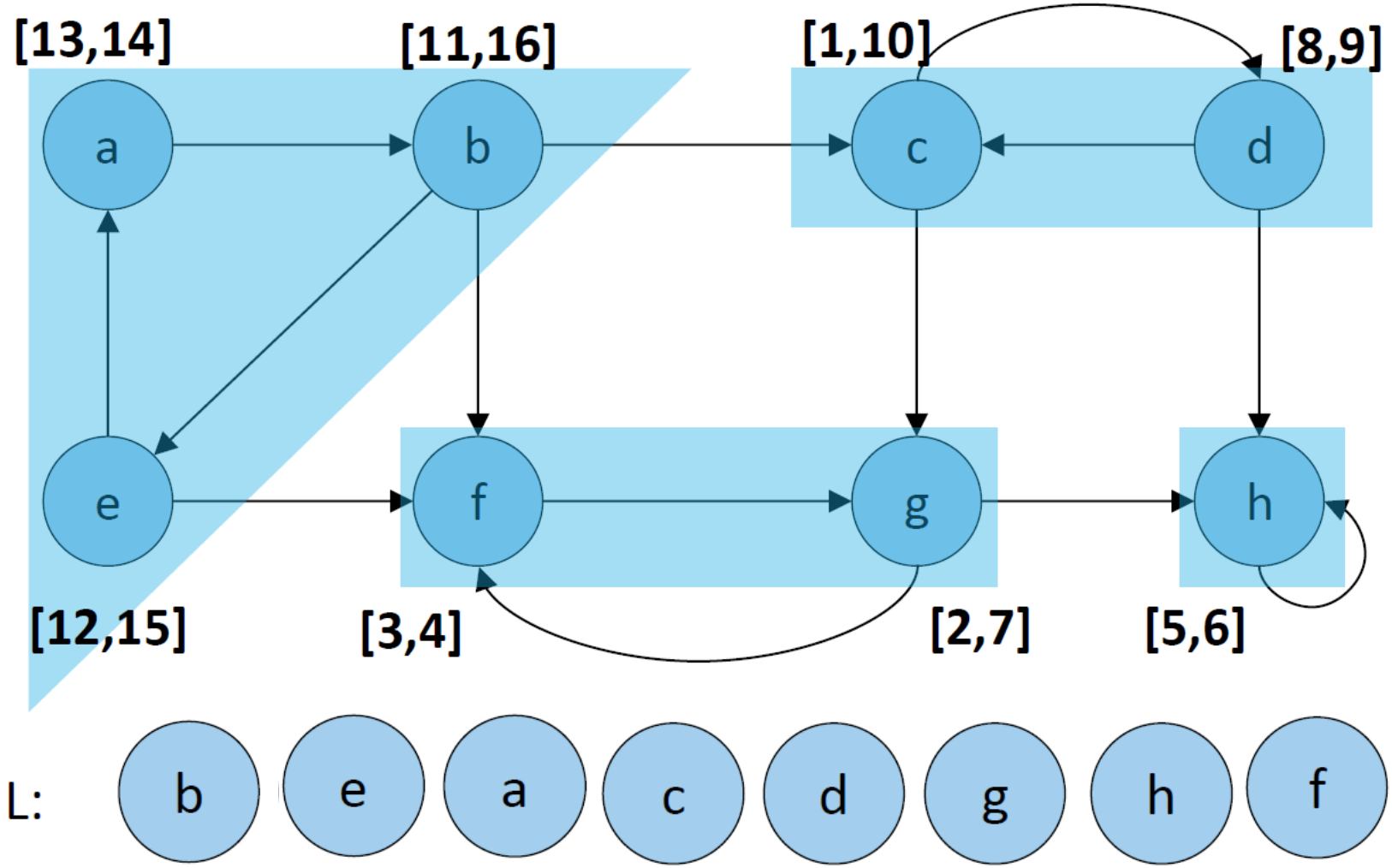
Grafo transpuesto, *G'*, tiene las mismas CFCs que *G*

El grafo *G'* tiene las mismas CFCs que *G*

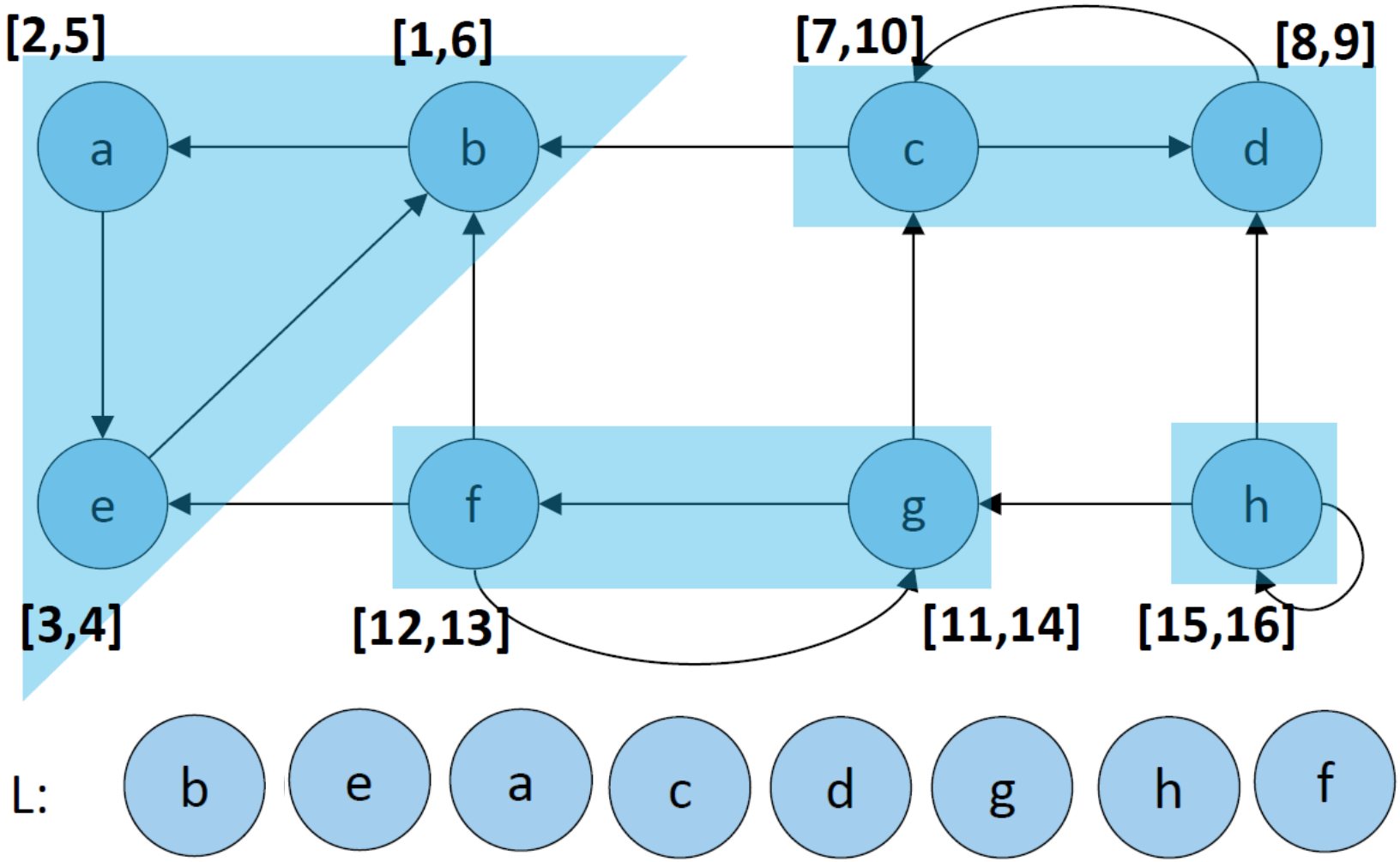



abe
cd
fg
h

Hagamos un recorrido DFS de *G*, anotando los tiempo de finalización de cada nodo



DFS sobre *G'*, pero en orden decreciente de tiempos ***end*** (según el recorrido anterior)



Algoritmo de Kosaraju para CFCs

Cada CFC tiene un nodo ***representante:***

si el representante de dos nodos es el mismo, entonces los nodos pertenecen a la misma CFC

$$assign(u, rep):$$

$$\quad if\ u.rep = \emptyset:$$

$$\quad\quad u.rep = rep$$

$$\quad\quad foreach\ v\ in\ \alpha'[u]:$$

$$\quad\quad\quad assign(v, rep)$$

Algoritmo de Kosaraju para CFCs

$kosaraju(G)$

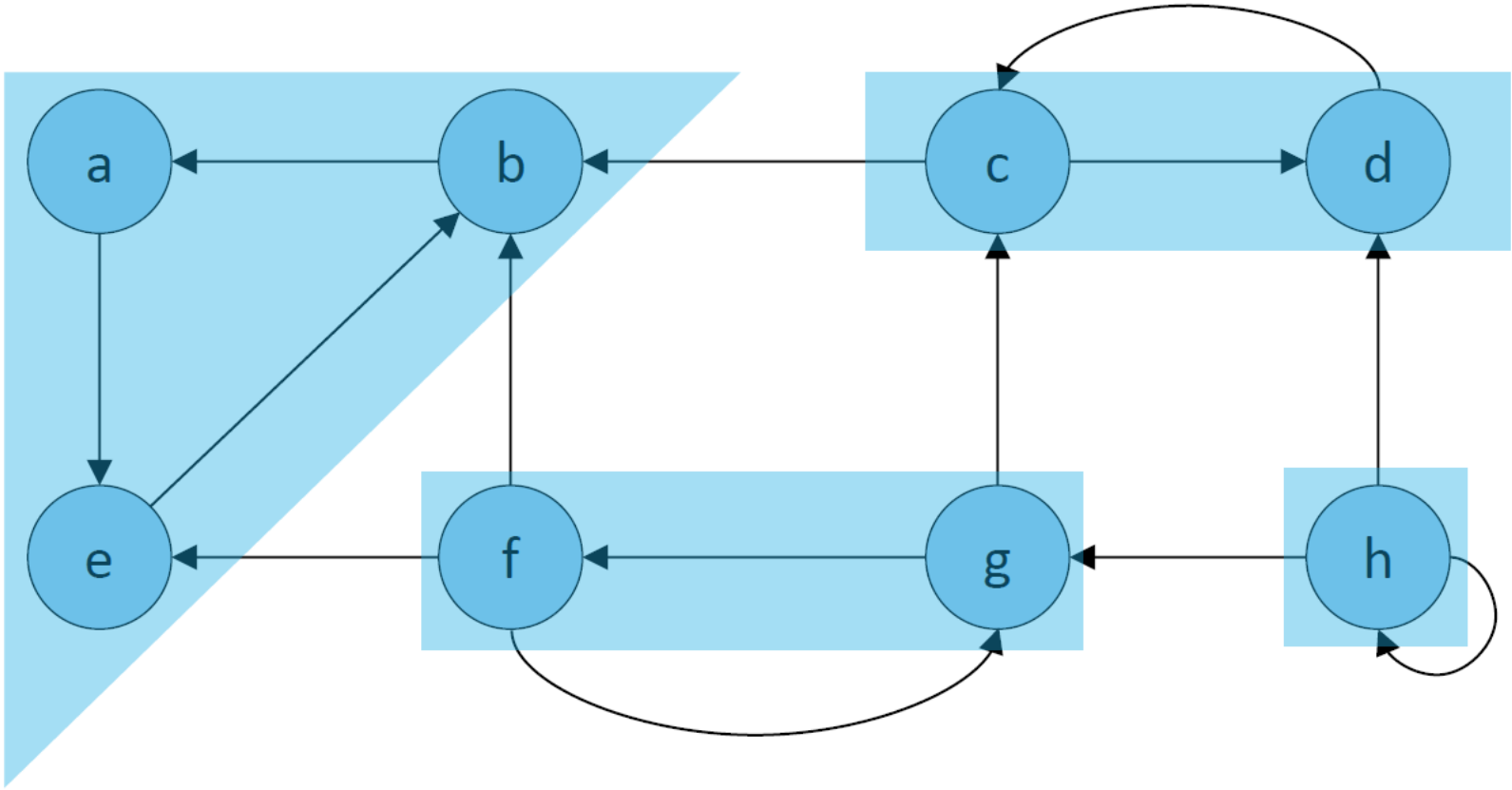
Crear lista L vacía

Ejecutar $dfs(G)$ con tiempos

Insertar vértices en L en orden descendiente de tiempos f

$for\ each\ u\ in\ L$:

$assign(u, u)$


a
b
c
d
e
f
g
h

Definamos el *grafo de componentes* G^{CFC}

Supongamos que G tiene las componentes fuertemente conectadas C_1, C_2, …, C_k

V^{CFC} es $\{C_1, C_2, \ldots, C_k\}$

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

El grafo G^{CFC} tiene un orden topológico

G^{CFC} es un grafo direccional acíclico

Esto, ya que si existiera un ciclo en G^{CFC}, este tendría CFCs, lo cual no es posible por construcción del grafo

Por lo que podemos encontrar un orden topológico en G^{CFC}

Resumen

- Podemos guardar los tiempos de inicio y fin de cada nodo al hacer DFS
- Usando los tiempos podemos encontrar un orden topológico en un grafo acíclico
- En un grafo cíclico podemos encontrar las componentes fuertemente conectadas
- Podemos encontrar el orden topológico de las componentes fuertemente conectadas en un grafo cualquiera