# Componentes Fuertemente Conexas

Bibliografía: "Introduction to Algorithms". Third Edition.

The MIT Press. Massachusetts Institute of Technology. Cambridge, Massachusetts 02142.

http://mitpress.mit.edu

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein


**Facultad de Matemática y Computación**
**Universidad de La Habana**

# Determinar Componentes Fuertemente Conexas

**PROBLEMA**

Determinar las **componentes fuertemente conexas (*SCC*)** de un **grafo dirigido** *G=(V, E)*

- Varios algoritmos que trabajan con grafos dirigidos comienzan su ejecución, descomponiendo previamente el **Grafo** en sus **componentes fuertemente conexas**

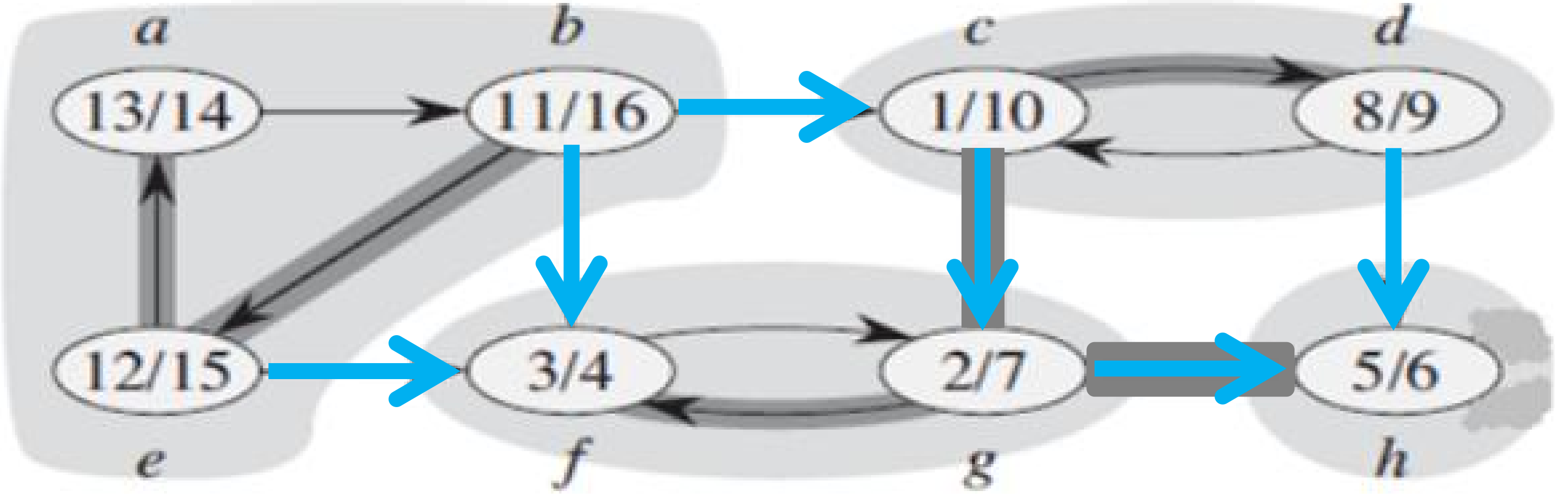
- Posteriormente, dichos algoritmos trabajan de manera independiente sobre cada una de ellas y, finalmente, combinan las soluciones en correspondencia con las conexiones existentes entre dichas componentes

# Una aplicación del DFS: determinar *CFC*

**El Algoritmo para descomponer un grafo dirigido en sus componentes fuertemente conexas, es una aplicación clásica del DFS**

En general, estas se determinan a partir de **dos recorridos *DFS***

Una **componente fuertemente conexa** de un ***grafo dirigido*** *G=(V, E)* es un **conjunto maximal de vértices**, $C \subseteq V$, que cumple que, para todo par de vértices $u$, $v \in C$, existe un camino de $u$ a $v$ y uno de $v$ a $u$, o sea, **hay caminos en ambas direcciones**




- Cada región sombreada es una **componente fuertemente conexa**

- Cada vértice está etiquetado con el **tiempo de descubrimiento** y el **tiempo de finalización** asignados en un DFS

- Los **arcos de árbol** son los que están sombreados

- **Todos los vértices de un grafo dirigido, están en alguna componente fuertemente conexa**, pero ciertos arcos pueden no estarlo: **arcos de cruce entre componentes**. Estos van, de un vértice de una componente fuerte a un vértice de otra componente fuerte

# Transpuesta de un Grafo Dirigido

## Definición

La **transpuesta de un grafo dirigid**o *G = (V, E)* es el Grafo $G^T = (V,\ E^T)$, donde, $E^T = \{(\boldsymbol{v},\ \boldsymbol{u}) \in VXV : (\boldsymbol{u},\ \boldsymbol{v}) \in E\}$

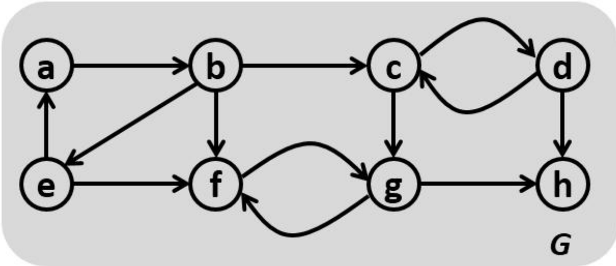
**$G^T$ es el grafo G con todos sus arcos invertidos**

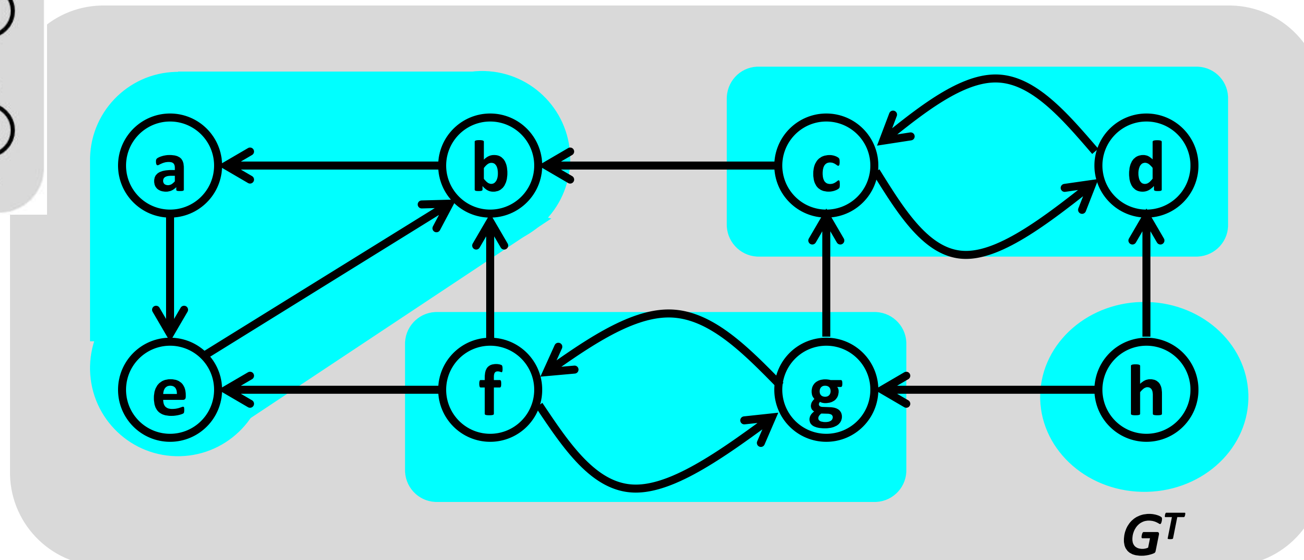
A partir de una representación por Listas de Adyacencia, obtener $G^T = (V,\ E^T)$, es *O(V+E)*

**¿Cómo hacerlo?**
***Ejercicio propuesto***

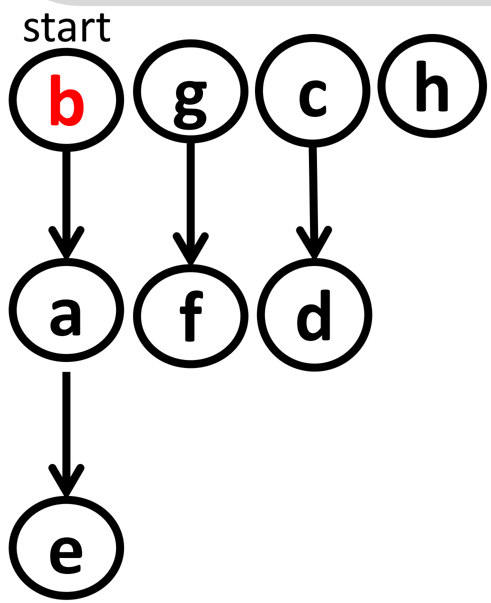
**Tanto *G* como $G^T$ tienen las mismas componentes fuertemente conexas:**

En *G* hay un camino, en ambas direcciones, entre u y v ⇔ en $G^T$ lo hay también

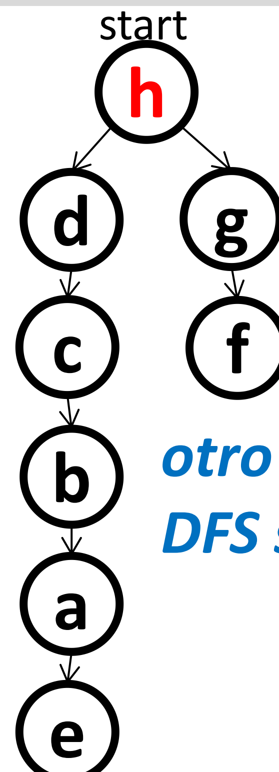



*Un posible DFS sobre $G^T$ :*




En este caso, las componentes fuertemente conexas de G coinciden con los árboles del bosque abracador que se forma tras este DFS




*otro posible DFS sobre $G^T$*

# Grafo Reducido de un Grafo Dirigido

Las interconexiones entre los **componentes fuertemente conexas** de un grafo dirigido **G** se pueden representar construyendo un **Grafo Reducido de *G***

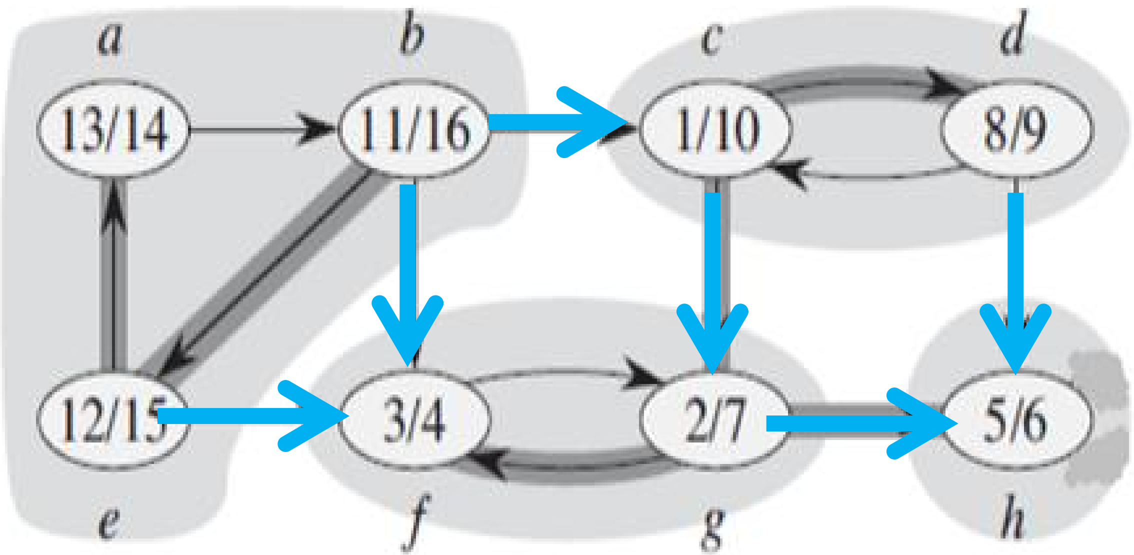
**Definición:**

Se le llama **grafo reducido de G** al grafo $G^{SCC}=(V^{SCC}, E^{SCC})$, que se define de la siguiente forma:

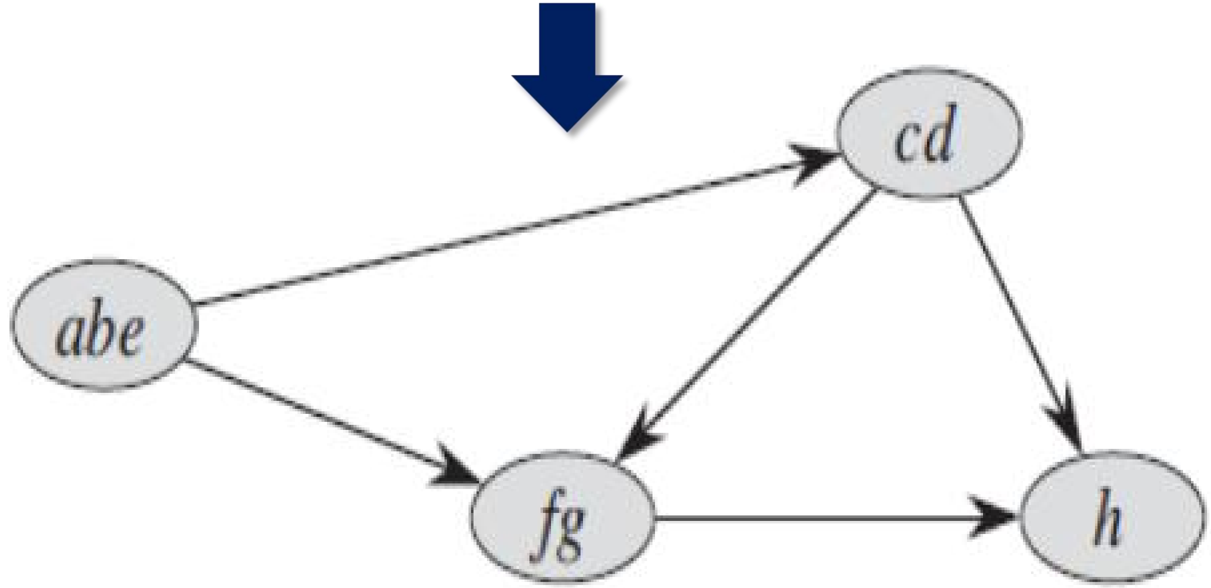
- Sean $C_1, C_2, \ldots, C_k$ las componentes fuertes de $G$
- Sea $V^{SCC} = \{v_1, v_2, \ldots, v_k\}$ el conjunto de vértices de $G^{SCC}$. En $V^{SCC}$ hay un vértice $v_i$ por cada componente fuerte $C_i$ de $G$
- $(v_i, v_j) \in E^{SCC}$ si en $G$ EXISTE, al menos, un arco $(x, y) \in E$ **para algún** $x \in C_i$ y **para algún** $y \in C_j$

A este grafo suele llamársele también **grafo de componentes**

# Ejemplo de Grafo Reducido




*G=(V, E)*




Grafo reducido

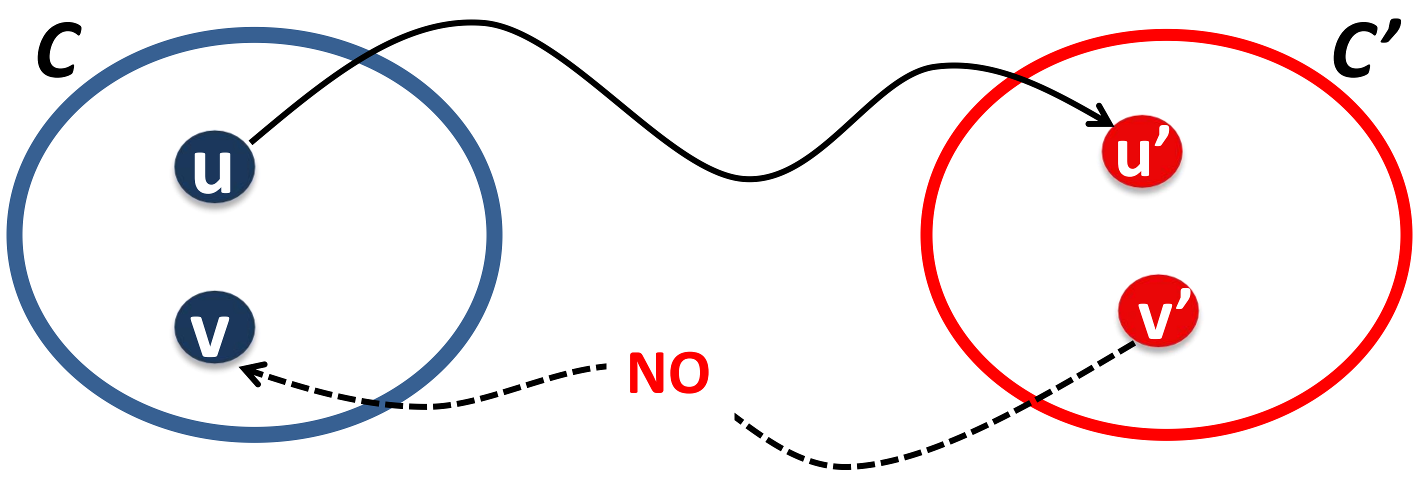
$G^{SCC}=(V^{SCC}, E^{SCC})$

# Propiedades del Grafo Reducido

El **grafo reducido de G es un DAG, o sea,** es dirigido y **acíclico,** lo cual se demuestra a partir del siguiente **Lema**

***Lema 1***

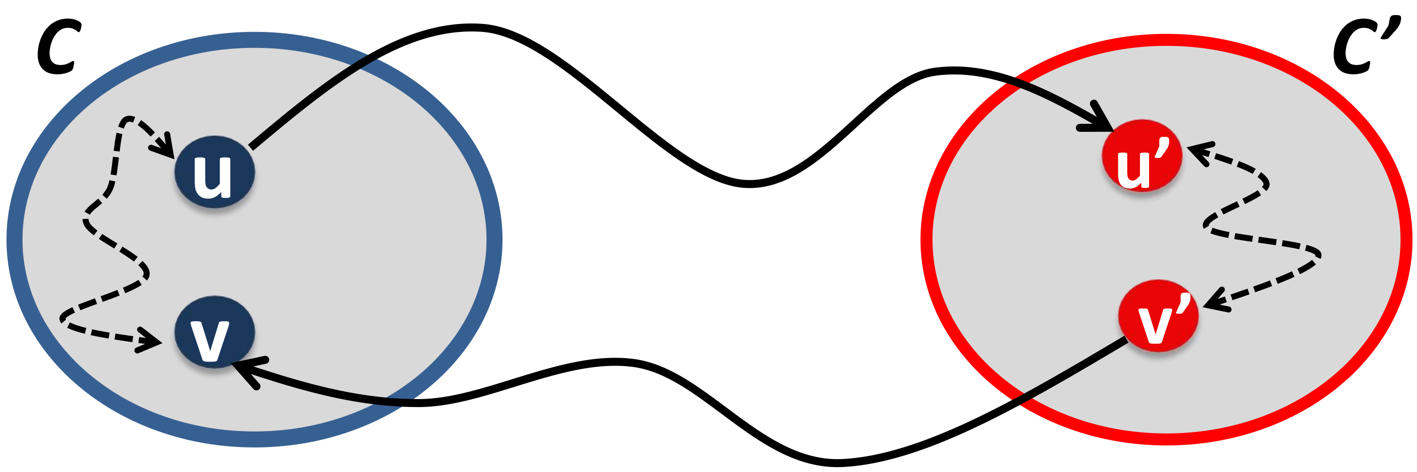
Sean *C* y *C'* dos **componentes fuertemente conexas** diferentes del grafo dirigido ***G****=(V, E)* y sean *u, v* dos vértices en *C* y sean *u', v'* dos vértices en *C'* y supongamos que existe un camino de *u* a *u'* en *G*
$\Rightarrow$ NO PUEDE haber un camino de *v'* a *v* en *G*



# Propiedades del Grafo Reducido

*Lema 1* - **Demostración:**

Si hubiera un camino de *v'* a *v* en *G,* entonces habría camino de ($u \sim u' \sim v'$) y de( $v' \sim v \sim u$) y por tanto *u* y *v'* serían alcanzables entre si, y en ambas direcciones, en *G,* lo cual contradice que *C* y *C'* sean dos componentes fuertes **diferentes** de *G*

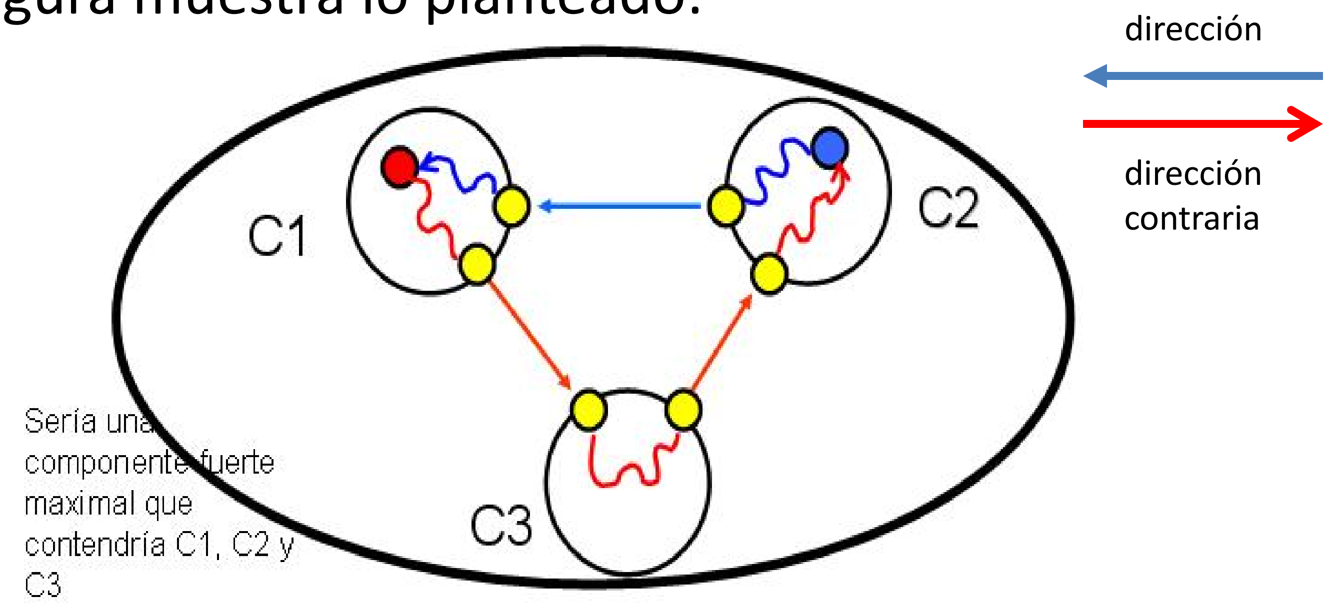


# Propiedades del Grafo Reducido

El planteamiento del ***Lema 1*** podría interpretarse también de la siguiente forma:

Si en el **grafo reducido** hubiera un ciclo, entonces, las componentes fuertes implicadas en el mismo **NO fueran maximales**, pues desde cada vértice de alguna de ellas, se puede ir a un vértice de cualquiera de las otras, en ambas direcciones

La siguiente figura muestra lo planteado:

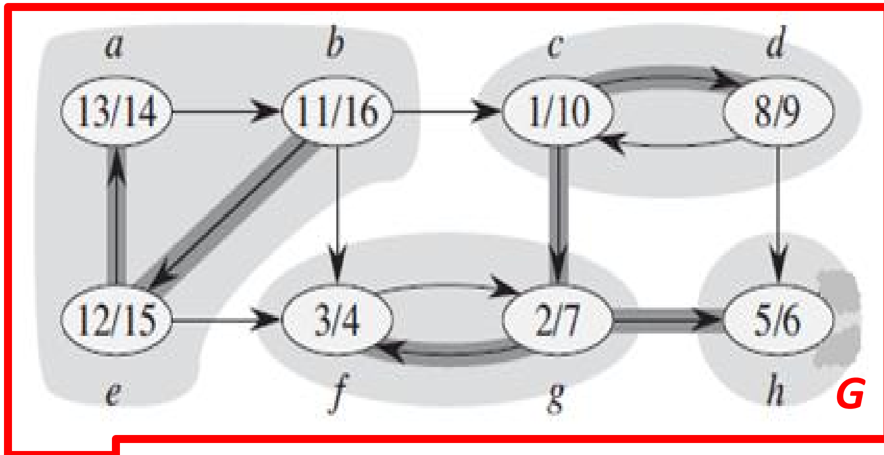


# Usando DFS para hallar las componentes fuertemente conexas de un grafo dirigido G

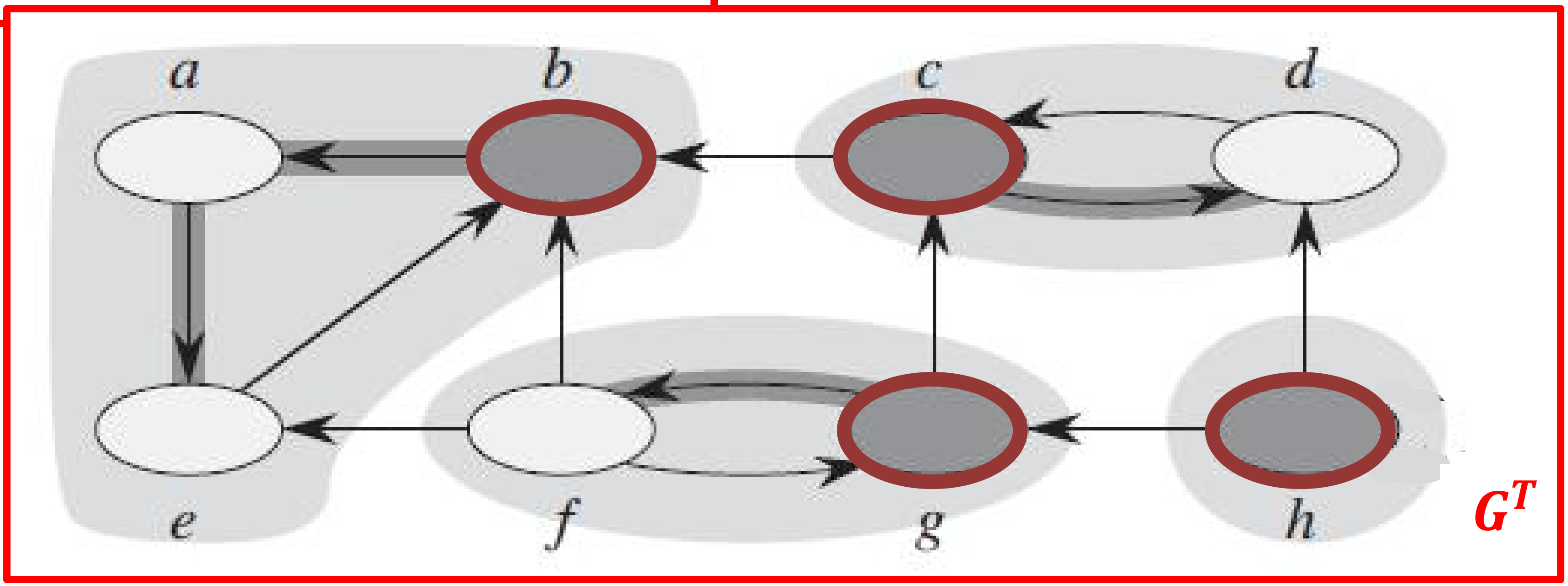
Propiedad fundamental en la cual se basa el algoritmo: El **grafo reducido de G es un DAG, o sea,** es ***dirigido*** y ***acíclico***

**STRONGLY-CONNECTED-COMPONENTS (*G*)**

**1.** DFS (*G*) para calcular $f[u]$ $\forall u \in$ V
**2.** Determinar $G^T$
**3.** DFS ($G^T$), comenzando por el vértice *u* de mayor *f*[*u*] (Si la búsqueda en profundidad no llega a todos los vértices, iníciese la siguiente búsqueda a partir del **vértice blanco** de mayor *f[u]*)
**4.** Cada árbol del bosque abarcador resultante es una **componente fuertemente conexa** de *G*




***b*, *c*, *g* y *h***, son las raíces de los árboles del bosque abarcador




**Tras aplicar DFS sobre $G^T$ en orden descendiente con respecto a f[], cada árbol del bosque abarcador primero en profundidad que se obtiene , se corresponde** con una **componente fuertemente conexa de G**

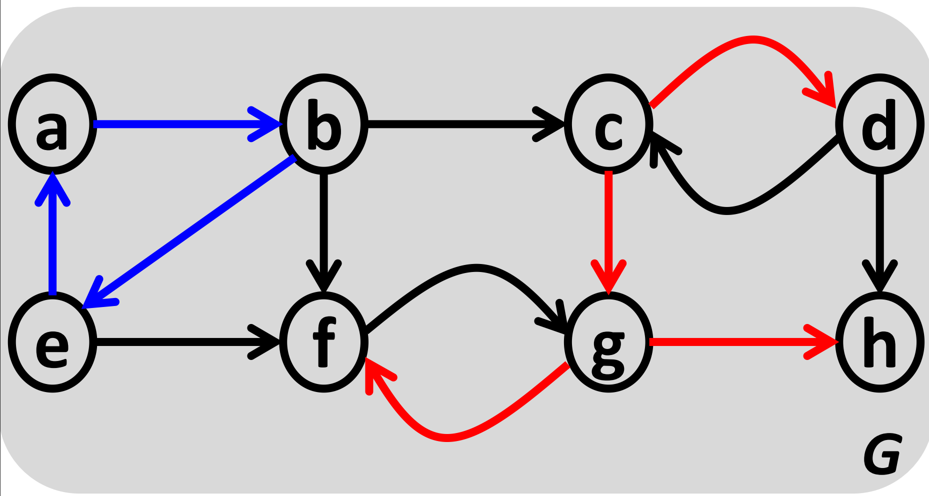
```
STRONGLY-CONNECTED-COMPONENTS (G)
   S = DFS_SCC_1(G)
   Calcular G^T
   DFS_SCC_2(G^T,S)
```

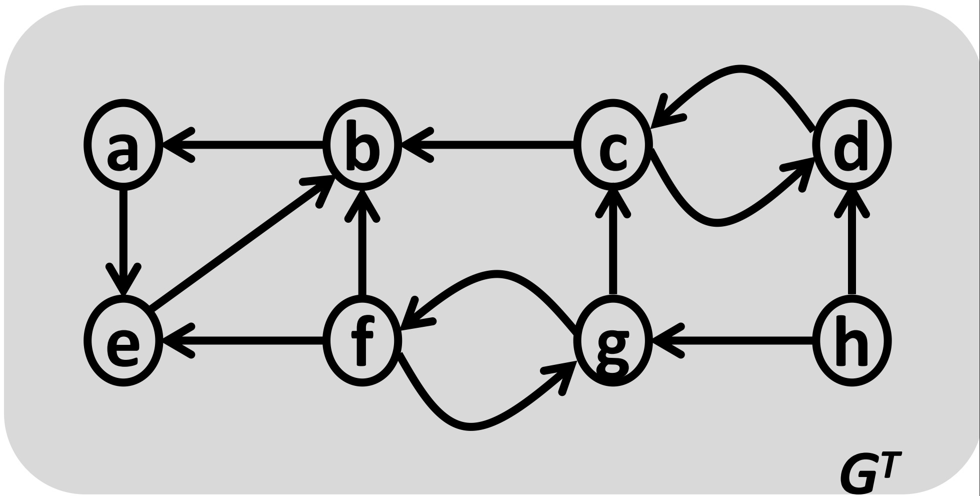
```
DFS_SCC_2(G, S)
    for each vertex u ∈ G.V
        u.color = WHITE
    c = 0;
    while S.Count > 0
        u = S.Pop() //O(1)
        if u.color == WHITE
            DFS-VISIT-2(G,u,c)
            c = c+1
```

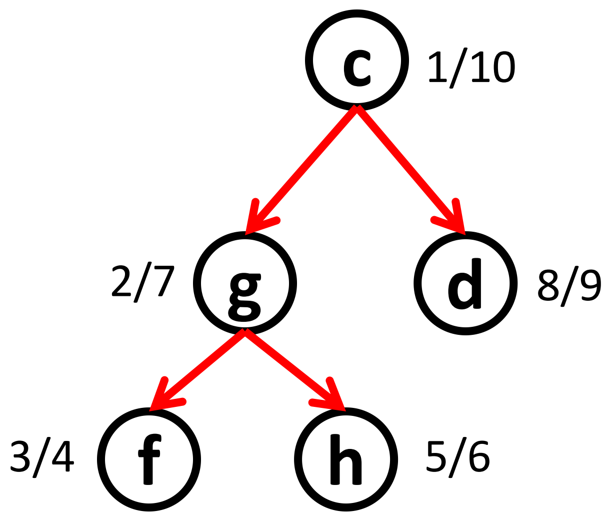
```
DFS-VISIT-2(G,u,c)
    u.color = GRAY
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            DFS-VISIT-2(G,v,c)
    u.color = BLACK
    u.CC = c
```

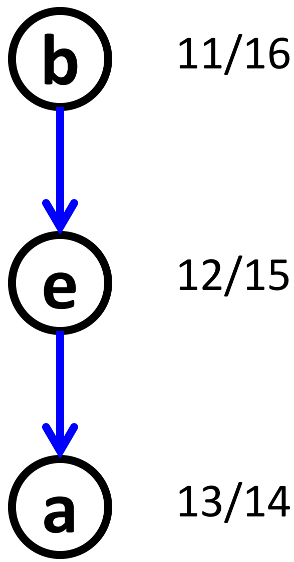
```
DFS_SCC_1(G)
    for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
    time = 0
    Stack S = new Stack();
    for each vertex u ∈ G.V
        if u.color == WHITE
            DFS-VISIT-1(G,u,S)
    return S;
```

```
DFS-VISIT-1(G,u,S)
  time = time + 1
  u.d = time
  u.color = GRAY
  for each v ∈ G.Adj[u]
      if v.color == WHITE
          v.π = u
          DFS-VISIT-1(G,v,S)
  u.color = BLACK
  time = time + 1
  S.Push(u) //O(1)
  u.f = time
```



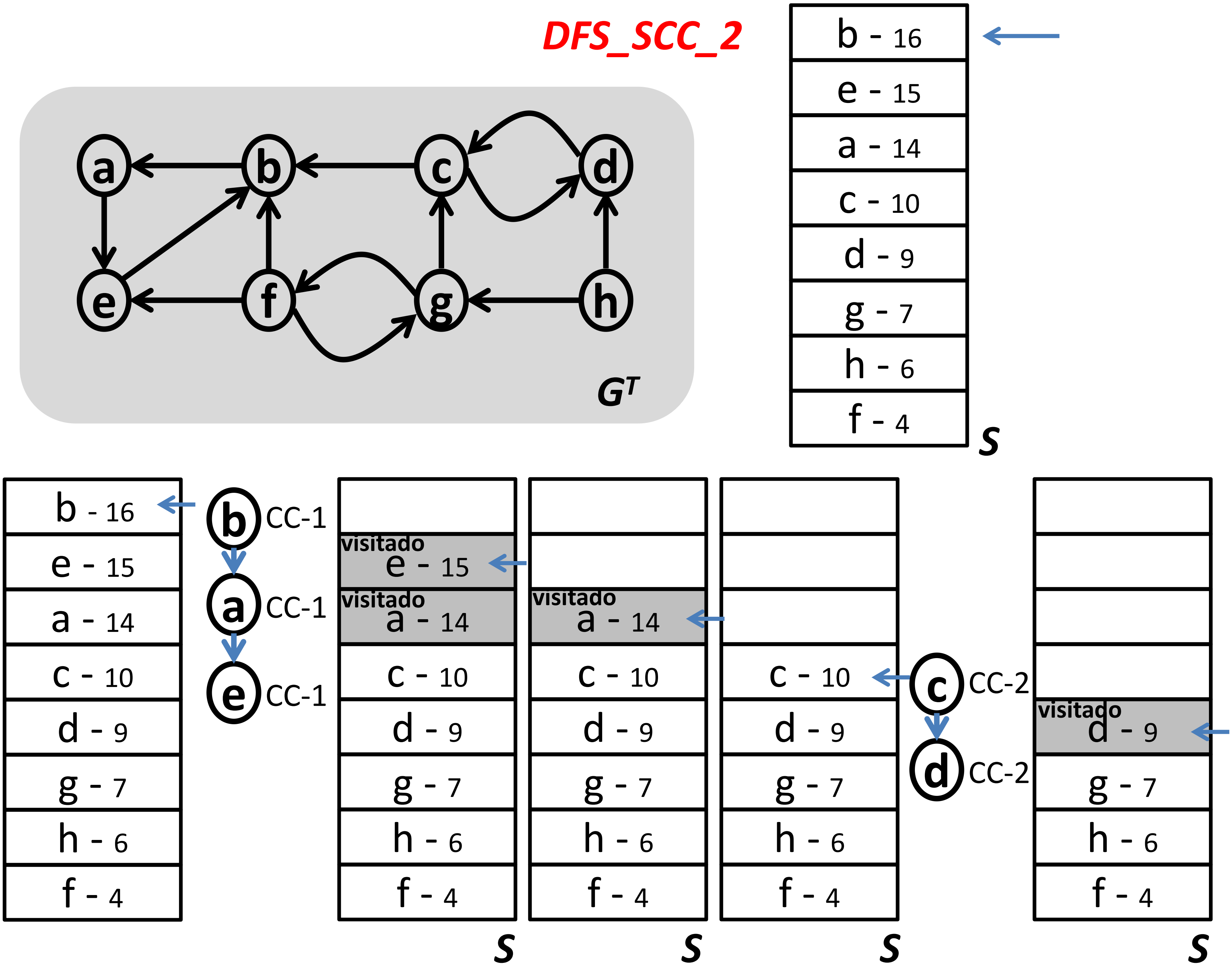



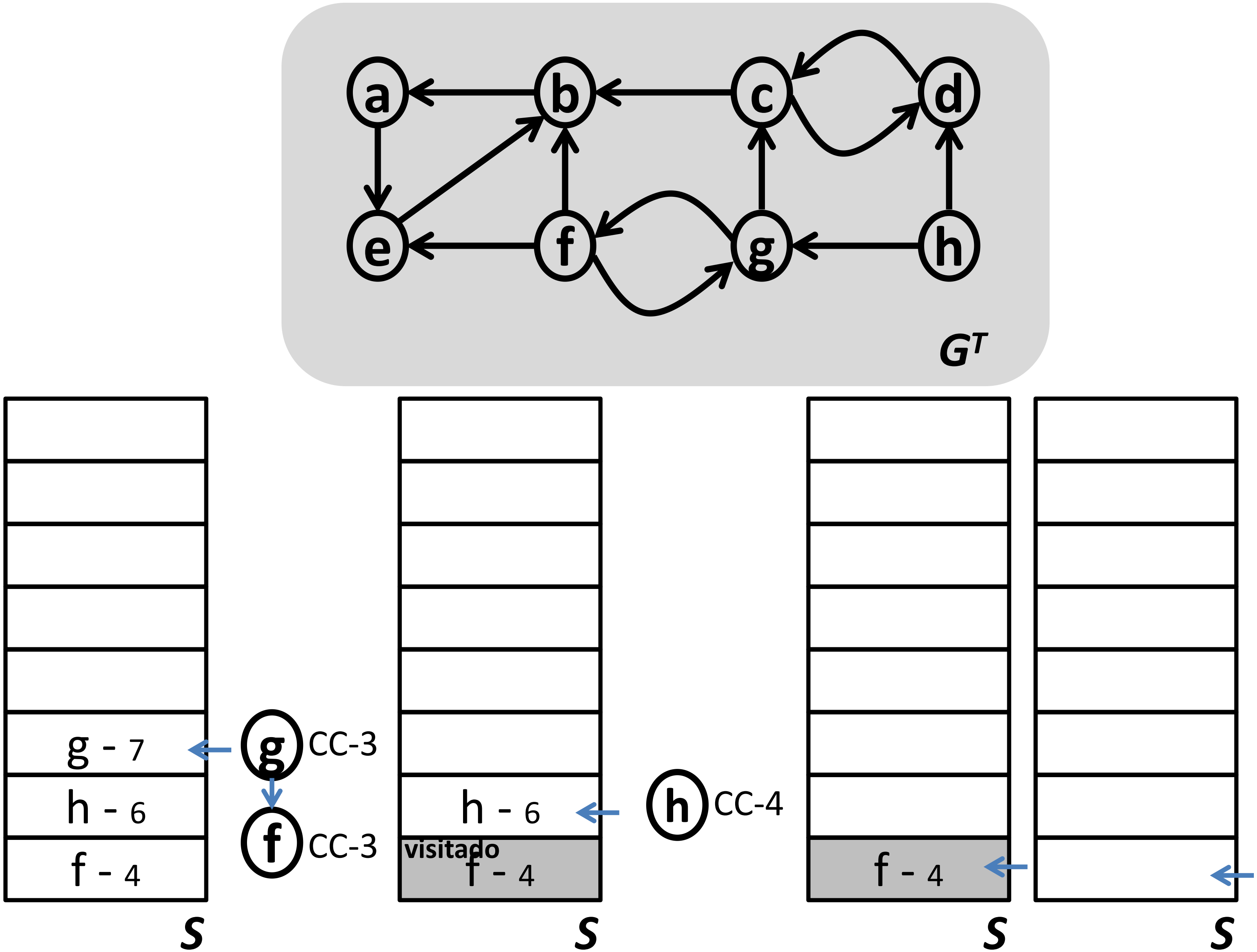




| |
|---|
| b - 16 |
| e - 15 |
| a - 14 |
| c - 10 |
| d - 9 |
| g - 7 |
| h - 6 |
| f - 4 |

***S***

***DFS_SCC_1***


DFS_SCC_2
a
b
c
d
e
f
g
h
G^T
b - 16
e - 15
a - 14
c - 10
d - 9
g - 7
h - 6
f - 4
S
b
CC-1
a
CC-1
e
CC-1
visitado
e - 15
visitado
a - 14
c
CC-2
d
CC-2
visitado
d - 9



a
b
c
d
e
f
g
h
$G^T$
g - 7
h - 6
f - 4
S
g
CC-3
f
CC-3
h - 6
visitado
f - 4
S
h
CC-4
f - 4
S
S

# Propiedades de las componentes fuertemente conexas de un grafo dirigido

Extendamos la notación *d[**u**]* y *f[**u**]* dada para vértices de G=(V, E): $u \in V$, a **conjuntos de vértices**:

Si $U \subseteq V$ entonces se define

$$\boldsymbol{d(U) = \text{mín}_{u \in U}\{d[u]\}}$$
$$\boldsymbol{f(U) = \text{máx}_{u \in U}\{f[u]\}}$$

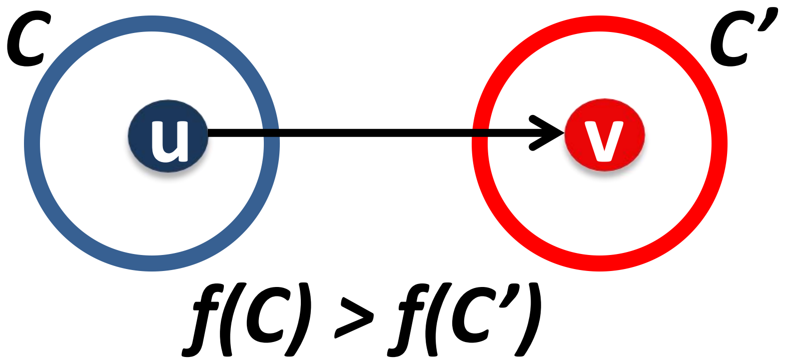
***d[U]:*** es el valor del **primer momento** en que se descubre un vértice de *U*

***f[U]:*** es el **último momento** en que un vértice de *U* finaliza la recursividad

# Propiedades de las componentes fuertemente conexas de un grafo dirigido

***Lema 2***

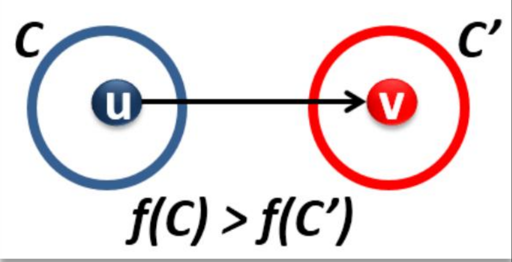
Sean ***C*** y ***C'*** dos **componentes fuertemente conexas** diferentes del grafo dirigido ***G=(V, E).*** Supongamos que existe un arco ***(u, v)*** $\in$***E***, con ***u*** $\in$***C*** y ***v*** $\in$***C'***. Entonces ***f(C) > f(C')***




***Demostración*** :

Existen dos casos, en dependencia de cuál de las dos componentes fuertes, ***C*** o ***C'***, tenga el vértice de menor **d[]**

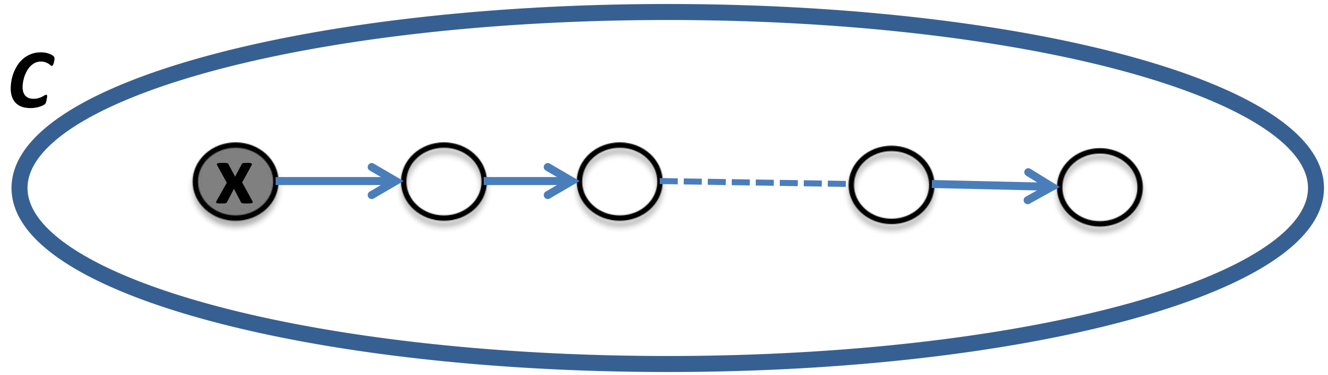
## *Lema 2 - Demostración* – CASO 1: *d(C) < d(C')*




Sea ***x*** el primer vértice que se descubre en ***C***

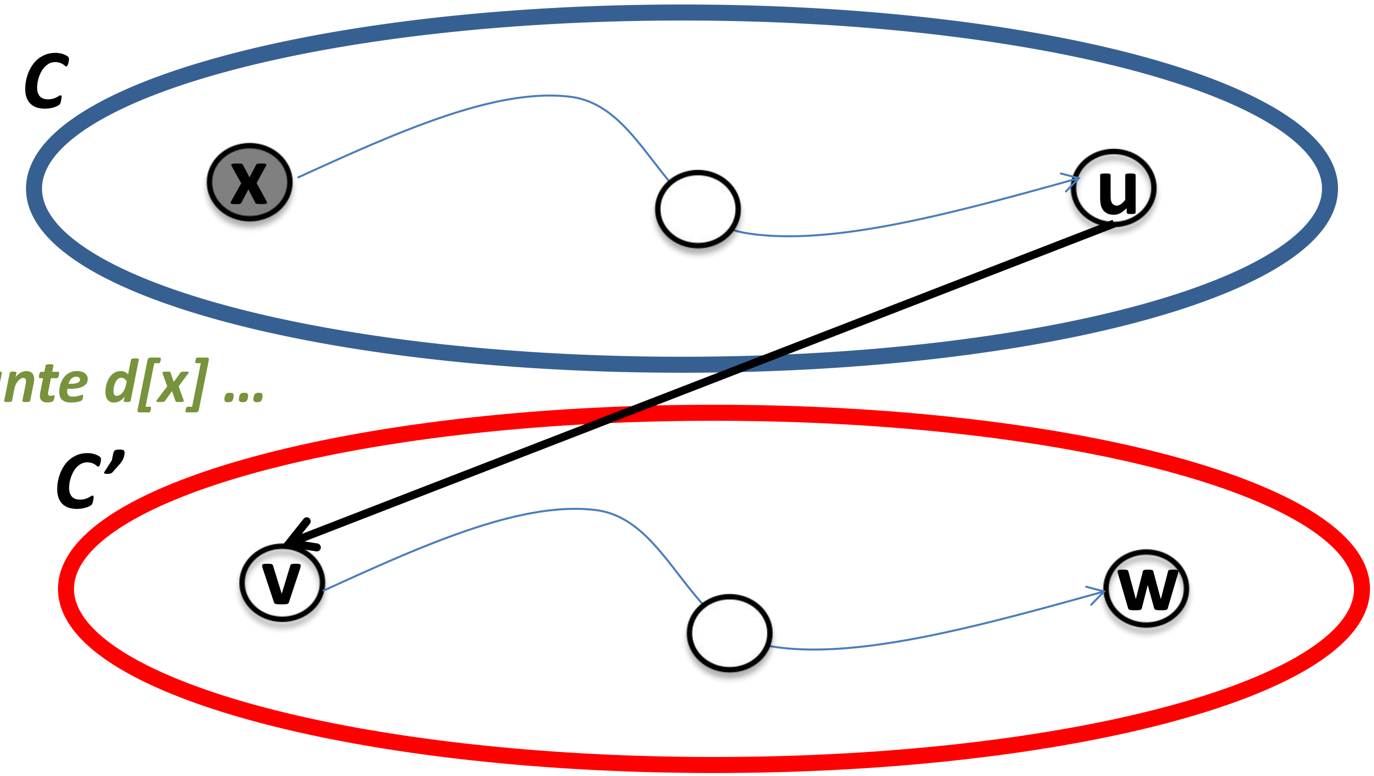
En el instante ***d[x]*** los restantes vértices en ***C*** y en ***C'*** son blancos
Entonces puede decirse que hay un camino en ***G,*** de ***x*** a los restantes vértices de ***C,*** donde todos vértices en dicho camino son blancos

*En el instante d[x] …*



## *Lema 2 - Demostración* – CASO 1 (continuación):

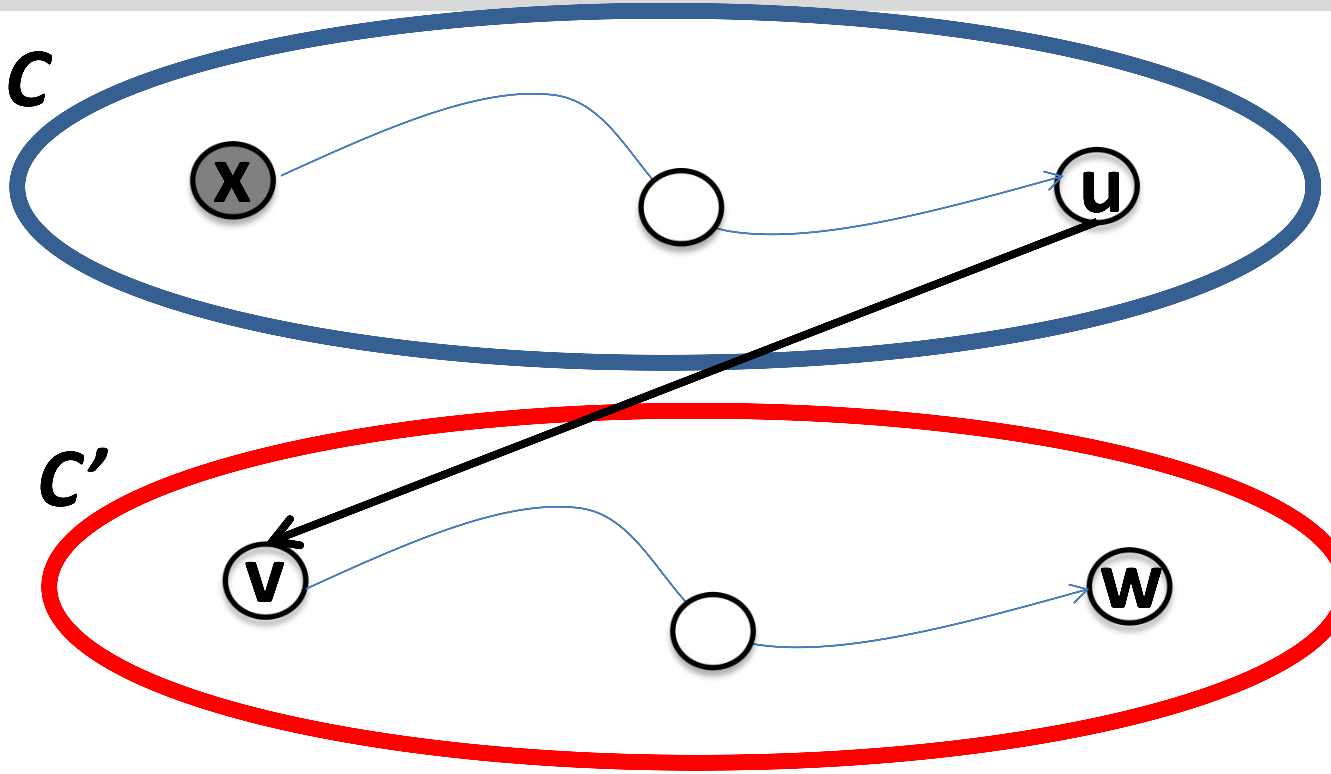
Como ***(u, v)*** $\in$ ***E*** , entonces para cualquier vértice *w* $\in$ *C'* , en el instante *d[x],* hay también un camino de *x* a *w* en el cual todos los vértices que pertenecen al mismo son blancos




Serán caminos de la forma ***x*** $\sim$ ***u*** $\rightarrow$ ***v*** $\sim$ ***w***
Por tanto, todos los vértices en *C* y en *C'*, se convierten en descendientes de ***x*** en el árbol en profundidad del cual ***x*** es raíz

***Lema 2 - Demostración – CASO 1 (continuación):***

**Corolario 22.8** (demostrado en el *I. to A.*): *El vértice **v** es un descendiente propio del vértice **u** en el DFS para un grafo dirigido **G si y solo si d[u] < d[v] < f[v] < f[u]***




*d[x] < d[u] < f[u] < f[x]*

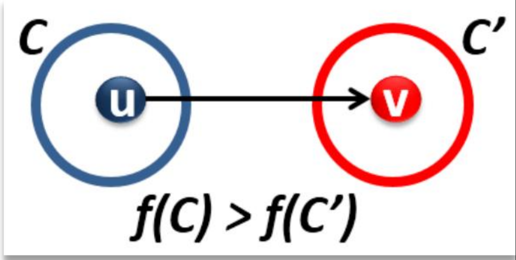
*d[u] < d[v] < f[v] < f[u]*

*d[v] < d[w] < f[w] < f[v]*

*d[x] < d[u] < d[v] <d[w] <**f[w]<f[v] < f[u] < f[x]***

Por tanto, a partir de todo lo anterior, *f[x] = f(C)> f(C')*

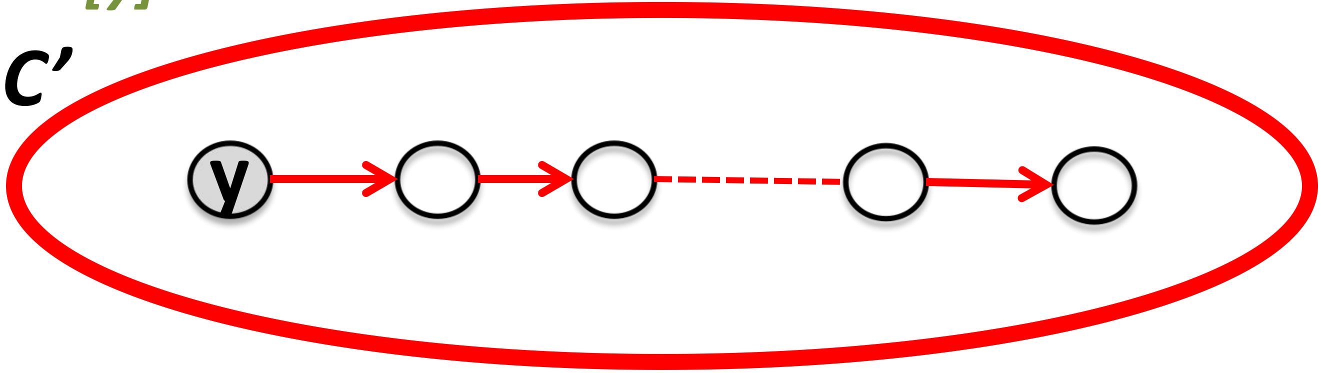
## *Lema 2 - Demostración –* **CASO 2**: *d(C) > d(C')*




Sea ***y*** el <u>primer vértice que se descubre</u> en ***C'.***

En el instante *d[y]* todos los vértices en ***C'*** son blancos y además, hay un camino en ***G***, desde ***y*** hasta cualquiera de los restantes vértices en ***C'***, donde todos los vértices en dicho camino son blancos

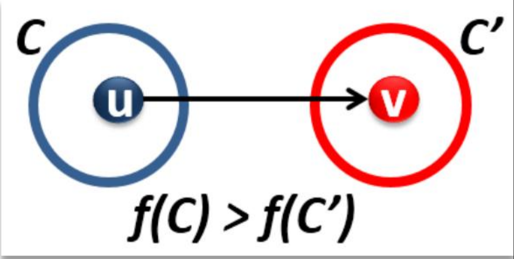
***En el instante d[y] …***




Por tanto, todos los vértices en *C'* se convierten en descendientes de ***y*** en el árbol abarcador en profundidad y por el **Corolario 22.8**

*f[y] = f(C')*

## *Lema 2 - Demostración* – CASO 2: *d(C) > d(C')*




Como existe un arco ***(u, v)*** de ***C*** a ***C'***, Por el **Lema 1** podemos afirmar que no puede existir un arco de ***C'*** a ***C***

Por tanto, ningún vértice en ***C*** es alcanzable desde ***C'***

Por consiguiente, en el instante ***f[y],*** todos los vértices en ***C*** siguen aún siendo blancos lo cual implica ***f(C) > f(C')***

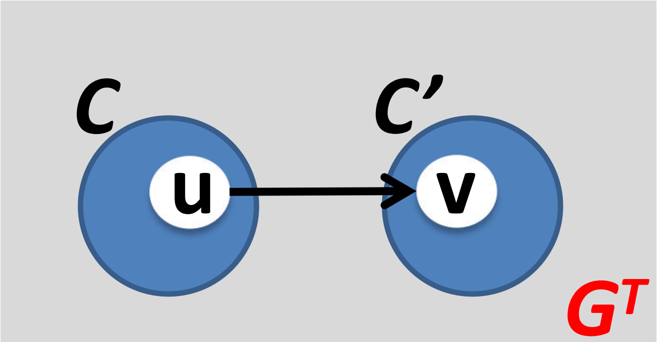
# Propiedades de las componentes fuertemente conexas de un grafo dirigido

***Corolario 3***

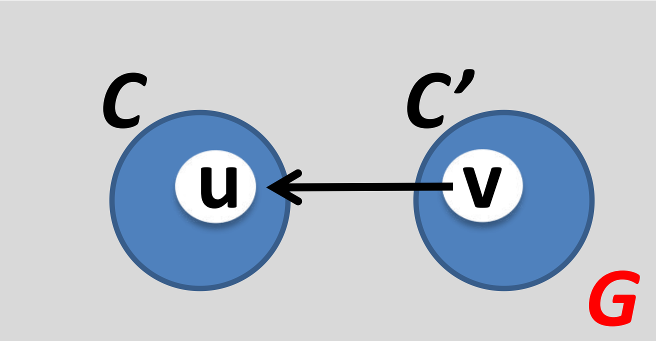
Sean ***C*** y ***C'*** dos **componentes fuertes diferentes** del **grafo dirigido *G=(V, E)***. Supongamos que existe un arco ***(u, v)***$\in$***$E^T$***, ***$G^T$=($V^T$, $E^T$)***, donde ***u***$\in$***C*** y ***v***$\in$***C'***. Entonces ***f(C) < f(C')***

**Demostración**

Como ***(u, v)***$\in$***$E^T$***, entonces ***(v, u)*** $\in$**E**. Como **las componentes fuertemente conexas en G y en $G^T$ son las mismas**, entonces por el **Lema 2** se tiene ***f(C)<f(C')***




$\Leftrightarrow$



# Correctitud del Algoritmo

El **Corolario 3** brinda la clave para demostrar la correctitud de
**STRONGLY-CONNECTED-COMPONENTS**

Examinemos que sucede cuando se aplica el segundo *DFS* a $G^T$:

- Se comienza por la ***C*** para la cual se cumple que ***f(C)* es máximo**
- La búsqueda comienza en algún vértice $x \in C$. A partir de este, se visitan los restantes vértices de dicha componente (todos *blancos*)

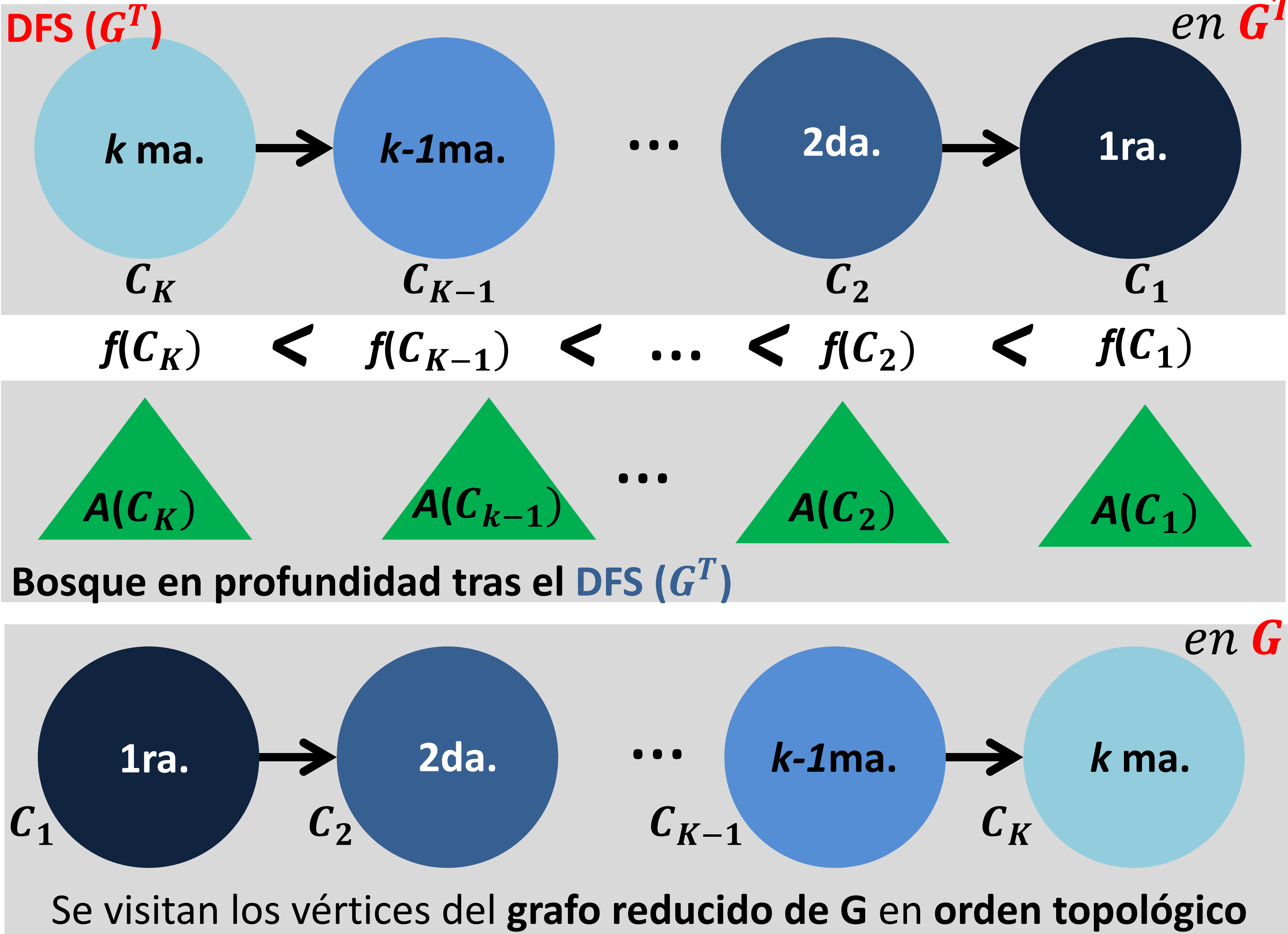
**- Por el Corolario 3 podemos afirmar que no hay arcos en $G^T$ que vayan de *C* a otra componente fuerte, por lo cual, la búsqueda que comenzó en *x* no alcanzará ningún vértice de otra componente que no sea *C***

- Por tanto, tras el segundo DFS, **el árbol en profundidad con raíz *x* contendrá exactamente a todos los vértices que pertenecen a la componente conexa *C***

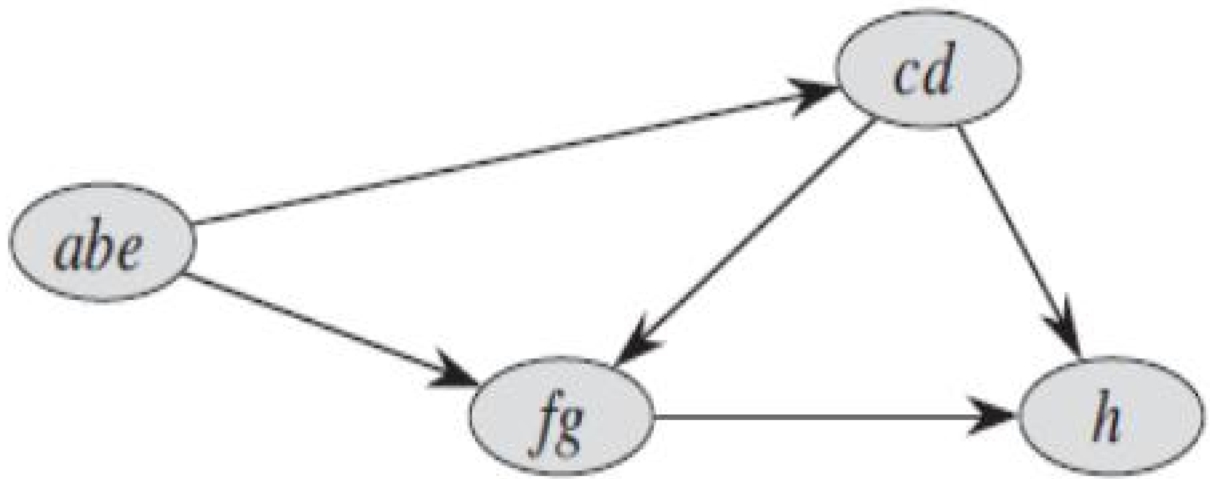
# Correctitud del Algoritmo

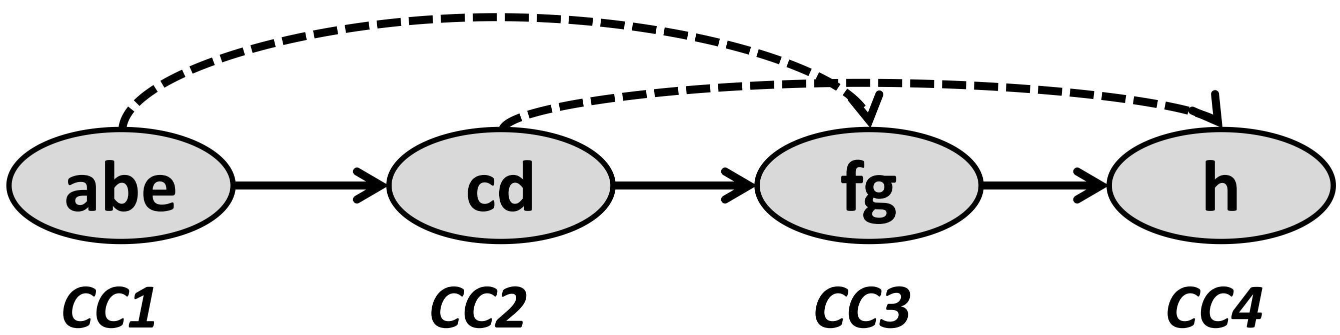
- Una vez completada la visita a **todos los vértices** de la componente conexa ***C***, la búsqueda selecciona como raíz para un nuevo árbol que se va a formar, a algún vértice de otra componente fuerte ***C'*** cuyo *f(**C'**)* es el mayor con respecto a las restantes componentes fuertes distintas de ***C***
- Nuevamente, la búsqueda alcanzará a todos los vértices en ***C'*** y por el **Corolario 3,** los únicos **arcos en $G^T$** que van de ***C'*** a otra componente fuerte, tienen que ir de ***C'*** a ***C*** y se sabe que ***C*** ya fue completamente visitada
- En general, cuando en el segundo DFS se visita alguna componente fuerte particular, los arcos que parten de esta tienen que estar dirigidos hacia una componente fuerte que ya fue completamente visitada durante dicho DFS
- Cada árbol en profundidad del bosque que se forma tras el segundo DFS, corresponde, exactamente, con una componente fuerte mente conexa del grafo

$C_1, C_2, \ldots, C_{k-1}, C_k$ componentes fuertes de G



La estrategia seguida en **DFS ($G^T$)** presupone visitar los vértices del **grafo reducido** en **orden topológico**



# Correctitud del Algoritmo

## Teorema 4

**STRONGLY-CONNECTED-COMPONENTS(*G*)** calcula correctamente las componentes fuertemente conexas de un grafo dirigido ***G***

**Demostración**

La demostración se hará por inducción sobre el número de árboles calculados **tras el segundo DFS** que se aplica sobre $G^T$

**Hipótesis de inducción:**

Los primeros ***k*** árboles que se forman durante la aplicación del segundo DFS sobre $G^T$ se corresponden con ***k*** componentes fuertemente conexas de ***G***

**Caso Base:**

$k=0$, es trivial que se cumple

**Paso de inducción:**
Asumimos que se cumple la hipótesis de inducción y **consideremos el *k+1*-*ésimo* árbol que se forma**:

Sea el vértice ***u*** la raíz de dicho árbol y sea ***C*** la componente fuerte a la cual pertenece el vértice ***u***

$\Rightarrow f[u] = f(C)$ - por la forma en que se seleccionan las raíces de los árboles en el segundo DFS que se aplica a $G^T$

$f[u] = f(C) > f(C')$ - para cualquier otra componente fuerte $C' \neq C$ **que aún no ha sido visitada**

En el momento en que el segundo *DFS* alcanza a ***u*,** los restantes vértices en ***C*** son ***blancos*** **-** por **hipótesis de inducción**

$\Rightarrow$ los restantes vértices que están en la componente ***C*** son descendientes de ***u*** en el árbol en profundidad del cual ***u*** es raíz

**Faltaría entonces demostrar que en dicho árbol no haya ningún otro vértice que no pertenezca a C:**

Por la propia hipótesis de inducción y por el **Corolario 3**, puede afirmarse que cualquier arco en $G^T$ que salga de $C$ tiene que llegar a algún componente fuerte que ya ha sido visitado

$\Rightarrow$ los vértices accesibles desde $u$ que no pertenezcan a $C$ serán **vértices de un componente fuerte que ya fue descubierto**,

$\Rightarrow$ no serán añadidos al árbol en profundidad del cual $u$ es raíz,

$\Rightarrow$ solo quedarán en el mismo los vértices que pertenecen a $C$

# Complejidad Temporal del Algoritmo

**STRONGLY-CONNECTED-COMPONENTS (*G*)**

1 DFS (*G*) para calcular *f*[*u*] $\forall u \in$ V
2 Determinar $G^T$
3 DFS ($G^T$), comenzando por el vértice *u* de mayor *f*[*u*] (Si la búsqueda en profundidad no llega a todos los vértices, iníciese la siguiente búsqueda a partir del vértice blanco de mayor *f[u]*)
4 Cada árbol del bosque abarcador resultante es una **componente fuertemente conexa** de *G*

***O(V+E)*** → Asumiendo una representación del Grafo por **Lista de Adyacencia**

# Ejercicio Propuesto

*Professor Bacon claims that the algorithm for strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices in order of increasing finishing times. Does this simpler algorithm always produce correct results?*

Esto es, hacer el algoritmo sin usar la transpuesta de G y comenzar el 2do DFS por el vértice $v \in V$ tal que, **f[v] es mínimo**. Si la búsqueda en profundidad no llega a todos los vértices, iniciar la siguiente búsqueda a partir del vértice blanco de menor *f[].*
*( o sea, se van considerando los f[] en orden ascendente)*

Serían correctos los resultados que se alcanzan ?