# TAREA SEMANAL 1-4:

# GRAFOS CFCs

# Y

# ORDEN TOPOLÓGICO

INTEGRANTES:
Andrés Monetta
Alejandro Clara
Sebastián Daloia

# Indice

# 1 Ejercicio

## Parte 1

Sea G un grafo dirigido y $G_C$ su grafo de componentes

1- Considere una DFS de G en la cual los vértices son apilados en una Pila P al terminar de ser procesados. Para cada CFC $C_i$ sea $f_i$, $1 \leq i \leq k$ el primer vértice visitado, y sea $f_k, \ldots, f_1$ el orden en que esos vértices son apilados en P.

### Observación 1:

Una recorrida DFS va marcando los vértices visitados en pre – procesamiento y los apila en pos procesamiento.

### Observación 2

Si en un recorrido DFS hay un camino entre $f_i$ y $f_j$, y $f_i$ es visitado antes que $f_j$, siendo $f_i$ y $f_j$ como en **1**, entonces $f_i$ no será procesado en pos procesamiento hasta recorrer todos los vértices para los cuales haya un camino y aún no hayan sido visitados, por lo que $f_j$ será pos procesado antes que $f_i$.

### Proposición a:

Si en una recorrida DFS el vértice $f_i$ es visitado antes que el vértice $f_j$ y hay un camino desde $f_i$ hacia $f_j$, entonces fi será apilado después de $f_j$.

Demostración:
Como $f_i$ y $f_j$ son tales que se relacionan como se indica en la Observación 2, y la pila va agregando elementos en pos procesamientos, entonces es cierta la proposición.

### Proposición b:

Para cada CFC $C_i$, $f_i$ es el ultimo de sus vértices que se apila en P.

Demostración:

Como $f_i$ pertenece a CFC $C_i$ hay un camino entre este vértice y todos los vértices que pertenecen a $C_i$.
Luego como $f_i$ es visitado antes que todos los demás por la observación previa es el último en ser apilado para esa CFC.

## Proposición c:

Si hay algún camino desde la CFC $C_i$ a la CFC $C_j$, entonces $f_i$ se apila en P despues que $f_j$.

Demostración:
Si $f_i$ es visitado antes que $f_j$ entonces por Proposición a esto es cierto.
Es decir sea ($u_i$, $u_j$) tal que $u_i \in CFC\ C_i$ y $u_j \in CFC\ C_j$
Como $C_i$ es CFC podemos armar un camino $\{u_i, \ldots, f_i, u_j, \ldots, f_j\}$
Es decir existe un camino $\{f_i, u_j, \ldots, f_j\}$ con $f_j$ aún no visitado.
Si $f_i$ es visitado despues que $f_j$ entonces la arista que va de un vertice $v_i$ de CFC $C_i$ a un vertice $v_j$ de CFC $C_j$ no forma parte del arbol de recubrimiento generado por DFS. Entonces CFC $C_j$ y CFC $C_i$ pertenecen a distintas ramas del árbol y la rama que contiene a CFC $C_j$ ha pasado ya por el pre y pos procesamiento, por lo que en este caso también $f_i$ se apila después que $f_j$.

## Proposición d:

Al terminar la DFS el vértice que queda en el tope de P pertenece a una CFC fuente.

Demostración:
$f_1$ es el último vértice en ser apilado entonces por Proposición c no hay camino alguno de $C_i$ a $C_1$ para $2 \leq i \leq k$ .
Entonces por definición de CFC fuente $C_1$ es fuente.

## Proposición e:

En el proceso de desapilar P, la secuencia $(f_1, \ldots, f_k)$ determina una secuencia $v = (v_1, \ldots, v_k)$ que es un ordenamiento topológico de $G_C$, donde $v_i$ representa la componente $C_i$

Demostración:
$(f_1, \ldots, f_k)$ por Proposición c son tales que no existen caminos que vayan de un $C_j$ a un $C_i$ con $j > i, i \neq i, i \leq k, j \leq k$ .
Entonces en el grafo de componentes no existen caminos que vayan de $v_j$ a $v_i$ con $j > i, i \neq i, i \leq k, j \leq k$
Entonces $v = (v_1, \ldots, v_k)$ es ordenamiento topológico.

# Parte 2

## Procedimiento encontrar_CFCS

```
PROCEDIMIENTO encotrar_CFCS
var
        pila: Pila
        ordenamiento: ColaPrioridad
        contador: entero

comienzo
        crearPila(pila)
        crearColaPrioridad(ordenamiento)

        para cada v perteneciente a V
                DFS_apilar(v,pila)
                desmarcar vertices de g
                dado G expresar su grafo transpuesto Gt
        fin para

        mientras No-vacia(pila)
                u = primero(pila)
                DFS_desapilar(u, Gt, pila, ordenamiento, contador)
                contador++
        fin mientras
fin
```

## Procedimiento DFS_apilar

```
PROCEDIMIENTO DFS_apilar (v: vertice, pila: Pila)
        comienzo
                marcar v
                para cada w adyacente a v
                        si w no marcado entonces
                                DFS_apilar(w, pila)
                        fin si
                fin para
                apilar(p,v)
        fin
```

## Procedimiento DFS_desapilar

```
PROCEDIMIENTO DFS_desapilar (v: vertice, G: grafo, pila: Pila, ordenamiento:
ColaPrioridad, contador: entero)

        comienzo
                marcar v
                desapilar(pila)
                encolarPrioridad(v, contador)
```

```
        para cada w adyacente a v
            si w no marcado entonces
                DFS_desapilar(w,pila)
            fin si
        fin para
fin
```

# 2 Descripción para encontrar colecciones

Dippy debería armar un grafo en base a los artículos y las referencias entre ellos, siendo éstos vértices y aristas respectivamente, como ejemplo véanse las Figuras 1, 2 y 3.

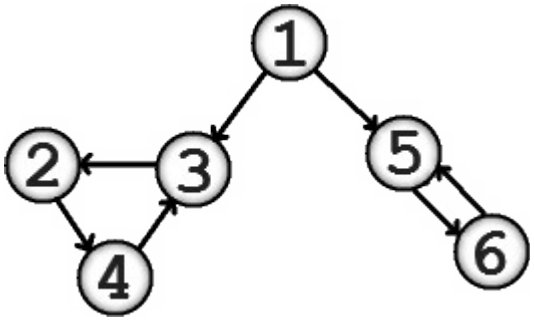


Figura 1 - En este caso los nodos numerados serían los artículos y las aristas las referencias entre ellos.

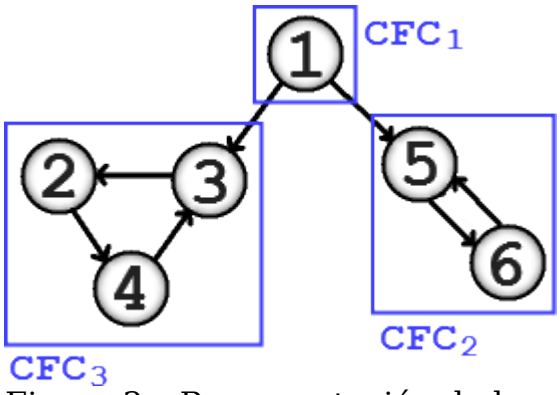


Figura 2 – Representación de las componentes CFC para el grafo de la figura 1

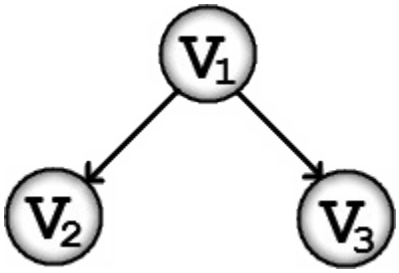


Figura 3 – Grafo de componentes para el grafo de la figura 1

Luego debería hacer una recorrida DFS agregando los artículos a una pila en pos-orden.

La inserción de elementos a la pila de esta forma, dotan a la estructura de las siguientes propiedades:

1) Al terminar la DFS, el vértice que queda en el tope de la pila pertenece a una CFC fuente.

2) En el proceso de desapilar, la secuencia de los f1,…,fk siendo estos los primeros vértices agregados de cada CFC determinan una secuencia v1,…,vk en el grafo de componentes, que es un ordenamiento topológico.

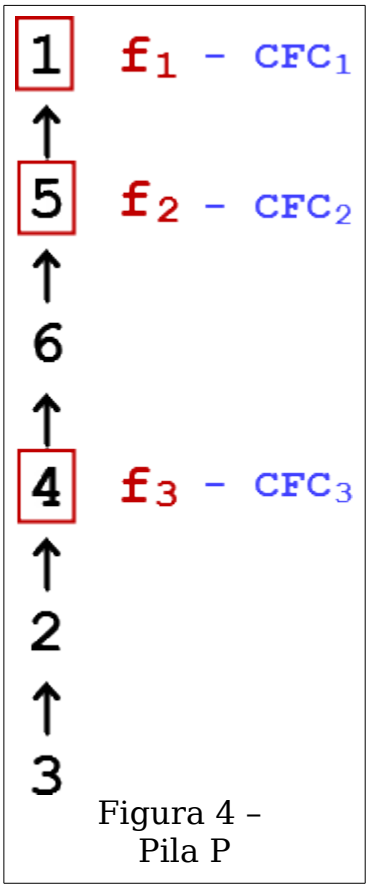

Figura 4 – Pila P

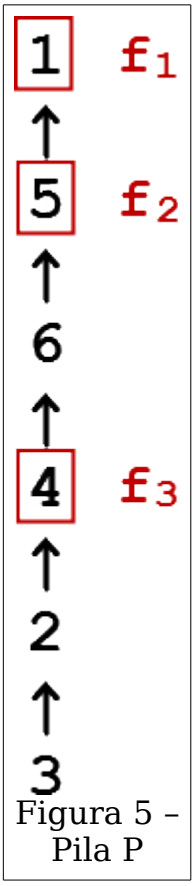

Figura 5 – Pila P

En las figuras 4 y 5 se representa la simulación de un recorrido DFS para el grafo de la figura 1 en donde el recorrido comienza en (1) y prosigue hacia los nodos con menor valor.

La pila P generada está dotada de las propiedades antes mencionadas, es decir la CFC de (1) es fuente (véase figura 2). Y el proceso de desapilar P determina una secuenca v1, v2, v3 a partir de los primeros vértices visitados en cada CFC, (1), (5), (4).

Esto sugiere una solución para identificar las componentes usando el grafo transpuesto. El procedimiento que debería implementar Dippy sería el siguiente:

La componente a la cual pertenece el primer elemento de la pila es fuente en G, y por lo tanto sumidero en G transpuesta. Entonces a partir de este primer elemento (haciendo un recorrido DFS) se encuentran todos los vértices pertenecientes a esa CFC. A medida que se recorren estos vértices se van desapilando y también agregando a una cola de prioridad, cuya prioridad es el número correlativo al del ordenamiento topológico. Estas operaciones se hacen en pre-orden. Entonces una vez recorrida la primer componente, se habrán extraído de la pila solo los vértices de la CFC que se estaba recorriendo. Luego por la observación 2 el siguiente elemento de la pila pertenecerá a una CFC fuente en G, la cual a su vez es sumidero en G transpuesta. De esta manera se podrá hacer el mismo recorrido para identificar los elementos de esta nueva componente, y desapilarlos de la pila agregándolos a la cola. Se continua de esta misma forma hasta que no haya más elementos en la pila, luego se devuelve la cola de prioridad con las colecciones identificadas.

A continuación se presenta un esquema de como sería la implementación del algoritmo para el grafo de la figura 1.

En las Figuras 6, 7 y 8 se esquematiza como sería el proces de desapilar P y

encolar una cola de prioridad con los vértices de cada CFC.

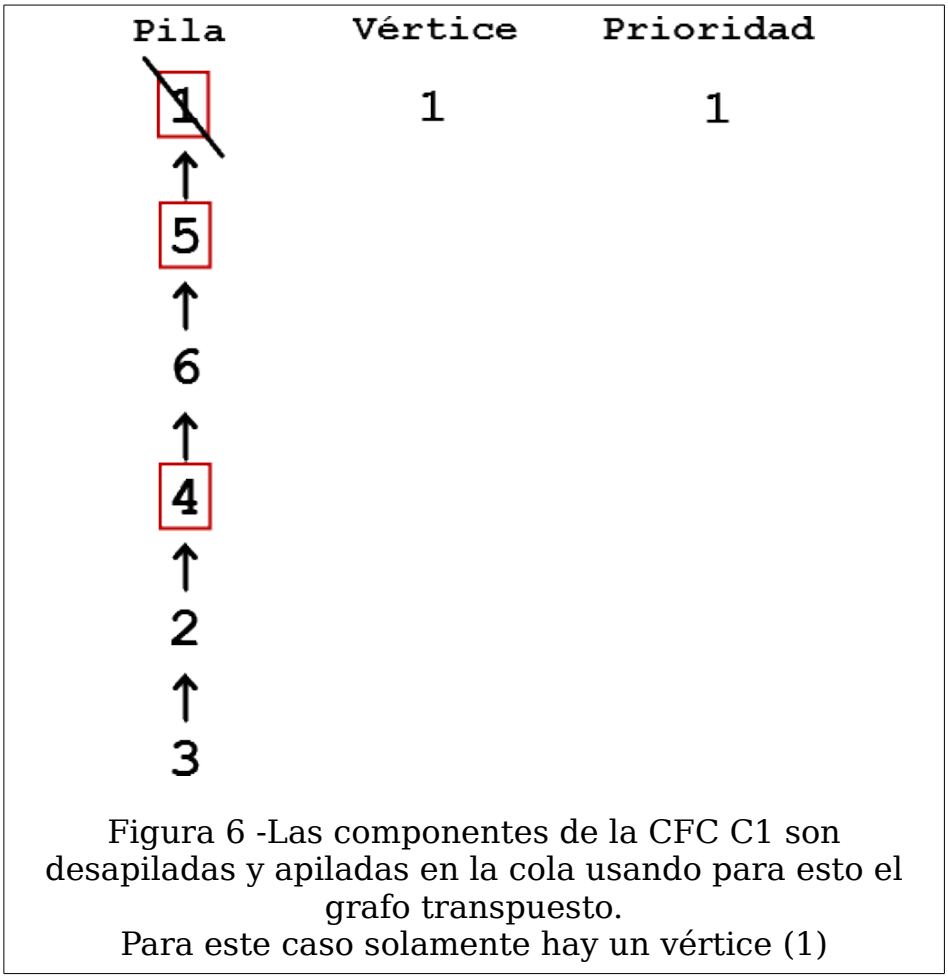


Figura 6 -Las componentes de la CFC C1 son desapiladas y apiladas en la cola usando para esto el grafo transpuesto.
Para este caso solamente hay un vértice (1)

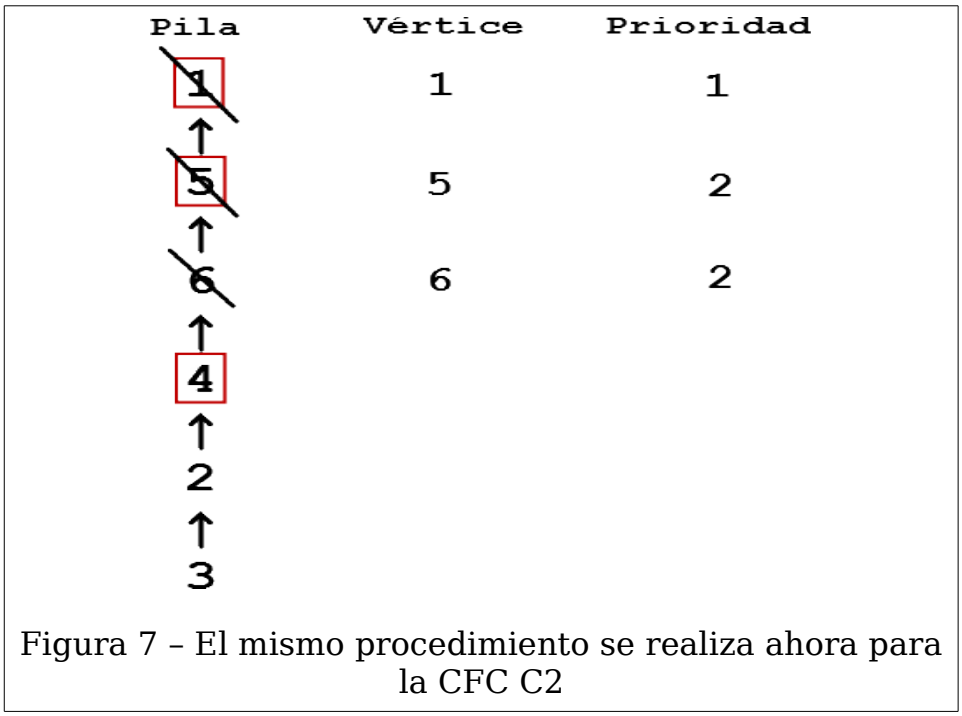


Figura 7 – El mismo procedimiento se realiza ahora para la CFC C2

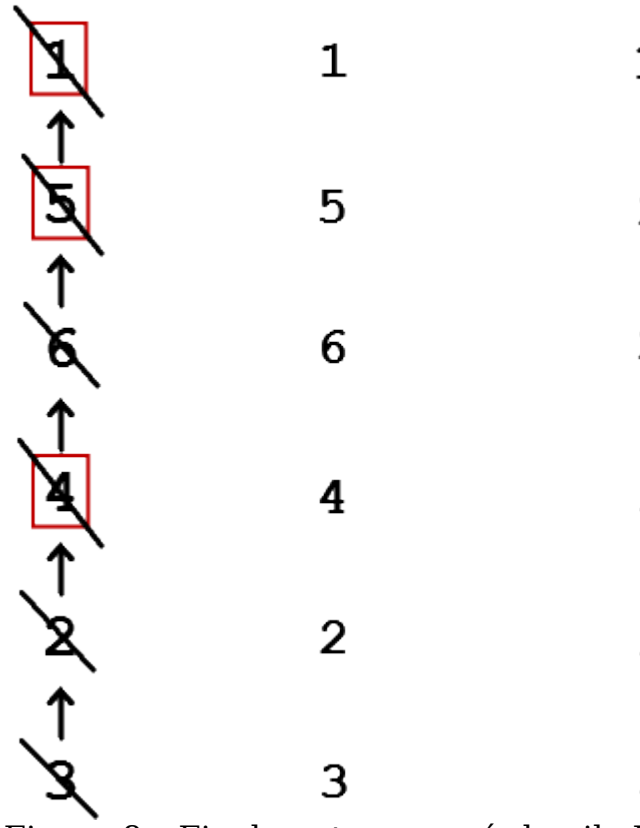


Figura 8 – Finalmente se vacía la pila P y se terminan encolando los vértices de la CFC C3 del ejemplo.

# 3 Seudocódigo: Grado de separacion

## Procedimiento separacion_a_articulosBFS

```
PROCEDIMIENTO separacion_a_articulosBFS(v:Vértice,lista_distancias
                                                        :Lista)

VAR   Q:Cola de vértice
      u,w: vértice
      distAux: entero

Comienzo

     CrearCola(Q)
     Marcar(v)
     setNivel(v, 0)/*Establece el nivel del vertice pasado como
                         parámetro*/
     InsBack(Q,v) /*”encolar”*/
     distAux = 0

     Mientras No-Vacia(Q)

          u = primero(Q)
          Q = resto(Q)

          Para cada w adyacente a u
               Si w no marcado entonces
                    Marcar(w)
                    setNivel(w,getNivel(u) + 1)   /*getNivel
     Devuelve el nivel del vértice pasado como parámetro*/
                    si getNivel(w) > distAux entonces
                         distAux = getNivel(w)
                    InsBack(Q,w)
               Fin si
          Fin para
     Fin Mientras
agregarALista(lista_distancias,identificador(v),distAux)
     /*identificador devuelve el nombre del vértice*/
Fin
```

## ProcedimientoGradoSeparacion

```
GradoSeparacion(g: grafoCFC): entero
var
     l :Lista
comienzo
     crearLista(l)
```

```
     para cada v en g
           separacion_a_articulosBFS(v,l)
           /*se encarga de encontrar la max distancia entre el
                                vertice v y todos los demas*/
     fin para
     retornar (MaxAux(l)) /*devuelve el maximo de la lista*/
fin
```