

# Grafos

Pedro O. Pérez M., MTI

Análisis y diseño de algoritmos  
Tecnológico de Monterrey

*pperezm@tec.mx*

04-2019

# Contenido

Introducción

Búsquedas en grafos

Topological Sort

Grafo bipartita

Punto Articulado

Flujo máximo

# Definición

Un grafo no direccionado  $G = (V, E)$  consiste de una colección de vértices,  $V$  y una colección de arcos,  $E$ . Representamos cada arco,  $e \in E$ , como un subconjunto de dos elementos  $e = \{u, v\}$  siendo  $u, v \in V$ , llamando a  $u$  y  $v$  los puntos terminales de  $e$ .

Un grafo direccionado  $G = (V, E)$  consiste de una colección de vértices,  $V$  y una colección de arcos,  $E$ . Representamos cada arco,  $e \in E$ , como un par ordenado de dos elementos  $e = (u, v)$  siendo  $u, v \in V$ . Llamamos a  $u$  el punto inicial y a  $v$  el punto final de  $e$ .

- ▶ Un grafo no direcccionado está conectado si, para cada par de nodos  $u$  y  $v$ , existe un camino de  $u$  a  $v$ .
- ▶ Un grafo direcccionado está fuertemente conectado si, para cada par de vértices  $u$  y  $v$ , existe un camino de  $u$  a  $v$  y de  $v$  a  $u$ .
- ▶ En un grafo no direcccionado  $G = (V, E)$  una secuencia de nodos  $P = [v_1, v_2, \dots, v_{k-1}, v_k]$  con la propiedad de que cada par consecutivo  $v_i, v_{i+1}$  está conectado por un arco en  $G$ , es llamado un camino (path) de  $v_1$  a  $v_k$ .
- ▶ Un ciclo es un camino  $P = [v_1, v_2, \dots, v_{k-1}, v_k]$  en el cual, para cualquier  $k > 2$ , los primeros  $k-1$  vértices son distintos y  $v_1 = v_k$ .

**Input:**  $u : \text{Vertex}, G : \text{Graph}, \text{Reached} : \text{Set}$

**for each  $(u, v)$  in  $G$  incident to  $u$  do**
$$DFS(v, G, Reached)$$

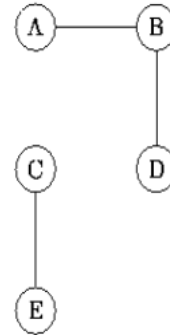
end for

end while

Considere un grafo  $G$  formado a partir de un gran número de vértices conectados por arcos.  $G$  se dice que está conectado si existe un camino entre cualquier par de vértices en  $G$ . Por ejemplo, el siguiente grafo no está conectado, porque no hay trayectoria de A a C.

Este grafo contiene, sin embargo, un número de subgrafos que están conectados, uno para cada uno de los siguientes conjuntos de vértices: (A), (B), (C), (D), (E), (A, B), (B,D), (C, E), (A, B, D). Un subgrafo conectado es máximo si no hay vértices y arcos en el grafo original que podrían añadirse al subgrafo y todavía dejarlo conectado. En la imagen anterior, hay dos subgrafos máximos, uno asociada con los vértices (A, B, D) y el otro con los vértices (C, E). Desarrollar un algoritmo para determinar el número de subgrafos máximos conectados de un gráfico dado.

<http://bit.do/eNTvC>





**Input:**  $G : Graph$

$$acum \leftarrow 0$$

Mark all the vertexes in *Reached* as *No Explored*

```
for vertex in G do
```

if *vertex* is not marked *Explored* then

$$acum \leftarrow acum + 1$$
$$DFS(vertex, G, Reached)$$

end if

end for

```
return acum
```

# Topological Sort

Un "Topological Sort" de un Grafo Direcccionado Acíclico (Directed Acyclic Graph, DAG) es un ordenamiento lineal de los vértices que aparecen en un DAG tal que si el vértice  $u$  aparece antes de  $v$  es porque existe un arco ( $u \rightarrow v$ ) en el DAG. Cada DAG tiene al menos, y posiblemente más, "topological sort".

---

**Input:**  $u : \text{Vertex}, G : \text{Graph}, \text{Reached} : \text{Set}, \text{TS} : \text{Stack}$

Mark  $u$  as *Explored* and add to *Reached*

**for** each  $(u, v)$  incident to  $u$  **do**

if  $v$  is not marked *Explored* then

$$DFS2(G, v, Reached, TS)$$

end if

end for

$$TS.push(u)$$

**Input:**  $G : Graph$

Reached : Set

$TS : Stack$

for each *vertex* in  $G$  do

if *vertex* is not marked *Explored* then

$$DFS2(G, v, Reached, TS)$$

end if

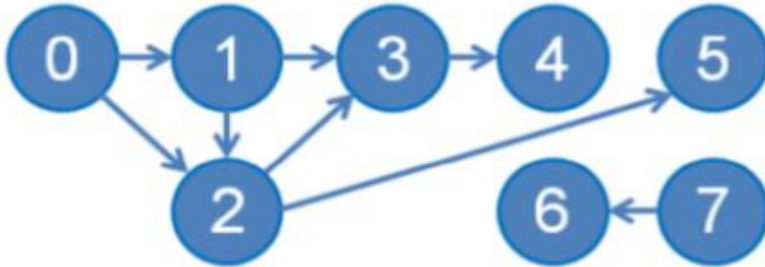
end for

```
while  $TS$  is not empty do
```

```
print TS.top()
```

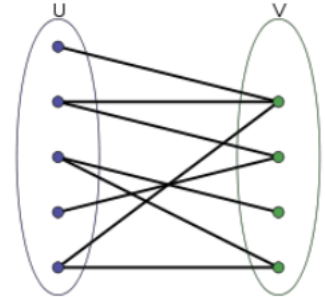
*TS.pop()*

end while



# Grafo bipartita

Un grafo bipartita (o bigrafo)  $G = (V, E)$  es un grafo cuyos vértices pueden ser divididos en dos conjuntos disjuntos  $R$  y  $S$  tal que cada arco conecta a un vértice en  $R$  con un vértice en  $S$ .



## Procedure 6 BIGRAPH

**Input:**  $G : \text{Graph}$

*Q : Queue*

*Color : Array*

*isBipartite* : *boolean*

$$INIT(Color, -1)$$

```
isBipartite  $\leftarrow$  true
```

$$vertex \leftarrow \text{some vertex in } G$$

*Q.enqueue(vertex)*

**while**  $Q$  is not empty **do**

NEXT SLIDE

end while

```
return isBipartite
```

---

```
u ← Q.dequeue()  
for each (u, v) incident in u do  
  if Color[v] = −1 then  
    Color[v] = 1 − Color[u]  
    Q.enqueue(v)  
  else  
    if Color[v] = Color[u] then  
      isBipartite = false  
    end if  
  end if  
end for
```

---



# Punto Articulado

Un punto articulado (o puente) se define como un vértice in un grafo  $G = (V, E)$  cuya remoción (todos los arcos que inciden sobre él también son removidos desconecta el grafo  $G$ . Un grafo que no tiene ningún punto de articulación se le conoce como biconectado.

**Input:**  $G : Graph$

```
for each  $v$  in  $G$  do
```

Remove  $v$  and its incident edges

$$newCC \leftarrow COUNTING_G RAPH S(G)$$

if  $newCC > originalCC$  then

```
return true
```

end if

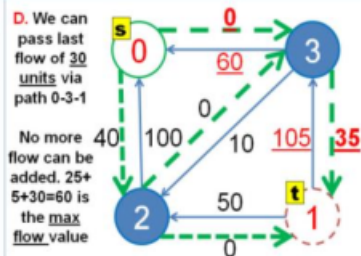
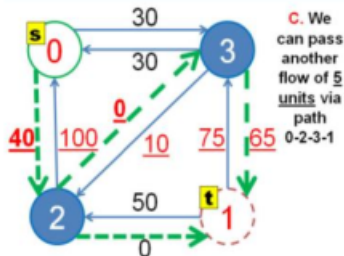
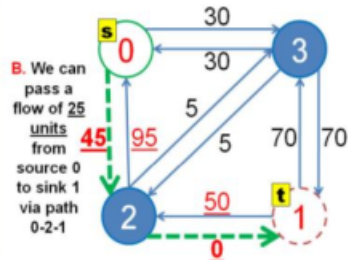
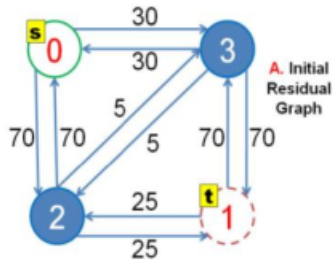
Restore  $v$  and its incident edges

end for

```
return false
```

# Flujo máximo

Una red de flujo es un grafo dirigido, donde cada arco tiene capacidad y cada arco recibe un flujo. La cantidad de flujo de un arco no puede exceder la capacidad del mismo. Un flujo debe satisfacer la restricción de que la cantidad de flujo en un nodo es igual a la cantidad de flujo que sale de ella, a menos que sea una *fuentes*, que sólo tiene flujo saliente, o un *sumidero*, que solo tiene flujo saliente.



## Procedure 8 MAX FLOW

---

**Input:**  $G : Graph$

Setup directed residual graph with edge capacity equal to original graph

$$max_{f low} \leftarrow 0$$

```
while there exists an augmenting path p from s to t do
```

Find  $f$ , the minimum edge weight along the path

Decrease capacity of the outgoing edges

Increase capacity of the incoming edges

$$\text{max}_f \text{low} \leftarrow \text{max}_f \text{low} + f$$

end while

```
return  $\max_f \text{low}$ 
```