

# Network Flows

## Práctica calificada V

Jarem Villalobos<sup>1</sup>   Joaquín Aynaya<sup>2</sup>

December 6, 2025



## 1 Introducción

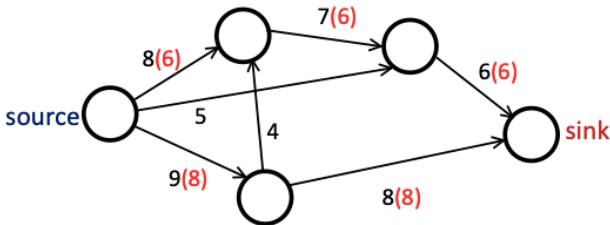
## 2 Ford-Fulkerson Algorithm



# Introducción

Nos enfocaremos en analizar problemas relacionados a redes de flujo, es decir problemas donde buscamos estudiar el tránsito de cierto material a través de una red.

Para ello, podemos modelar el problema usando un grafo, donde el peso de cada arista representa *la capacidad máxima* de cada conexión, y cada vértice representa un nodo de conexión. Por cada arista pasa un flujo, el cual es menor a su capacidad



De los modelos de redes de flujo, nos interesa en especial aquellos donde un vértice representa la fuente de la red, y otra un sumidero de ésta. A tal red se le denomina red de flujo fuente-destino (*st flow network*)

## Definition

Una red de flujo *st* es un dígrafo ponderado con pesos positivos (capacidades  $C : V \times V \rightarrow \mathbb{N}$ ) donde existen dos vértices especiales: la fuente (*source*) y el destino (*terminal* o *sink*). Además, cada vértice, exceptuando  $s$  y  $t$ , obedece la ley de conservación:

$$\sum_{v \in V} f(u, v) = 0, \quad \forall u \in V - \{s, t\} \quad (1)$$

## Definition

Un flujo-*st*  $f : V \times V \rightarrow \mathbb{N}$  es un conjunto de valores no negativos asociados a cada arista, referidos como *los flujos de cada arista*. Un flujo-*st* es **factible** si ninguno de los flujos sobrepasa la capacidad de su arista. Se denomina *valor del flujo-*st** al flujo de entrada del destino (que equivale al flujo de salida de la fuente)



Sobre este modelo, se enuncia el problema

## Flujo-*st* máximo (*maxflow*)

Dado una red de flujo-*st*, buscar un flujo-*st* factible tal que no exista ningún otro flujo sobre la red que tenga un mayor valor.

Notar lo siguiente: si la suma de las capacidades de las conexiones que entran a un nodo es igual que la suma de las que salen de éste, para todos los nodos en la red, entonces el problema estaría resuelto, pues bastaría rellenar la red hasta el tope.

Es en el caso donde las capacidades *no cumplen el equilibrio* en donde el problema es de especial relevancia, lo que suele ocurrir en el mundo real.



## 1 Introducción

## 2 Ford-Fulkerson Algorithm



El algoritmo Ford-Fulkerson logra resolver el problema *maxflow* antes tratado. También conocido como *Augmenting-path algorithm*, pues va incrementando de forma progresiva los flujos en cada arista basado en una idea simple pero delicada de manejar:

## Idea

Dado un camino desde  $s$  a  $t$ , podemos aumentar el flujo en todo el camino de manera segura basándonos en la arista con capacidad mínima en el camino.

Realizar esto sin ningún criterio puede no llegar a una solución que sea la máxima, por lo que debemos establecer un criterio adecuado.



Por ello, el algoritmo trabaja sobre el *grafo residual* del grafo original.

## Definición: Grafo residual

Dado  $G$  que representa una red de flujo- $st$ , su grafo residual  $G_f$  es aquel que contiene las mismas aristas de  $G$ , pero tiene aristas etiquetadas con que siguen la siguiente regla:

$$C_f(u, v) = C(u, v) + f(u, v)$$

$$C_f(u, v) > 0, \forall u, v$$

Donde  $f$  es la función de flujo. Además cumple que  $f(u, v) = -f(v, u)$

La última condición es importante para el algoritmo, pues va a permitir la existencia de *aristas residuales* en sentido contrario a su contraparte en  $G$



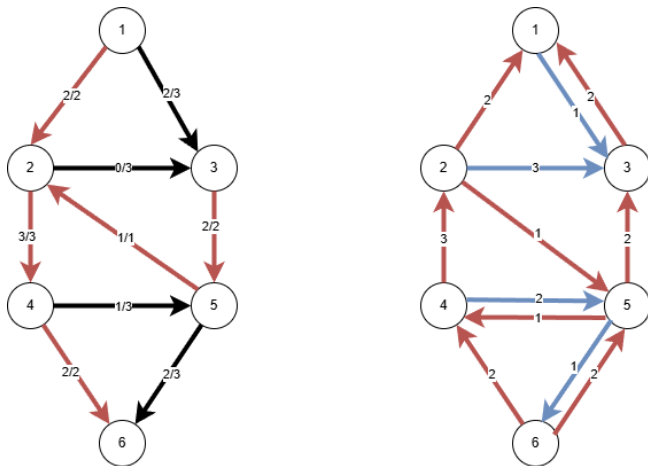


Figure: Grafo *st* y su grafo residual



El algoritmo se basa en los siguientes pasos:

- ❶ Establecer  $f(u, v) = 0$  para todos los vértices en el grafo  $G$
- ❷ Mientras  $G_f$  tenga un *camino incrementable*  $p$ 
  - Hallar la capacidad residual mínima de  $p$
  - Incrementar las aristas en el grafo original  $G$  por aquel valor
  - Repetir (2)
- ❸ Fin

Para el pseudocódigo, dado que es ineficiente tener que crear un grafo que represente  $G_f$ , simplemente mantendremos los valores  $c(u, v) - f(u, v)$  y  $f(u, v)$ . Además usaremos *breadth-first search* para encontrar el camino incrementable en el grafo residual



## Algorithm 1 Ford-Fulkerson Algorithm (with BFS)

```
procedure MAXFLOW( $G, s, t$ )
  for  $(u, v) \in E$  do
     $f(u, v) \leftarrow 0$ 
     $f(v, u) \leftarrow 0$ 
  end for
  parent[ ]  $\leftarrow$  Array of size  $G.v$ 
  while HASAUGMENTINGPATH( $G, s, t, \text{parent}$ ) do
     $\delta \leftarrow \infty$ 
     $v \leftarrow t$ 
    while  $v \neq s$  do
       $u \leftarrow \text{parent}[v]$ 
       $\delta \leftarrow \min(\delta, \text{RESCAPACITY}((u, v)))$ 
       $v \leftarrow u$ 
    end while
     $v \leftarrow t$ 
    while  $v \neq s$  do
       $u \leftarrow \text{parent}[v]$ 
      if  $(u, v) \in E$  then
         $f(u, v) \leftarrow f(u, v) + \delta$ 
      else
         $f(v, u) \leftarrow f(v, u) - \delta$ 
      end if
       $v \leftarrow u$ 
    end while
  end while
end procedure
```



---

**Algorithm 2** BFS approach for augmented path finder

---

```
procedure HASAUGMENTINGPATH( $G, s, t, \text{parent}$ )  
  for  $v \in V(G)$  do  
     $\text{parent}[v] \leftarrow \text{Empty}$   
  end for  
   $Q \leftarrow \text{Priority queue}$   
  ENQUEUE( $Q, s$ )  
   $\text{parent}[s] \leftarrow s$   
  while ! $Q.\text{empty}$  do  
     $u \leftarrow \text{DEQUEUE}(Q)$   
    for  $w \in \text{adj}[u]$  do  
      if  $\text{parent}[v] \neq \text{Empty}$  and  $\text{RES\_CAP}((u, v)) > 0$  then  
         $\text{parent}[v] \leftarrow u$   
        if  $v = t$  then  
          return True  
        end if  
        ENQUEUE( $Q, v$ )  
      end if  
    end for  
  end while  
  return False  
end procedure
```



---

## Algorithm 3 Residual capacity function

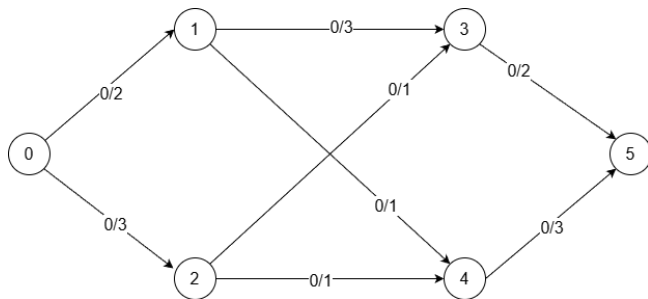
---

```
procedure RES_CAP( $u, v$ )  
  if  $(u, v) \in E$  then  
    return  $c(u, v) - f(u, v)$   
  else  
    return  $f(u, v)$   
  end if  
end procedure
```

---

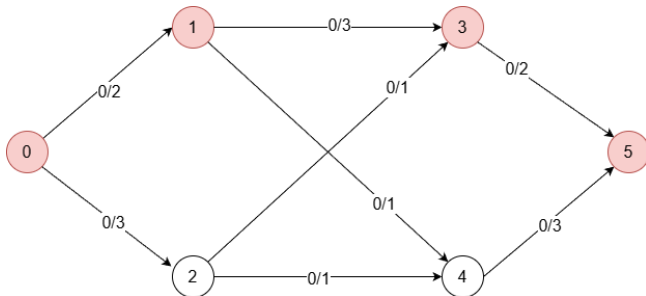


## Inicio



## Primera iteración

Grafo inicial:



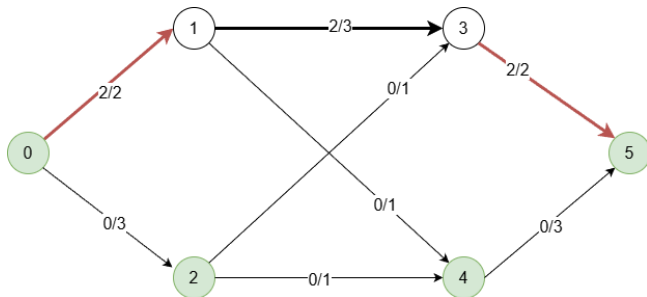
Camino encontrado con BFS: 0-1-3-5

Delta: 2  $\implies$  Aristas 0-1, 1-3, 3-5 incrementadas en 2



## Segunda iteración

Grafo inicial:

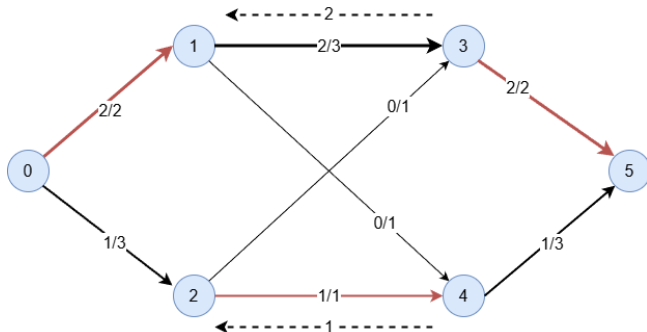


Camino encontrado con BFS: 0-2-4-5

Delta: 1  $\Rightarrow$  Aristas 0-2, 2-4, 4-5 incrementadas en 1



## Tercera iteración Grafo inicial:

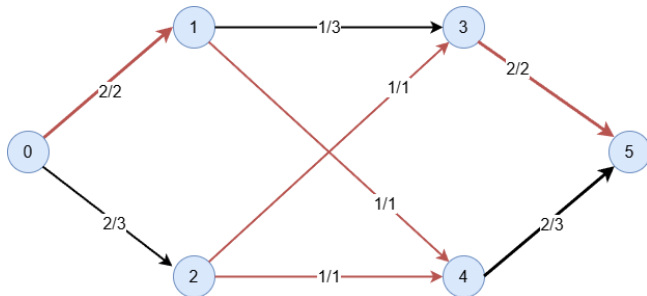


**Camino encontrado con BFS: 0-2-3-1-4-5**

**Delta: 1  $\Rightarrow$  Aristas 0-2, 2-3, 1-4, 4-5 incrementadas en 1, arista 1-3 decrementada en 1**



## Cuarta iteración Grafo inicial:



**Camino encontrado con BFS: Ninguno**  
**El algoritmo termina. MaxFlow = 4**



## Propiedades del flujo

La función de flujo cumple que:

$$f(X, X) = 0$$

$$f(X, Y) = -f(Y, X)$$

$$f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$$

## Lema

El valor del flujo de un corte  $st$  es igual al valor del flujo  $|f|$

## Teorema: MinCut-MaxFlow

Las siguientes proposiciones son equivalentes

- El valor del flujo  $|f|$  de un *corte-st* es igual a su capacidad
- $f$  es un flujo máximo
- No existe ningún camino incrementable sobre  $G_f$