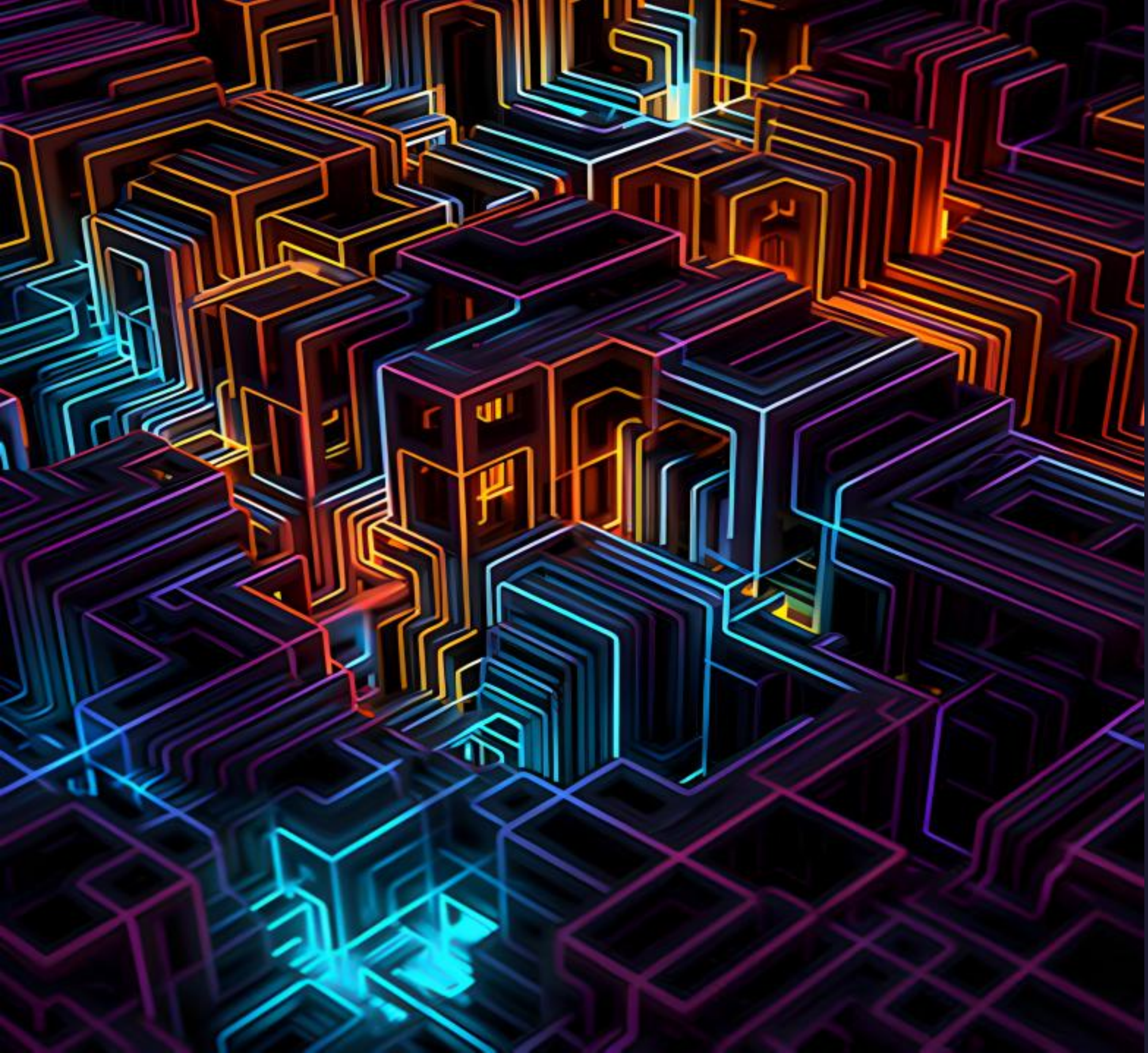


Algoritmos y problemas en grafos

Análisis y diseño de algoritmos avanzados

Dra. Valentina Narváez Terán

Tecnológico de Monterrey

Problema del flujo máximo (max flow)

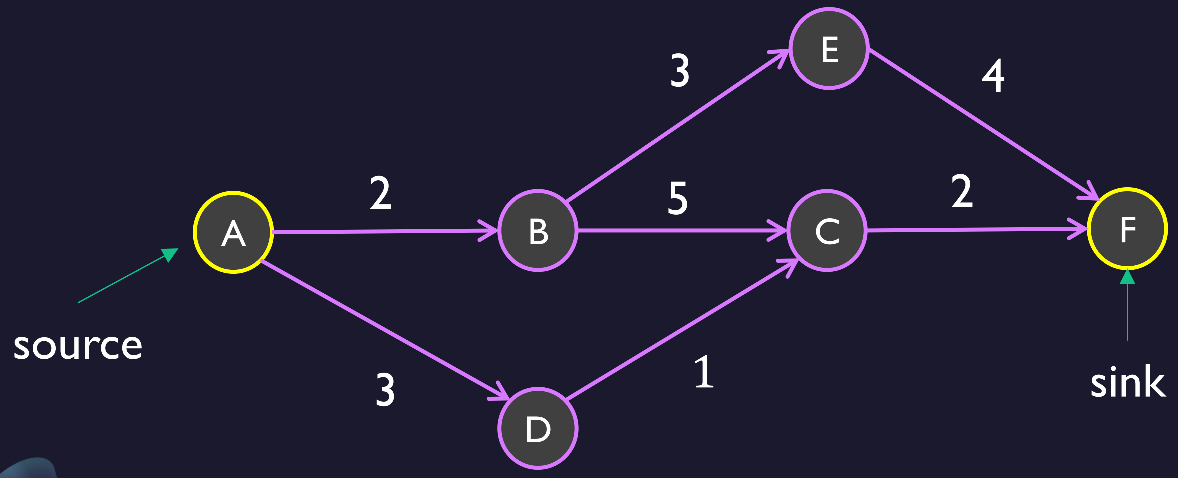
Es un problema en grafos dirigidos y ponderados

Los pesos de las aristas representan la capacidad de flujo que la arista puede soportar

- Un nodo es el origen del flujo (source)
- Otro nodo es el destino (sink)

Aplicaciones: el flujo podría representar electricidad, agua, datos… etc.

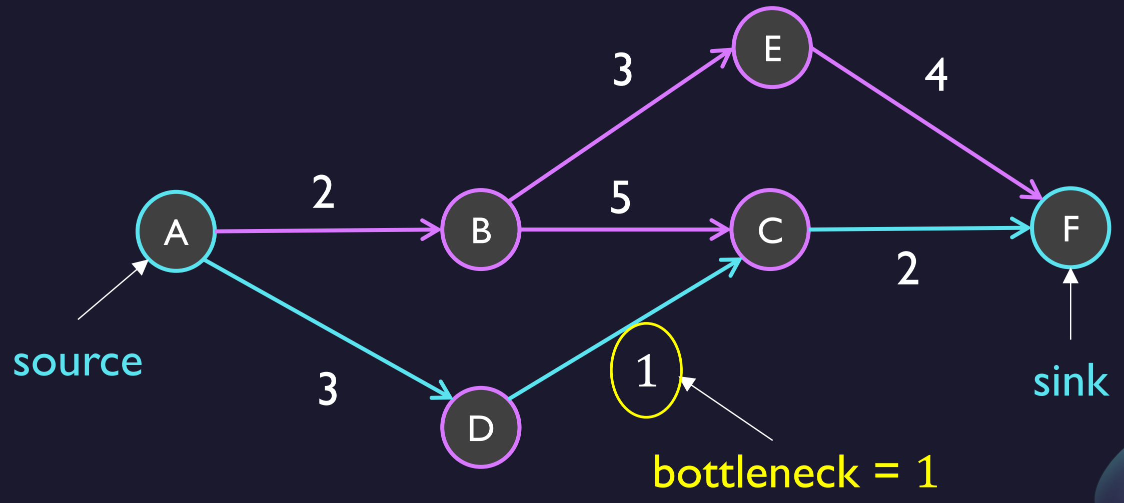
Max Flow: maximizar la cantidad de flujo recibida en el sink, sin sobrepasar la capacidad de las aristas



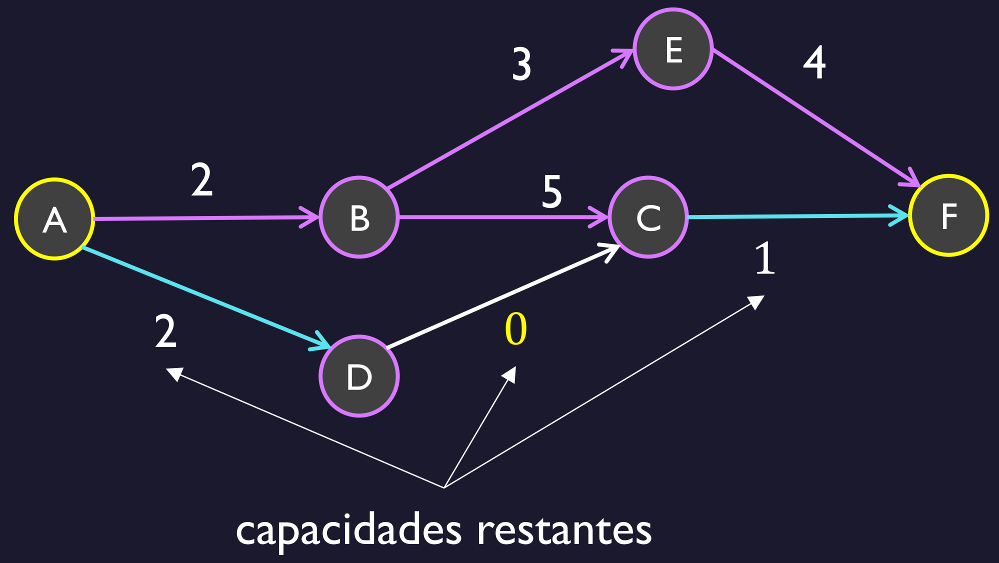
¿Cuál será la máxima cantidad de flujo que puede ir de A hasta F?

Problema del flujo máximo (max flow)

Dado un camino de source hasta sink, la menor capacidad de las aristas en el camino es un bottleneck



Al usar el camino, gastamos capacidad de las aristas que lo forman. La arista del bottleneck queda saturada, y ya no se podrá usar



Algoritmo Ford-Fulkerson

R = una copia de la matriz de adyacencia M

$maxFlow\ =\ 0$

Mientras exista un camino de $source$ a $sink$ en R:

  $path$ = camino de source a sink, formado por aristas no-saturadas

  $bottleneck$ = la menor capacidad de una arista de $path$

  Para cada arista del nodo u al nodo v en $path$:

  a) $\boxed{R[u,v]\ -=\ bottleneck}$

  b) $\boxed{R[v,u]\ +=\ bottleneck}$

  $\underbrace{maxFlow\ +=\ bottleneck}$

El flujo máximo será la acumulación de los bottlenecks de cada camino

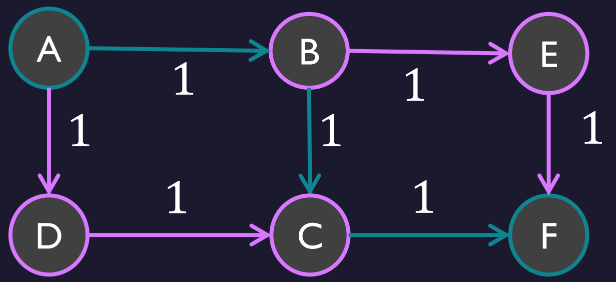
Se basa en encontrar caminos (con BFS o DFS) de source a sink, y descontar los bottlenecks de las capacidades restantes

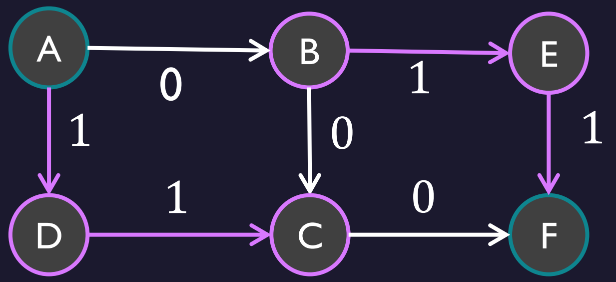
Se detiene cuando no queden caminos.

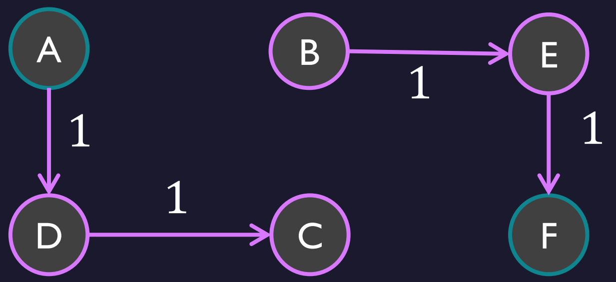
Usa una matriz (llamada residual R) para:

a) mantener actualizada la capacidad restante de las aristas…
b) y actualizar las aristas inversas (no existen en la matriz de adyacencia original. La sig diapositiva las explica)

¿Aristas inversas? ¿para que?







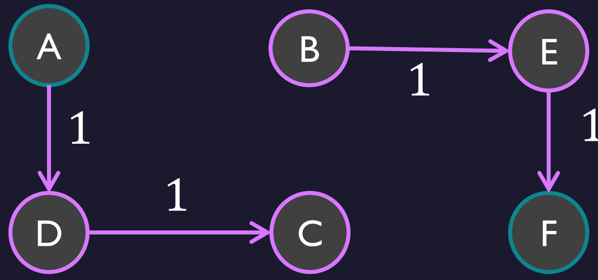
Ejemplo con un grafo sencillo, donde todas las capacidades son 1

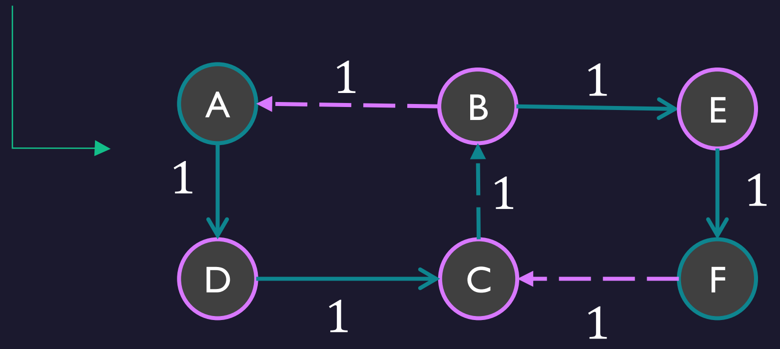
1) Digamos que el primer camino encontrado es A, B, C, F
Su bottleneck es 1 (no importa de cual arista)

2) Nota que sus aristas quedarían **saturadas** e inutilizables,
como si no existieran. Nota que en la matriz R se volverán 0's

3) Entonces, no podríamos encontrar mas caminos de A a F

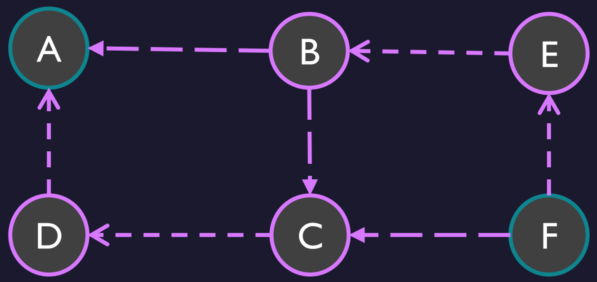
¿Aristas inversas? ¿para que?





4) Agregando las aristas inversas, podemos encontrar el camino A,D,C,B,E,F cuyo bottleneck es 1

5) Nuevamente actualizamos las aristas inversas por cada arista del camino A,D,C,B,E,F

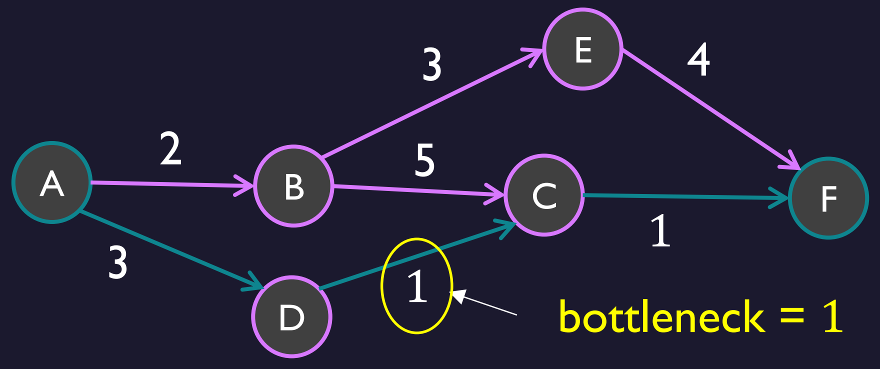


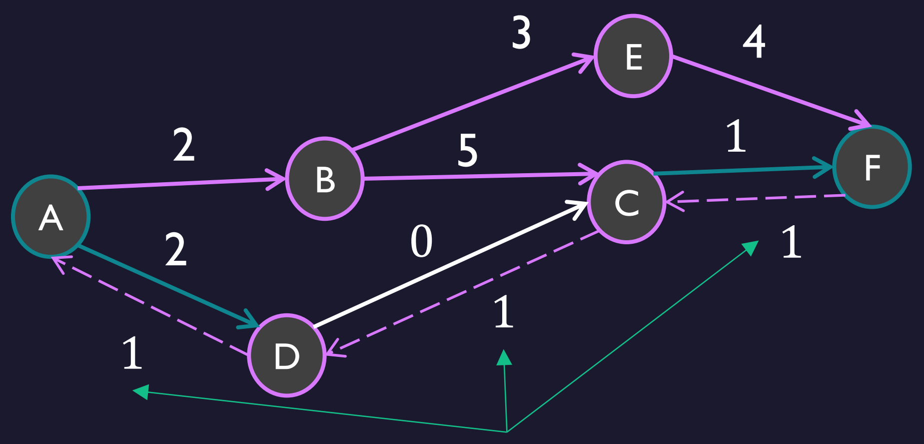
6) En este punto ya no hay caminos de A a F. Entonces, terminamos.

7) El flujo máximo es la acumulación de los bottlenecks de cada camino encontrado.

Para este ejemplo, es maxFlow = 2

Algoritmo Ford-Fulkerson





La suma de la arista real, y su respectiva arista inversa es la capacidad original

Este proceso de actualizar las aristas y las aristas inversas se conoce como *path augmentation*

Para cada arista del nodo u al nodo v en $path$:

$$R[u,v] \ -= \ bottleneck$$
$$R[v,u] \ += \ bottleneck$$

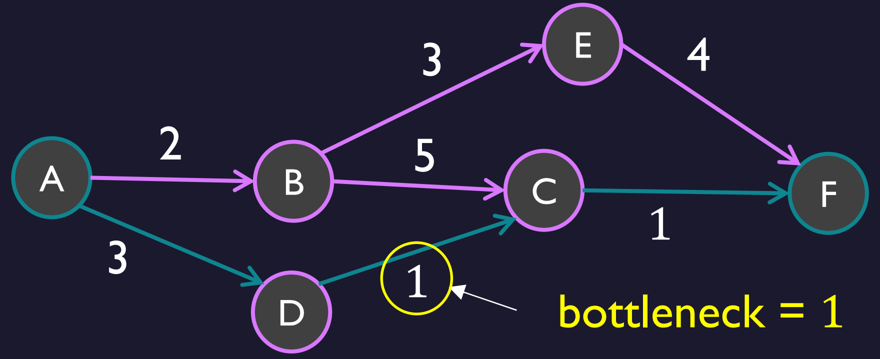
	A	B	C	D	E	F
A		2		3		
B			5		3	
C						2
D			1			
E						4
F						

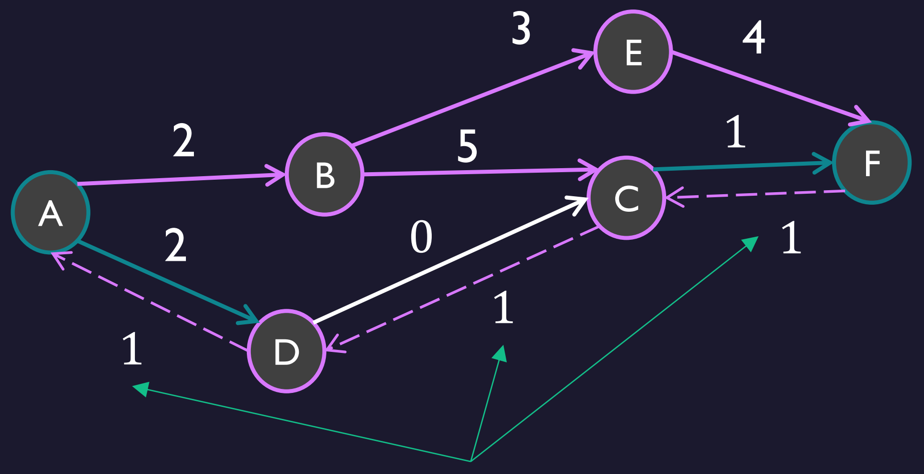
Matriz residual R

	A	B	C	D	E	F
A		2		2		
B			5		3	
C				1		1
D	1		0			
E						4
F			1			

R actualizada

Algoritmo Ford-Fulkerson





La suma de la arista, y la inversa
es la capacidad original

El proceso de actualizar las capacidades de las aristas y las aristas inversas se conoce como *path augmentation*

Para cada arista del nodo u al nodo v en $path$:

$$R[u, v] \ -= \ bottleneck$$
$$R[v, u] \ += \ bottleneck$$

	A	B	C	D	E	F
A		2		3		
B			5		3	
C						2
D			1			
E						4
F						

Matriz residual R

	A	B	C	D	E	F
A		2		2		
B			5		3	
C				1		1
D	1		0			
E						4
F			1			

R actualizada

Edmon-Karp = Ford-Fulkerson + BFS

$q\ =$ una fila (queue)
El nodo $source$ entra en la fila q
Mientras q no este vacía:
    $u\ =$ nodo que sale de la fila
    Para j de 0 a n:
        Si $R[u][j] > 0$ and $visitado[j] = 0$:
            $label[j]\ =\ R[u][j]$
            $visitado[j]\ =\ 1$
            $predecesor[j]\ =\ u$
            Agregar j a la fila q
$i = sink$
Mientras $true$:
    Añade i a $path$, y añade $label[i]$ a $flujo$
    Si $i\ ==\ source$:
        break
    $i\ =\ predecesor[i]$
$flujo.reverse()$
$path.reverse()$
return $path, \min(flujo[1{:}n])$

Hay diferentes formas de encontrar los caminos.
El algoritmo de Edmon-Karp lo hace con BFS

$predecesor$, $visitado$ y $labels$ son listas/arreglos de tamaño n e inician en $-1's$

$path$ y $flujo$ inician como listas/arreglos vacíos
El camino se reconstruye en reversa desde $sink$, usando $predecesor$

$\min(flujo[1{:}n])$ es el bottleneck