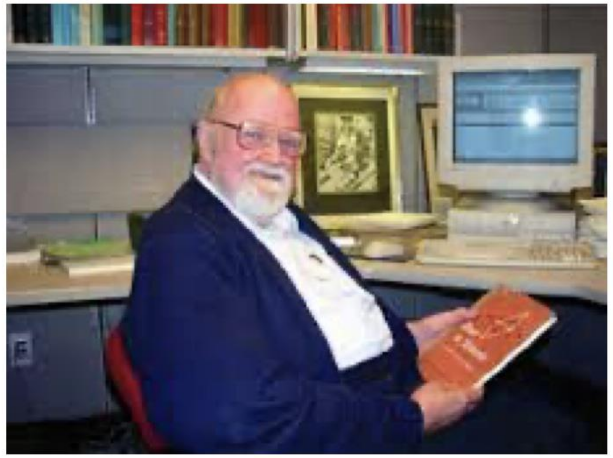
Algoritmo de Flujo Máximo
Método de Ford-Fulkerson.
Variante Edmonds-Karp

L.R. Ford, D.R. Fulkerson:
Maximal flow through a network. Canadian J. of Math. 1956.



Dado un digrafo con pesos positivos G y dos nodos destacados S (fuente) y T (sumidero), podemos interpretar cada peso de los arcos del grafo como la cantidad de material que se puede trasladar desde el origen al extremo del arco por unidad de tiempo (capacidad o velocidad de flujo).
La fuente S produce material y el sumidero T lo consume. El resto de arcos se pueden interpretar como conductos con la capacidad indicada por el peso y los nodos como distribuidores de material.

Ejemplos:

Los nodos podrían representar distribuidores de agua, y los pesos de los arcos la cantidad de agua que se puede trasvasar del origen al extremo de cada arco en un segundo.

Los nodos podrían ser nodos de conmutación de una red de datos y los pesos de los arcos la cantidad de gigabits que se pueden trasladar del origen al extremo de cada arco por segundo.

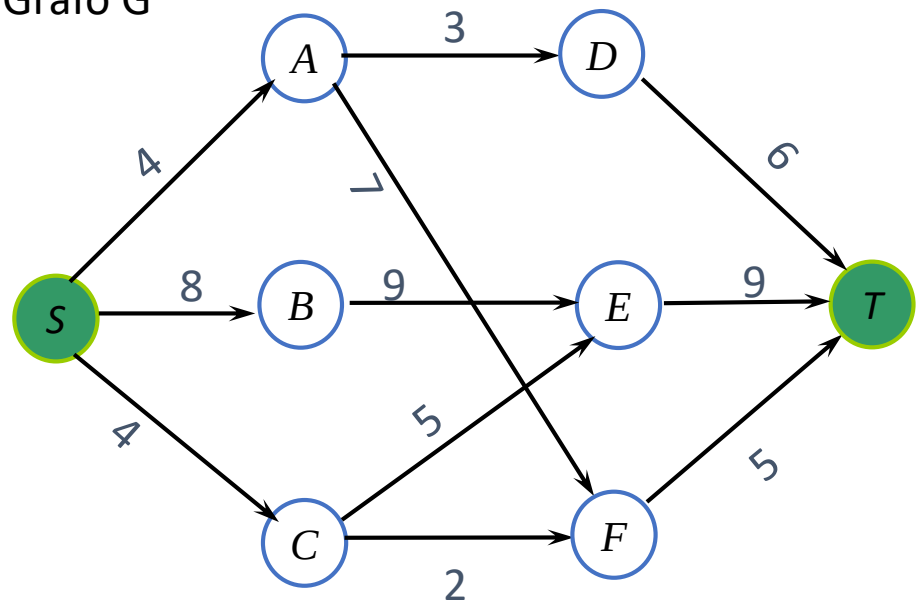
Problema:
Calcular la cantidad máxima de material por unidad de tiempo que se puede trasladar de S a T y por cuáles nodos se podría trasladar.

Definición: Dado un camino de A a B en un grafo G, se llama *flujo máximo del camino* al mínimo peso de todos los arcos que componen el camino. Por ejemplo, el flujo máximo de [a-5->b, b-7->c, c-9->d, d-2->f] es 2.

Definición: Dado un camino, se llama *camino contrario* al mismo camino pero recorrido del final al principio, y con todos los arcos con sentido contrario. Por ejemplo, el contrario de [a-5->b, b-7->c, c-9->d, d-2->f] es [f-2->d, d-9->c, c-7->b, b-5->a].

Método de Ford-Fulkerson

Grafo G



sol = []

Inicialmente se crea una lista de arcos vacía para representar la solución.

Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

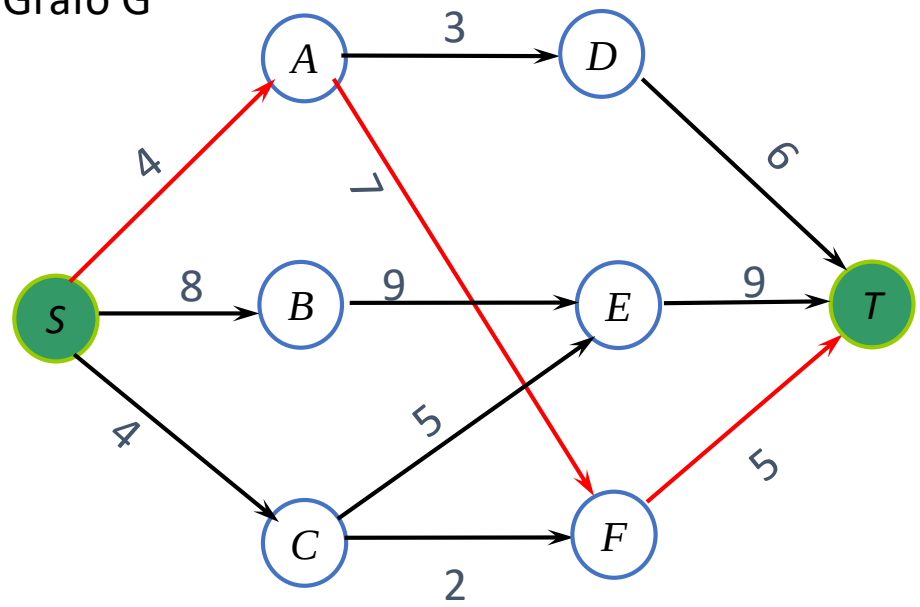
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.

La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->A, A-7->F, F-5->T]

sol = []

Inicialmente se crea una lista de arcos vacía para representar la solución.

Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

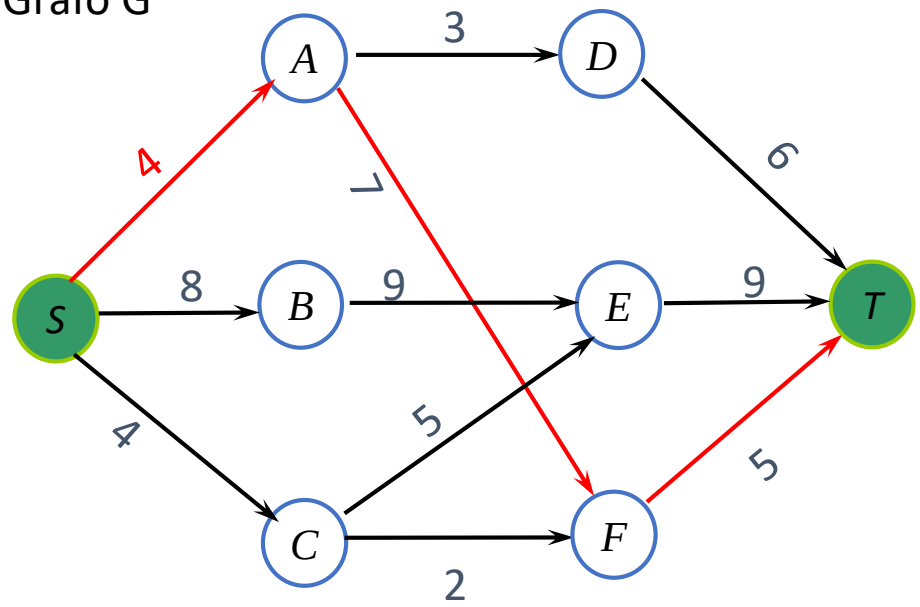
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.

La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->A, A-7->F, F-5->T]

mf = 4

sol = []

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

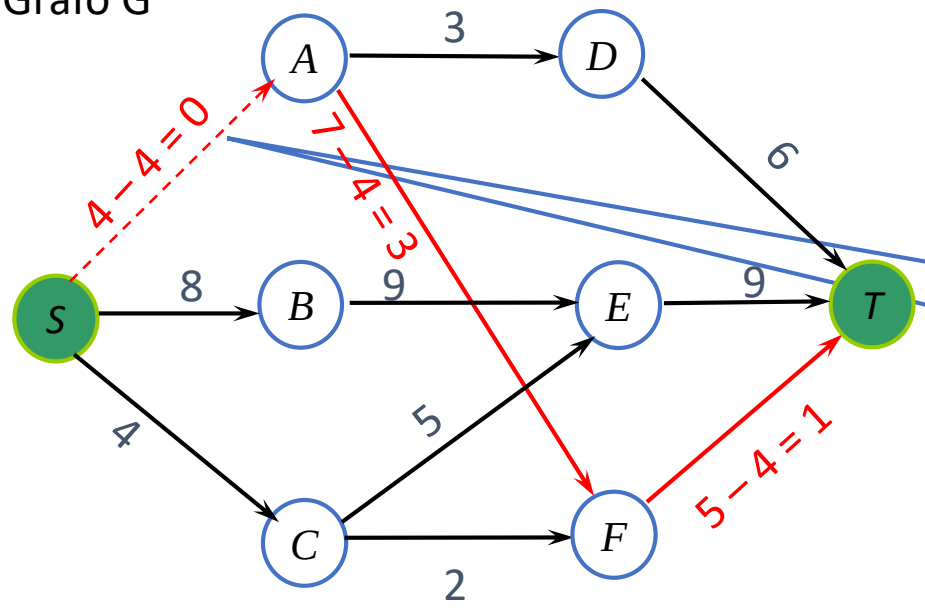
Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->A, A-7->F, F-5->T]

mf = 4

sol = []

Si el peso resultante es 0, el arco se elimina del grafo. Véase la definición posterior de la función `updateEdge` para los detalles.

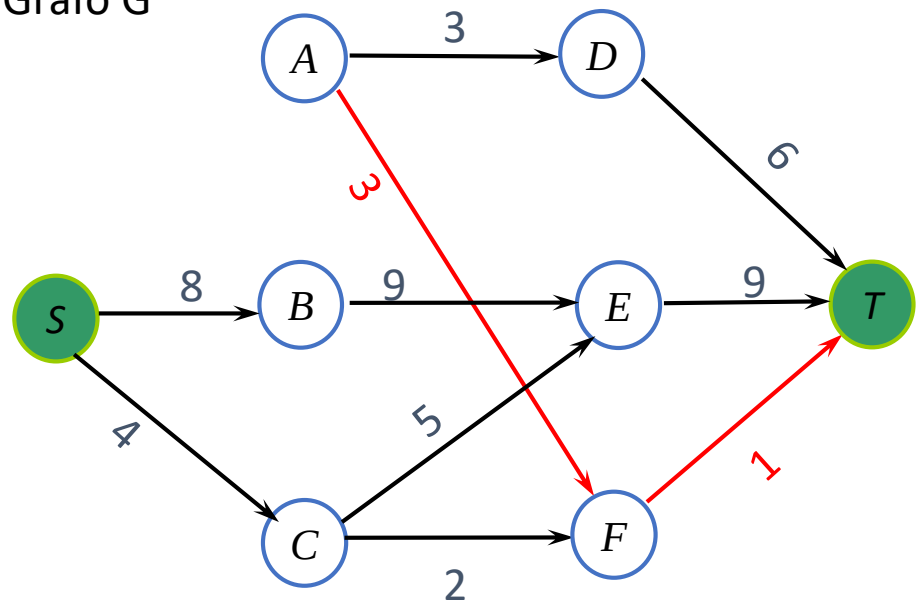
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->A, A-7->F, F-5->T]

mf = 4

sol = []

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

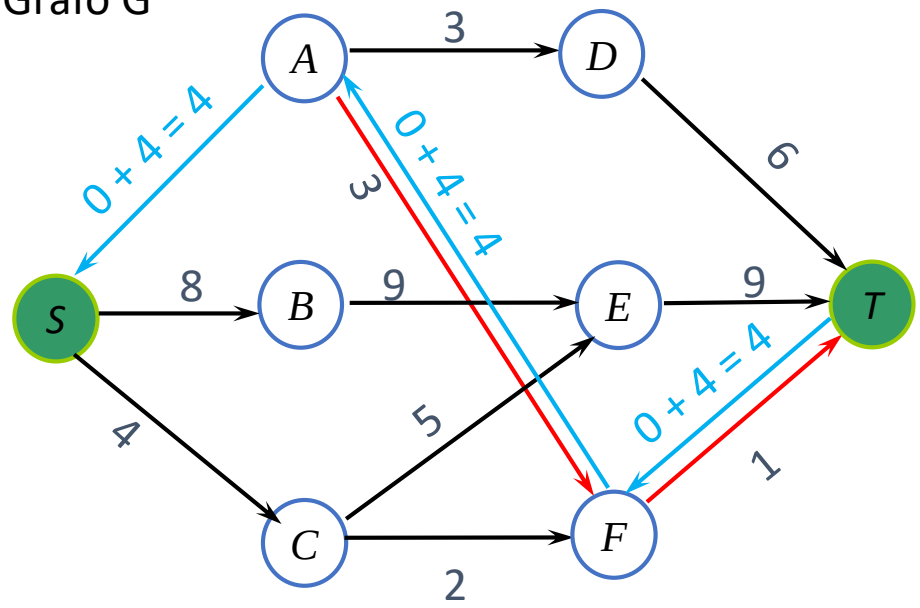
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->A, A-7->F, F-5->T]

mf = 4

sol = []

Si dicho arco no existía previamente en el grafo, su peso se considera 0 y, por tanto, el arco se añade al grafo con peso mf. Véase la definición posterior de la función `updateEdge` para los detalles.

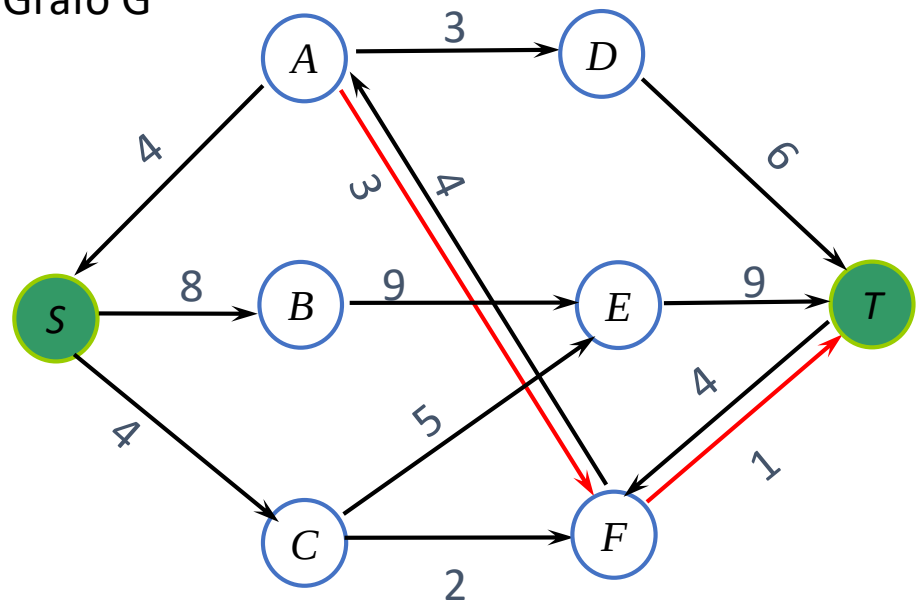
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->A, A-7->F, F-5->T]

mf = 4

sol = [S-4->A, A-4->F, F-4->T]

flow = [S-4->A, A-4->F, F-4->T]

Véase la definición posterior de la función `addFlow` para los detalles.

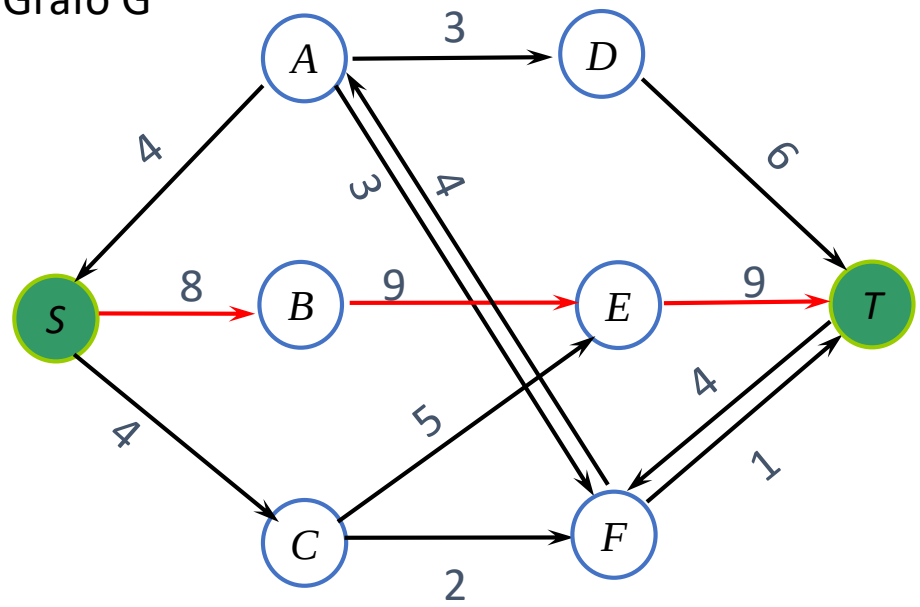
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).
Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-8->B, B-9->E, E-9->T]

sol = [S-4->A, A-4->F, F-4->T]

Inicialmente se crea una lista de arcos vacía para representar la solución.

Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

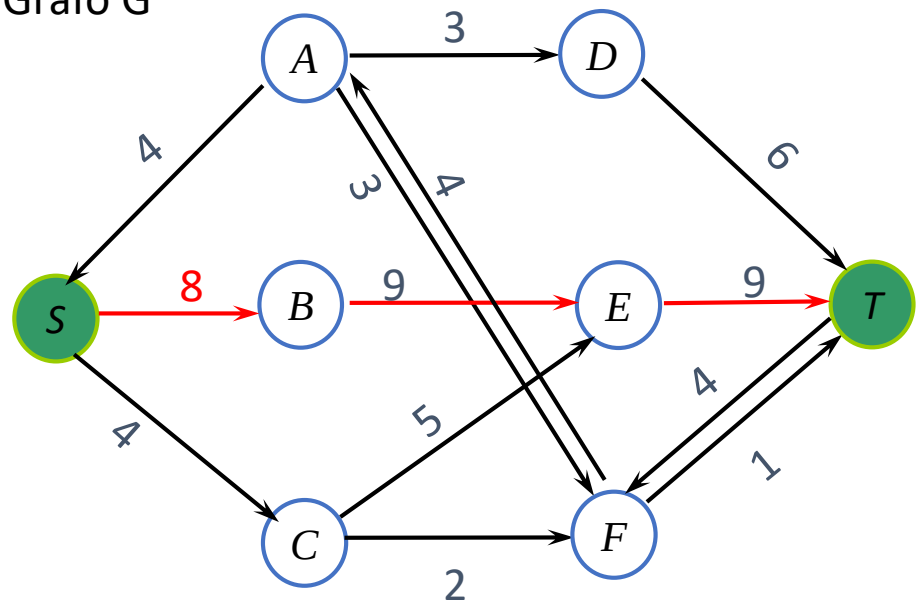
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.

La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-8->B, B-9->E, E-9->T]

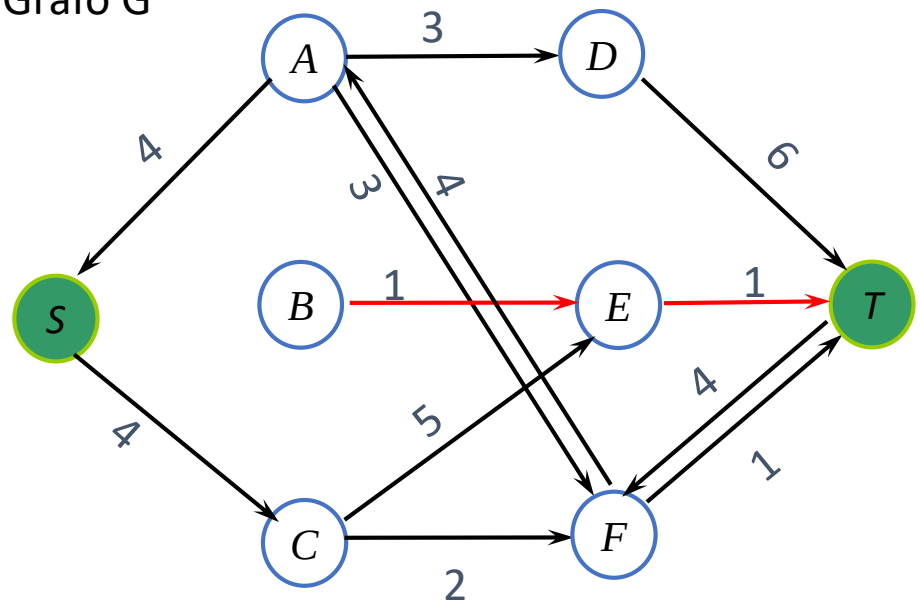
mf = 8

sol = [S-4->A, A-4->F, F-4->T]

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:
Se calcula el flujo máximo (mf) de dicho camino (path).
Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.
La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-8->B, B-9->E, E-9->T]

mf = 8

sol = [S-4->A, A-4->F, F-4->T]

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

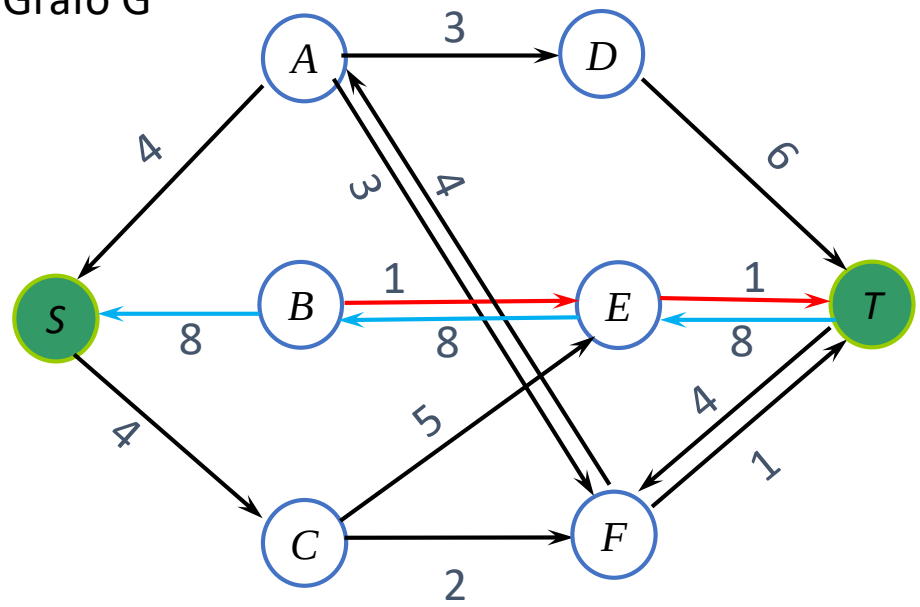
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-8->B, B-9->E, E-9->T]

mf = 8

sol = [S-4->A, A-4->F, F-4->T]

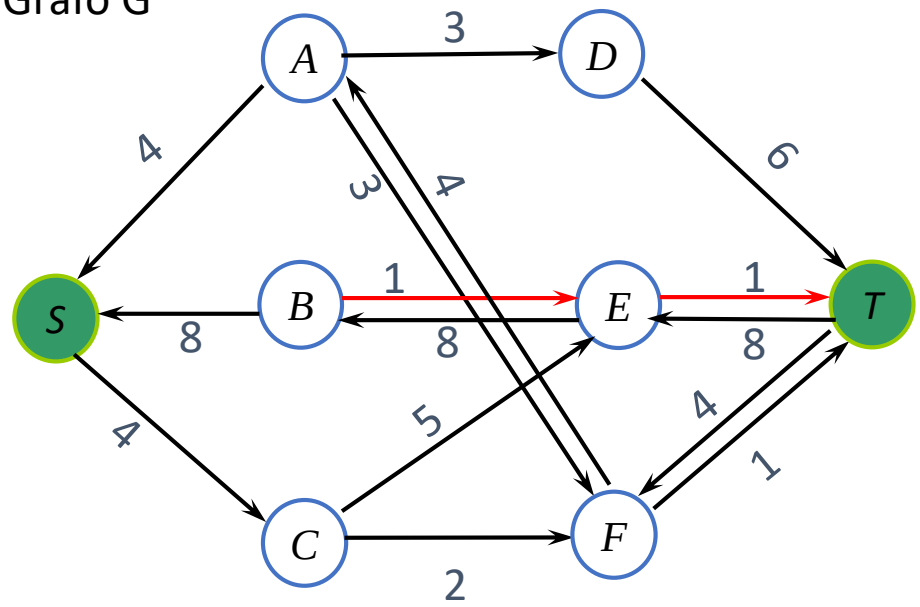
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-8->B, B-9->E, E-9->T]

mf = 8

sol = [S-4->A, A-4->F, F-4->T, S-8->B, B-8->E, E-8->T]

flow = [S-8->B, B-8->E, E-8->T]

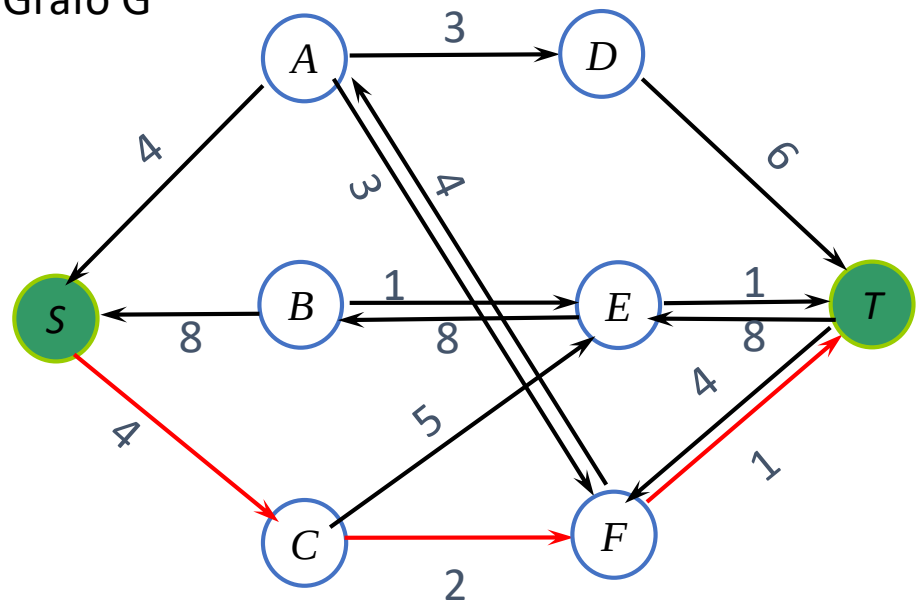
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).
Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->C, C-2->F, F-1->T]

sol = [S-4->A, A-4->F, F-4->T, S-8->B, B-8->E, E-8->T]

Inicialmente se crea una lista de arcos vacía para representar la solución.

Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

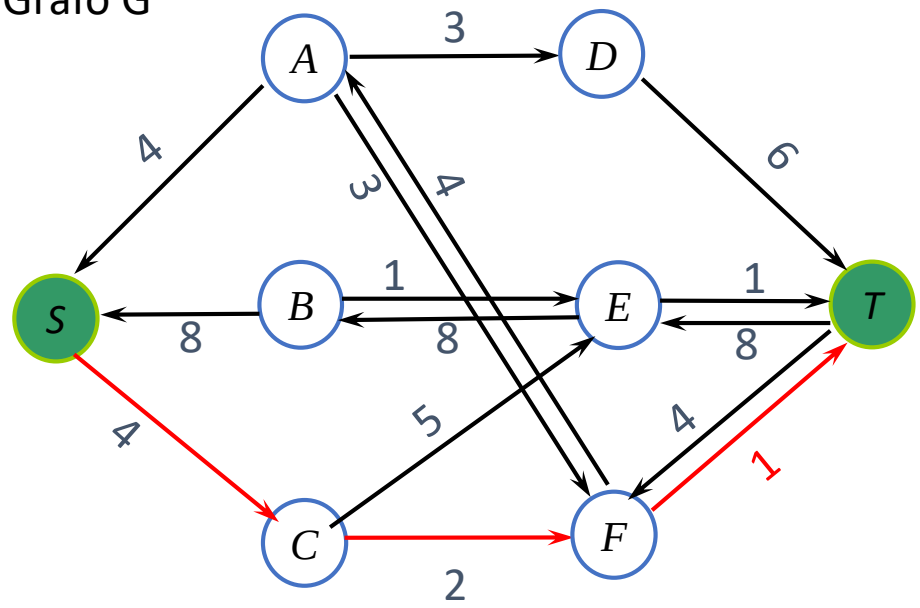
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.

La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->C, C-2->F, F-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-4->T, S-8->B, B-8->E, E-8->T]

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

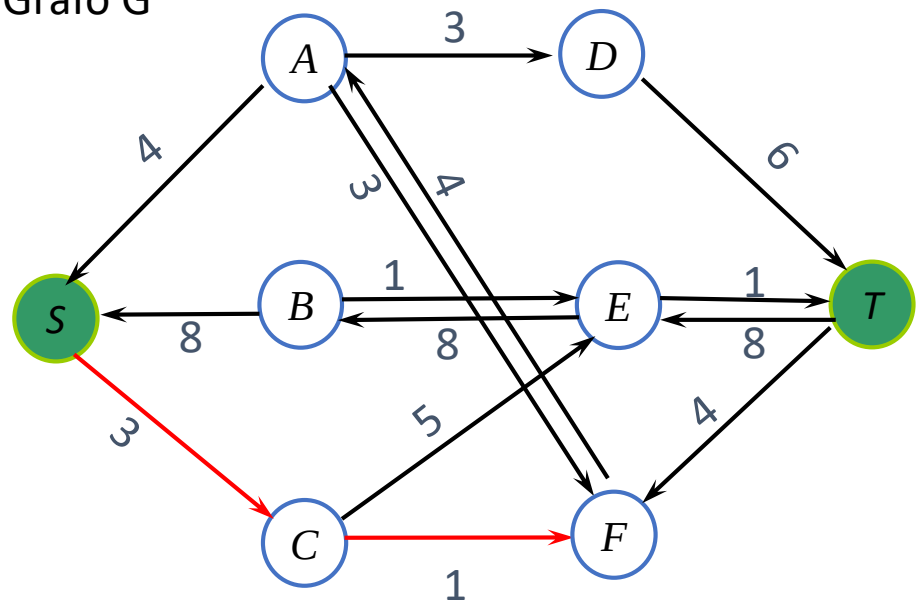
Se calcula el flujo máximo (mf) de dicho camino(path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->C, C-2->F, F-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-4->T, S-8->B, B-8->E, E-8->T]

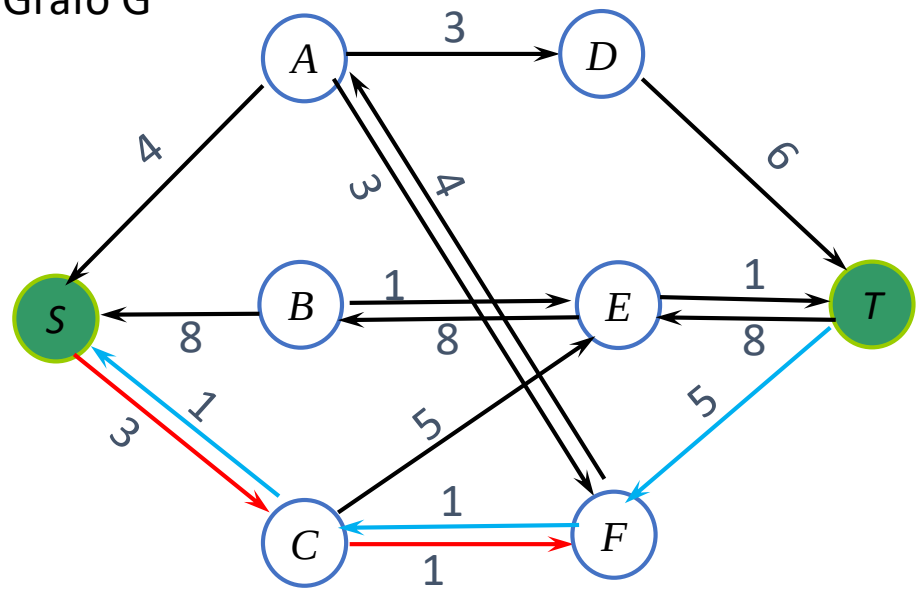
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->C, C-2->F, F-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-4->T, S-8->B, B-8->E, E-8->T]

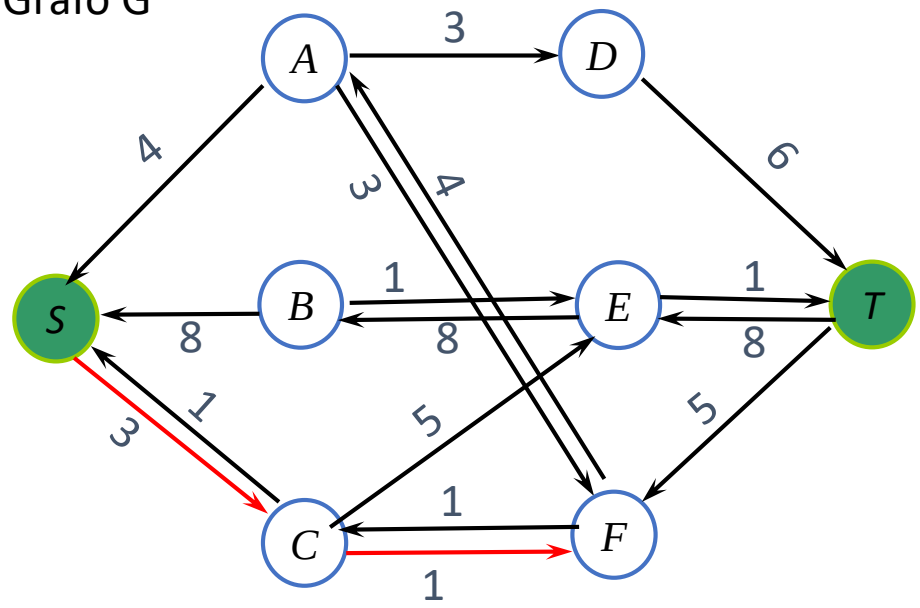
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-4->C, C-2->F, F-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-8->T, S-1->C, C-1->F]

flow = [S-1->C, C-1->F, F-1->T]

Al añadir el flujo F-1->T a F-4->T se obtiene F-5->T

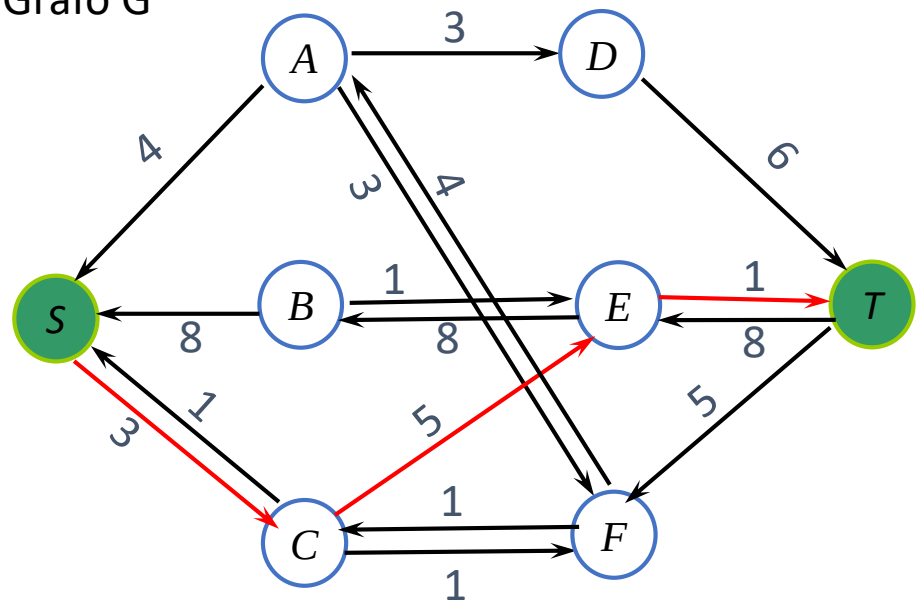
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).
Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-3->C, C-5->E, E-1->T]

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-8->T, S-1->C, C-1->F]

Inicialmente se crea una lista de arcos vacía para representar la solución.

Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

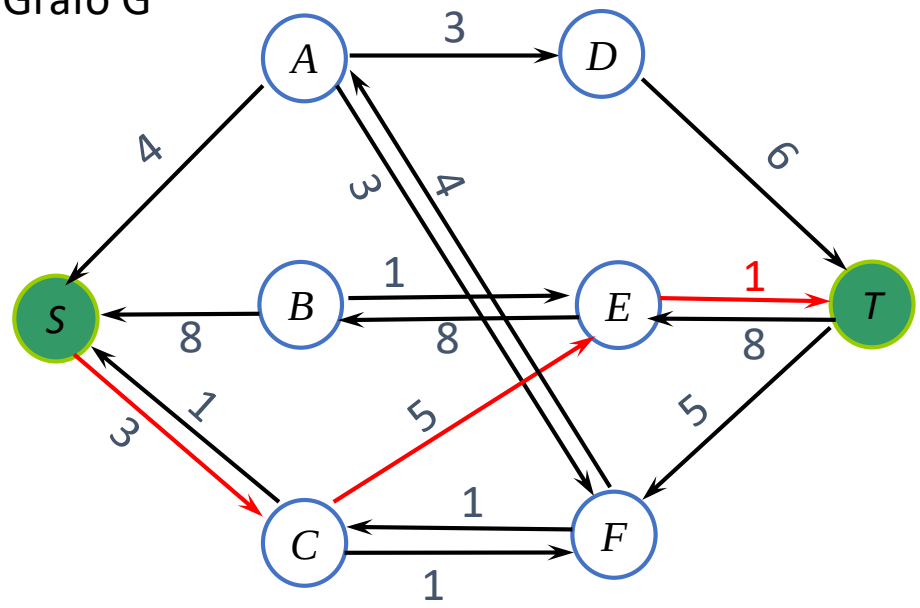
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.

La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-3->C, C-5->E, E-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-8->T, S-1->C, C-1->F]

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

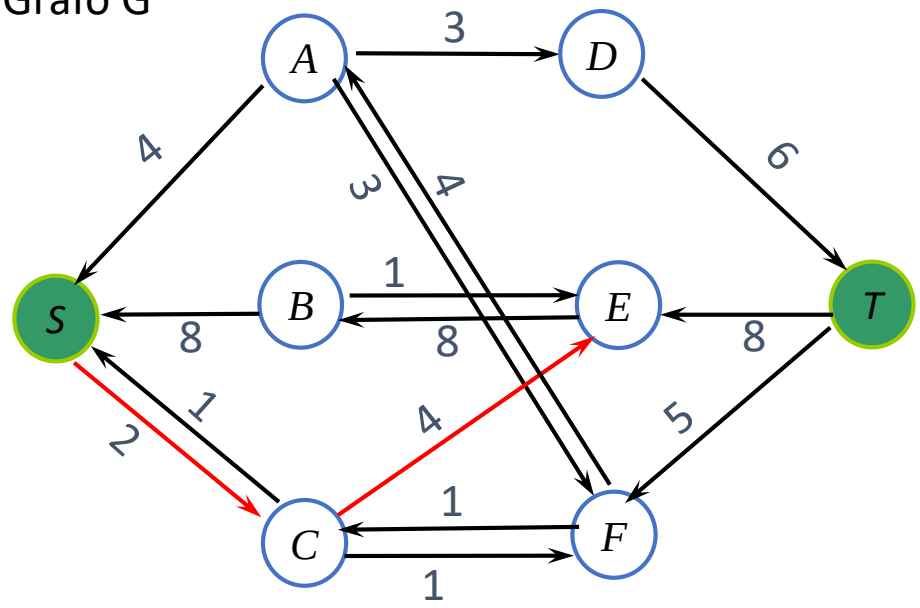
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-3->C, C-5->E, E-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-8->T, S-1->C, C-1->F]

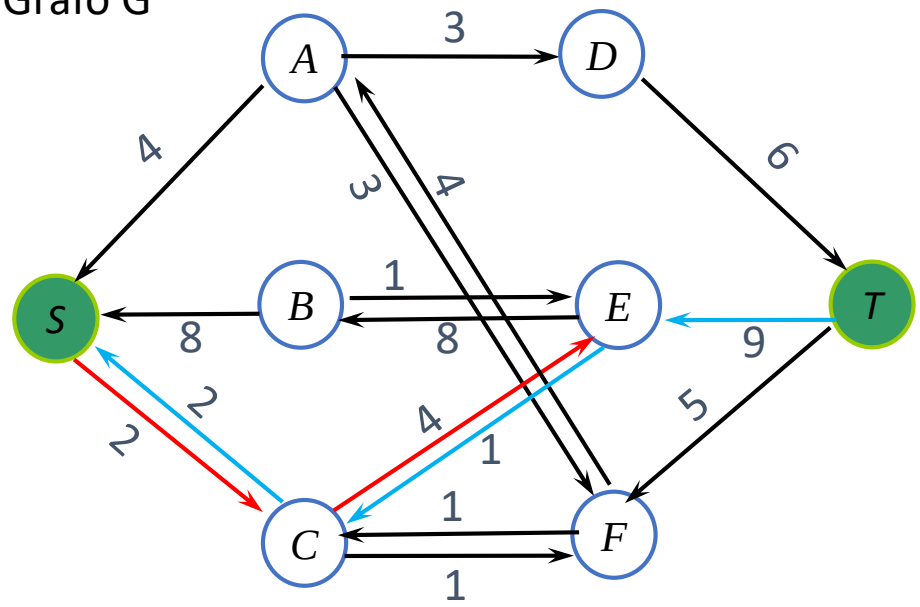
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-3->C, C-5->E, E-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-8->T, S-1->C, C-1->F]

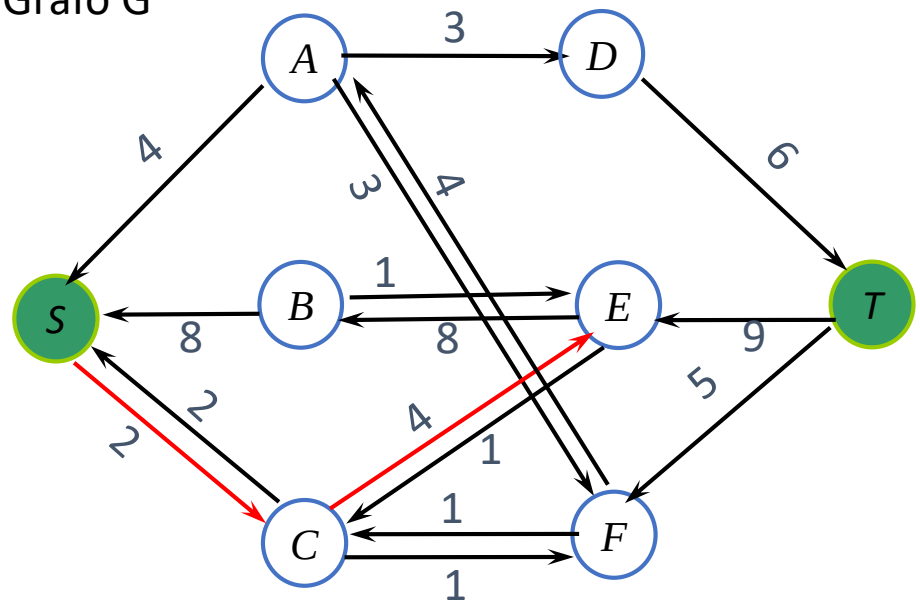
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-3->C, C-5->E, E-1->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-9->T, S-2->C, C-1->F, C-1->E]

flow = [S-1->C, C-1->E, E-1->T]

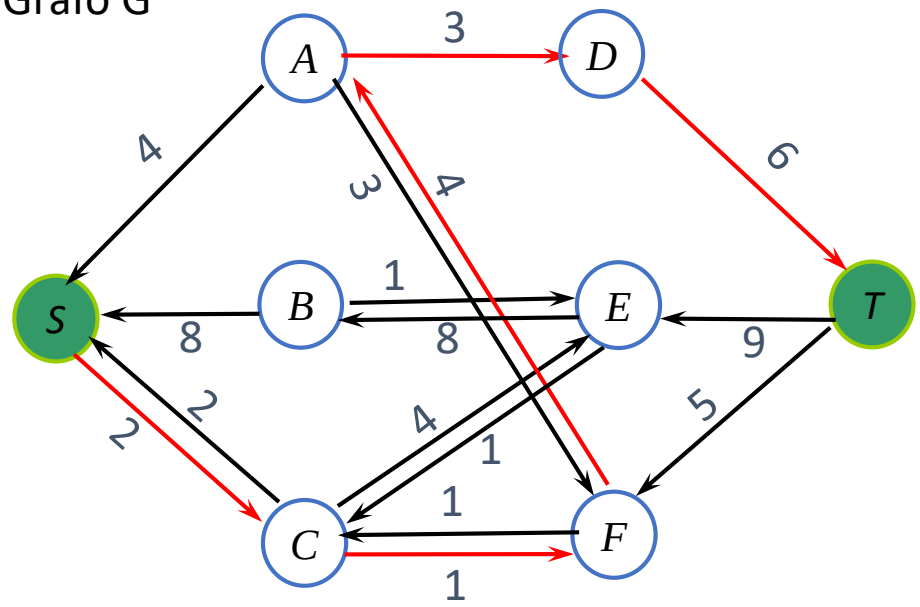
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).
Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-2->C, C-1->F, F-4->A, A-3->D, D-5->T]

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-9->T, S-2->C, C-1->F, C-1->E]

Inicialmente se crea una lista de arcos vacía para representar la solución.

Mientras haya un camino (path) de S a T en G:

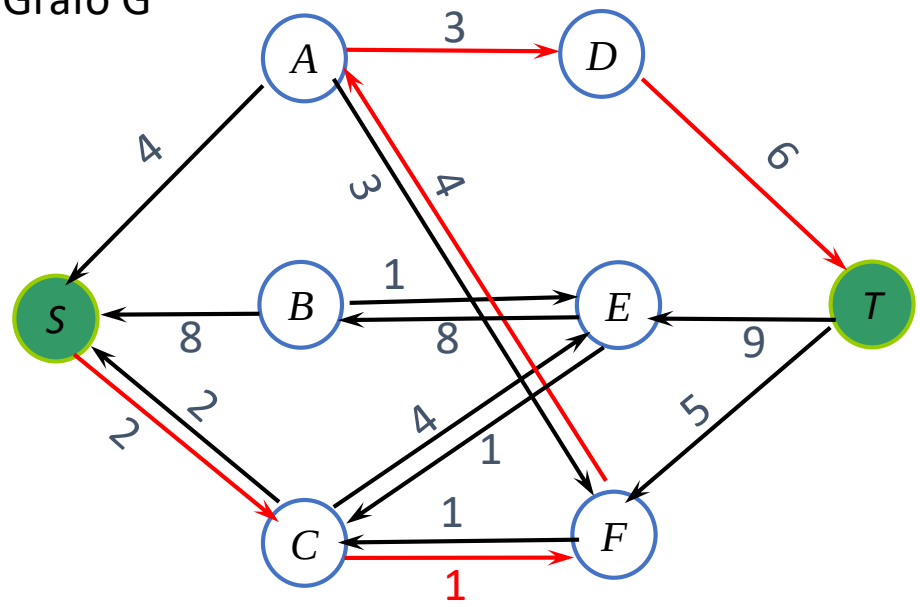
- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.

La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-2->C, C-1->F, F-4->A, A-3->D, D-5->T]

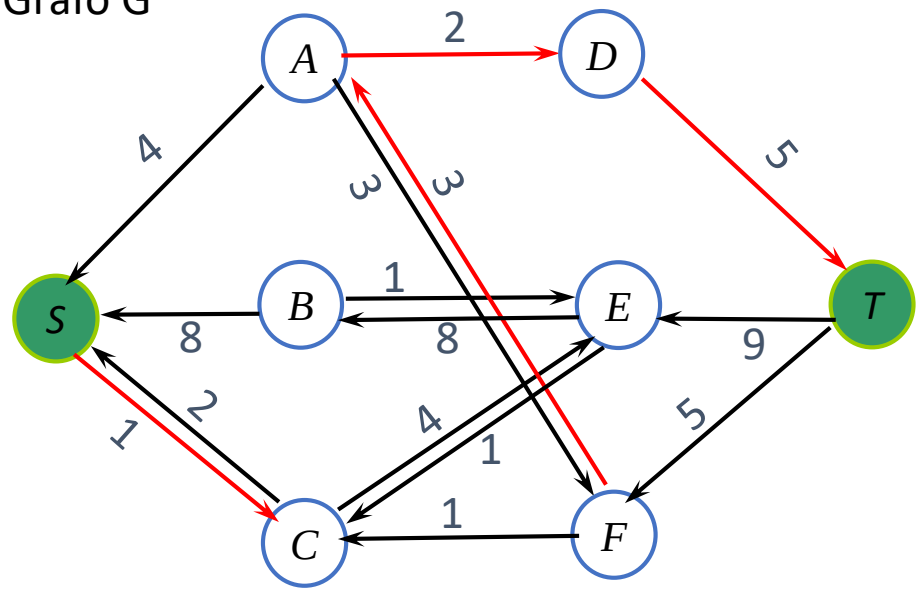
mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-9->T, S-2->C, C-1->F, C-1->E]

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:
Se calcula el flujo máximo (mf) de dicho camino (path).
Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.
La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-2->C, C-1->F, F-4->A, A-3->D, D-5->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-9->T, S-2->C, C-1->F, C-1->E]

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

Se calcula el flujo máximo (mf) de dicho camino (path).

Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.

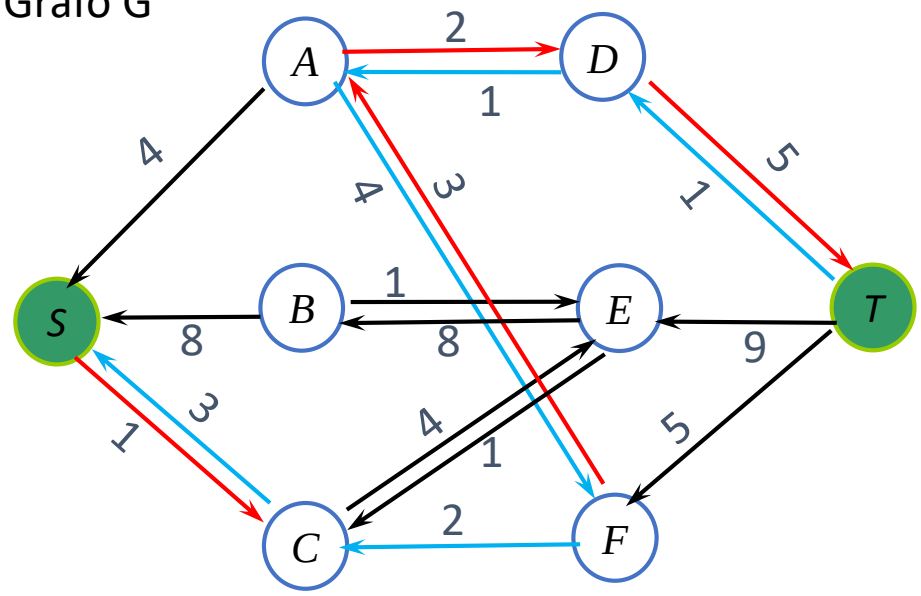
Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.

A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-2->C, C-1->F, F-4->A, A-3->D, D-5->T]

mf = 1

sol = [S-4->A, A-4->F, F-5->T, S-8->B, B-8->E, E-9->T, S-2->C, C-1->F, C-1->E]

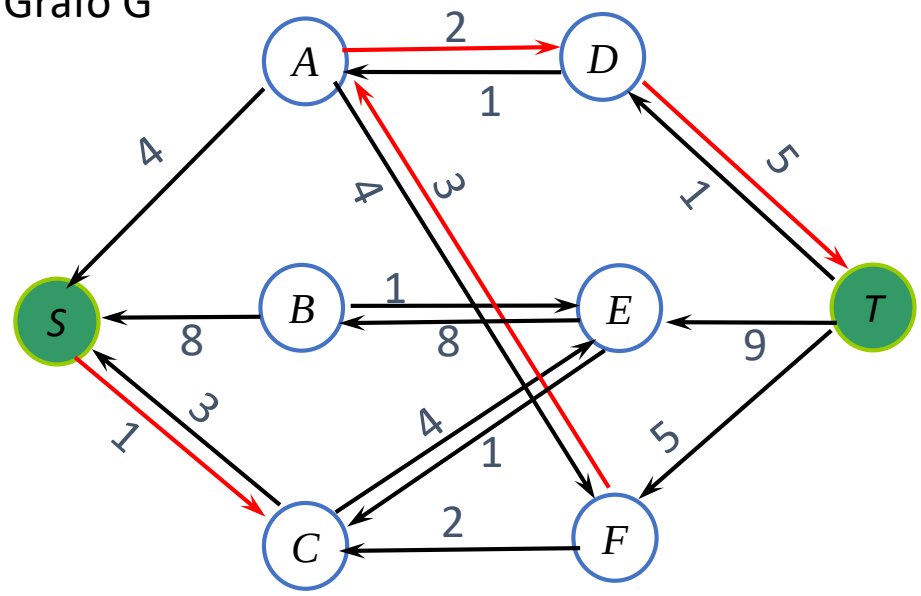
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



path = [S-2->C, C-1->F, F-4->A, A-3->D, D-5->T]

mf = 1

sol = [S-4->A, A-3->F, F-5->T, S-8->B, B-8->E, E-9->T, S-3->C, C-2->F, C-1->E, A-1->D, D-1->T]

flow = [S-1->C, C-1->F, F-1->A, A-1->D, D-1->T]

Al añadir el flujo F-1->A a A-4->F se obtiene A-3->F

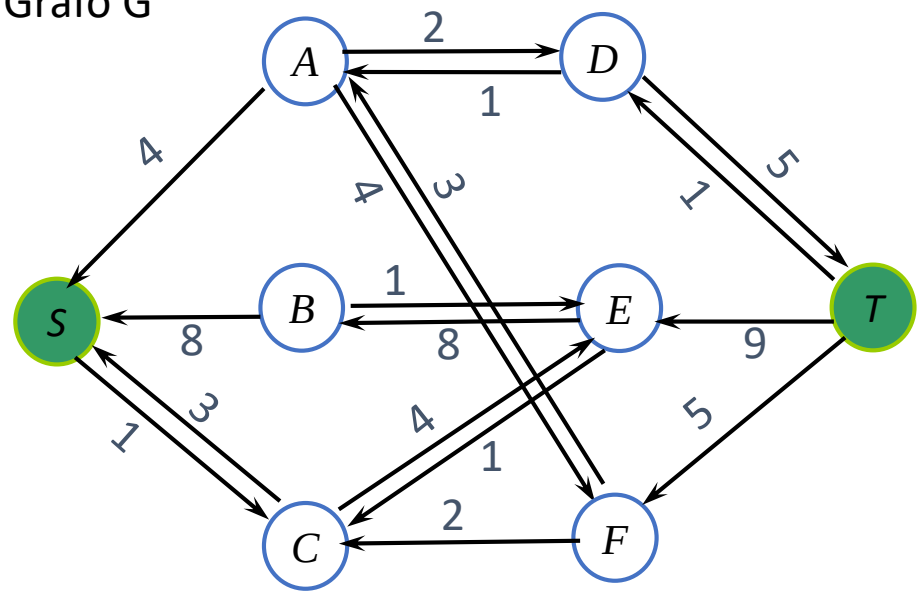
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



sol = [S-4->A, A-3->F, F-5->T, S-8->B, B-8->E, E-9->T, S-3->C, C-2->F, C-1->E, A-1->D, D-1->T]

flujo máximo = 4 + 8 + 3 = 15

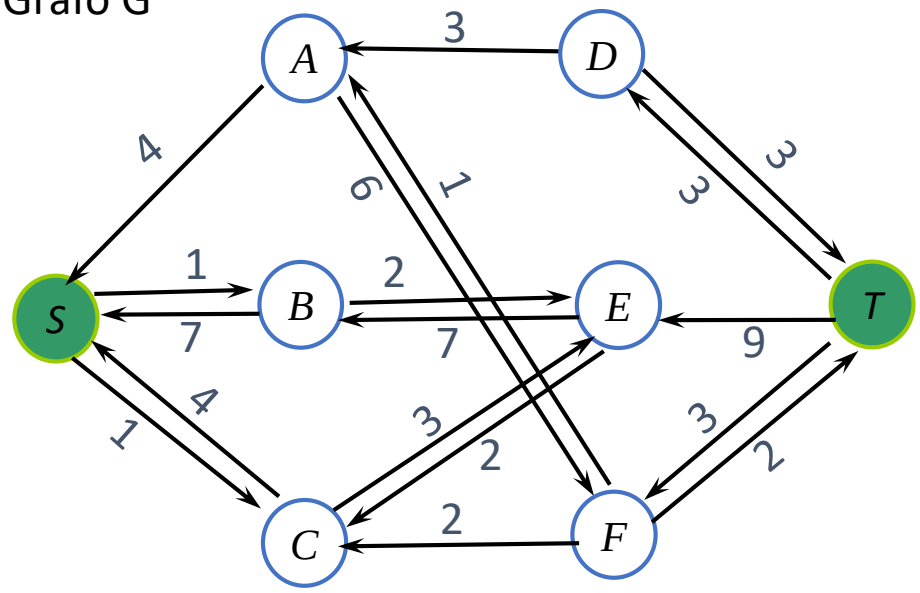
Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Grafo G



La solución puede no ser única. Depende de los caminos seleccionados en cada paso

sol = [S-4->A, A-3->D, D-3->T, A-1->F, F-3->T, S-4->C, C-2->F, C-2->E, E-9->T, S-7->B, B-7->E]

flujo máximo = 4 + 4 + 7 = 15

El máximo flujo sí es siempre el mismo

Inicialmente se crea una lista de arcos vacía para representar la solución.
Mientras haya un camino (path) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (path).
- Para cada arco del grafo G que forme parte del camino (path), se decrementa su peso en mf unidades.
- Para cada arco del grafo G que forme parte del camino contrario a path, se incrementa su peso en mf unidades.
- A la lista solución se le añade el flujo de los arcos del camino (path) pero con peso mf.

La lista solución contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución que salen de S será el flujo máximo de la red.

Método de Ford-Fulkerson

Como añadir, buscar y eliminar arcos de un digrafo con pesos puede ser complicado, modificamos el algoritmo para, en cada paso, extraer la lista de arcos del grafo, realizar las operaciones sobre esa lista y por último volver a construir el grafo con los arcos de la lista.
Si al operar sobre la lista, un arco queda con peso 0, se elimina de la lista antes de construir el nuevo grafo.

Inicialmente se crea una lista SOL de arcos vacía para representar la solución.
Mientras haya un camino (P) de S a T en G:

- Se calcula el flujo máximo (mf) de dicho camino (P).
- Se extrae la lista L de arcos dirigidos del grafo G.
- Para cada arco de la lista L que forme parte del camino (P), se decrementa su peso en mf unidades.
- Para cada arco de la lista L que forme parte del camino contrario a P, se incrementa su peso en mf unidades.
- Se genera un nuevo grafo G con los mismos vértices y con los arcos de la lista L.
- A la lista solución SOL se le añade el flujo de los arcos del camino (P) pero con peso mf.

La lista solución (SOL) contendrá los arcos que forman el grafo con flujo máximo.
La suma de los pesos de los arcos de la solución (SOL) que salen de S será el flujo máximo de la red.

Haskell

Trabajaremos con digrafos con pesos **WeightedDiGraph a w** con las operaciones usuales de digrafos:

```
data WDiEdge a w = E a w a deriving Eq   E "A" 4 "B"  representa el arco "A" - 4 -> "B"
type Path a = [a] Path representa un camino

mkWeightedDiGraphSuc :: (Eq a) => [a] -> (a -> [(a,w)]) -> WeightedDiGraph a w
mkWeightedDiGraphEdges :: (Eq a) => [a] -> [WDiEdge a w] -> WeightedDiGraph a w
successors :: WeightedDiGraph a w -> a -> [(a,w)]
predecessors :: (Eq a) => WeightedDiGraph a w -> a -> [(a,w)]
vertices :: WeightedDiGraph a w -> [a]
weightedDiEdges :: WeightedDiGraph a w -> [WDiEdge a w]
outDegree :: WeightedDiGraph a w -> a -> Int
inDegree :: (Eq a) => WeightedDiGraph a w -> a -> Int
```

Usaremos un algoritmo de búsqueda en anchura (ya definido) para hallar un camino entre dos vértices:

```
bftPathTo :: (Ord a) => WeightedDiGraph a w -> a -> a -> Maybe (Path (WDiEdge a w))
```

que, dados un digrafo con pesos, una fuente y un destino, proporciona o bien un camino de la fuente al destino con `Just`, o el valor `Nothing` si tal camino no existe.

Las siguientes funciones se deberán implementar en el examen:

```
Devuelve el flujo máximo de un camino (menor peso de los arcos que componen el camino).
@param path    Camino donde se busca el flujo máximo.
@return        Flujo máximo.
maxFlowPath :: Path (WDiEdge a Integer) -> Integer
```

0.75 puntos

```
Sea edges una lista de arcos dirigidos con peso donde no hay dos arcos con la misma fuente
y el mismo destino.
updateEdge crea una lista de arcos a partir de los arcos de edges del siguiente modo:
     Si un arco x-w->y está en edges, se incrementa su peso en p unidades (p puede ser
     negativo). Si tras incrementarlo, el peso resultante es 0, se elimina el arco de
     la lista resultado.
     Si no hay un arco x-w->y en edges, se agrega éste pero con peso p.
     Los demás arcos de edges se mantienen en el resultado.
@param x     Origen del arco a actualizar.
@param y     Destino del arco a actualizar.
@param p     Peso para asignar o con el que incrementar el arco.
@param edges Lista de arcos de entrada.
@return      Una nueva lista con el arco entre x e y descrito y los demás arcos de edges.
updateEdge :: (Eq a) => a -> a -> Integer -> [WDiEdge a Integer] -> [WDiEdge a Integer]
```

1 punto

Las siguientes funciones se deberán implementar en el examen:

```
Sea edges una lista de arcos dirigidos con peso donde no hay dos arcos con la misma fuente
y el mismo destino. Sea path otra lista de arcos en las mismas condiciones.

updateEdges  crea una lista de arcos con los arcos de edges actualizados de acuerdo a
             los arcos de path. Para ello, con peso p y con los extremos de cada arco
             de path, se actualiza de forma acumulativa edges (usando updateEdge) y se
             devuelve la lista de arcos resultante.
@param path     Camino que contiene los arcos usados para actualizar edges.
@param p        Peso con el que actualizar.
@param edges    Los arcos a actualizar.
@return         La lista obtenida a partir de edges tras todas las actualizaciones.
updateEdges :: (Eq a) => Path (WDiEdge a Integer) -> Integer
                      -> [WDiEdge a Integer] -> [WDiEdge a Integer]
```

0.50 puntos

Las siguientes funciones se deberán implementar en el examen:

```
Sea sol una lista de arcos representando la solución. Además, por construcción,
si hay un arco de x a y, entonces no habrá uno de y a x.
addFlow crea una nueva solución con los mismos arcos que sol pero el arco
con extremos x e y se procesa del siguiente modo:
   Si hay un arco x-w->y en edges, se sustituye por x-(w+p)->y.
   Si el arco y-p->x está en edges, se elimina.
   Si hay un arco y-w->x (con w < p) en edges, se reemplaza por x-(p-w)->y.
   Si hay un arco y-w->x (con w > p) en edges, se reemplaza por y-(w-p)->x.
   Si no hay ningún arco con extremos x e y en edges, se introduce el arco x-p->y.
@param x       Un extremo del arco a considerar.
@param y       Otro extremo del arco a considerar.
@param p       El peso.
@param sol     La lista de arcos.
@return        Una nueva lista con los arcos de sol y el de extremos x e y descrito.
addFlow :: (Eq a) => a -> a -> Integer -> [WDiEdge a Integer] -> [WDiEdge a Integer]
```

1 punto

Las siguientes funciones se deberán implementar en el examen:

```
Crea una lista de arcos con los arcos de sol procesados mediante addFlow.
@param path  Camino que contiene los arcos de los flujos que se añadirán a sol
             pero con peso p.
@param p     Valor del peso con el que se añadirá cada flujo.
@param sol   Los arcos a procesar.
@return      Con los extremos de cada arco de path, peso p, y partiendo de la lista sol,
             se aplica addFlow de forma acumulativa y se devuelve la lista de arcos
             resultante.
addFlows :: (Eq a) => Path (WDiEdge a Integer) -> Integer
                      -> [WDiEdge a Integer] -> [WDiEdge a Integer]
```

0.50 puntos

Las siguientes funciones se deberán implementar en el examen:

```
Algoritmo de Ford-Fulkerson:

Se construye una lista vacía sol de arcos con peso
Sea wdg inicialmente el grafo g
Mientras haya un camino (path) de src a dst en wdg:
    Sea mf = el flujo máximo del camino path (mínimo peso de los arcos de path).
    Se crea una lista edges con los arcos del grafo wdg.
    Se usan los arcos de path y el peso -mf para actualizar edges con updateEdges.
    Se usan los arcos del camino inverso a path y el peso mf para actualizar edges con updateEdges.
    Se crea un nuevo grafo wdg con los mismos vértices pero con los arcos de edges.
    Se usan los arcos de path pero con peso mf para añadir flujo a la solución sol con addFlows.
sol será la lista de arcos del grafo de flujo máximo.

@param g    Grafo sobre el que obtener el flujo máximo.
@param src  Origen del flujo (fuente).
@param dst  Destino del flujo (sumidero).
@return     La lista (sol) con los arcos del grafo que proporcionan el flujo máximo.
fordFulkerson :: (Ord a) => WeightedDiGraph a Integer -> a -> a -> [WDiEdge a Integer]
```

2 puntos

Las siguientes operaciones se deberán definir en el examen:

```
Calcula el valor del flujo máximo de un grafo.
@param sol  Lista con los arcos del grafo de flujo máximo.
@param src  Origen del flujo (fuente).
@return     El valor del flujo máximo partiendo de src.
maxFlow :: (Ord a) => [WDiEdge a Integer] -> a -> Integer
```

0.75 puntos

```
Obtiene el flujo máximo del grafo aplicando el teorema MIN-CUT:
Si dividimos el conjunto de vértices del grafo de flujo máximo en dos conjuntos
K y K', exhaustivos y complementarios, de manera que la fuente quede en K y el
sumidero en K', el flujo máximo se calcula sumando el peso de todos los arcos
que parten de vértices de K y llegan a vértices de K' y restando los pesos de
todos los arcos que parten de vértices de K' y llegan a K.
@param g     Grafo al que queremos encontrar el flujo máximo.
@param src   Origen del flujo (fuente).
@param dst   Destino del flujo (sumidero).
@param set   Una lista que representa uno de los conjuntos (K o K'; el otro
             se debe calcular si hace falta).
@return      Flujo máximo
maxFlowMinCut :: (Ord a) => WeightedDiGraph a Integer -> a -> a -> [a] -> Integer
```

1 punto

Java

Trabajaremos con digrafos con pesos, `WeightedDiGraph<V,W>` con el constructor y los métodos usuales de digrafos:

```
WeightedDiGraph<V,W>() // V es el tipo de los vértices y W de los pesos
boolean isEmpty()
void addVertex(V x)
void deleteVertex(V x)
void addDiEdge(V x, W w, V y)
void deleteDiEdge(V x, W w, V y)
int numVertices()
int numEdges()
Set<V> vertices()
List<WDiEdge<V,W>> wDiEdges()
int inDegree(V v)
int outDegree(V v)
Set<Tuple2<V,W>> successors(V x)
Set<Tuple2<V,W>> predecessors(V dst)
Object clone()
```

También con arcos dirigidos con pesos, `WDiEdge<V,W>` con el constructor y métodos siguientes:

```
WDiEdge(V src, W weight, V dst) // Un arco dirigido de src a dst con peso weight
V getSrc()
V getDst()
W getWeight()
```

Y para calcular caminos en anchura desde un vértice a otro usaremos la clase ya implementada `WeightedBreadthFirstTraversal`, a cuyo constructor se le proporciona como parámetros un digrafo con pesos y una fuente:

```
WeightedBreadthFirstTraversal(WeightedDiGraph<V,W> g, V src)
```

Para obtener un camino de dicha fuente hasta un vértice destino `dst`, disponemos del método (ya implementado)

```
List<WDiEdge<V,W>> pathTo(V dst)
```

que devuelve la lista con los arcos de un camino que vaya desde `src` a `dst` en el grafo `g` o `null` si no existe tal camino.

Los siguientes métodos se deberán implementar en el examen:

```
Devuelve el flujo máximo de un camino (menor peso de los arcos que componen el camino).
@param path    Camino donde se busca el flujo máximo.
@return        Flujo máximo.
static <V> int maxFlowPath(List<WDiEdge<V,Integer>> path)
```

0.75 puntos

```
Sea edges una lista de arcos dirigidos con peso donde no hay dos arcos con la misma fuente
y el mismo destino.
updateEdge crea una lista de arcos a partir de los arcos de edges del siguiente modo:
     Si hay un arco x-w->y n edges, se incrementa su peso en p unidades (p puede ser
     negativo). Si tras incrementarlo, el peso resultante es 0, se elimina el arco de
     la lista resultado.
     Si no hay un arco x-w->y en edges, se agrega éste pero con peso p.
     Los demás arcos de edges se mantienen en el resultado.
@param x     Origen del arco a actualizar.
@param y     Destino del arco a actualizar.
@param p     Peso para asignar o con el que incrementar el arco.
@param edges Lista de arcos de entrada.
@return      Una nueva lista con el arco entre x e y descrito y los demás arcos de edges.
static <V> List<WDiEdge<V,Integer>>
                      updateEdge(V x, V y, Integer p, List<WDiEdge<V,Integer>> edges)
```

1 punto

Los siguientes métodos se deberán implementar en el examen:

```
Sea edges una lista de arcos dirigidos con peso donde no hay dos arcos con la misma fuente
y el mismo destino. Sea path otra lista de arcos en las mismas condiciones.

updateEdges  crea una lista de arcos con los arcos de edges actualizados de acuerdo a
             los arcos de path. Para ello, con peso p y con los extremos de cada arco
             de path, se actualiza de forma acumulativa edges (usando updateEdge) y se
             devuelve la lista de arcos resultante.
@param path     Camino que contiene los arcos usados para actualizar edges.
@param p        Peso con el que actualizar.
@param edges    Los arcos a actualizar.
@return         La lista obtenida a partir de edges tras todas las actualizaciones.
static <V> List<WDiEdge<V,Integer>>
       updateEdges(List<WDiEdge<V,Integer>> path, Integer p, List<WDiEdge<V,Integer>> edges)
```

0.50 puntos

Los siguientes métodos se deberán implementar en el examen:

```
Sea sol una lista de arcos representando la solución. Además, por construcción,
si hay un arco de x a y, entonces no habrá uno de y a x.
addFlow crea una nueva solución con los mismos arcos que sol pero el arco
con extremos x e y se procesa del siguiente modo:
   Si hay un arco x-w->y en edges, se sustituye por x-(w+p)->y.
   Si el arco y-p->x está en edges, se elimina.
   Si hay un arco y-w->x (con w < p) en edges, se reemplaza por x-(p-w)->y.
   Si hay un arco y-w->x (con w > p) en edges, se reemplaza por y-(w-p)->x.
   Si no hay ningún arco con extremos x e y en edges, se introduce el arco x-p->y.
@param x       Un extremo del arco a considerar.
@param y       Otro extremo del arco a considerar.
@param p       El peso.
@param sol     La lista de arcos.
@return        Una nueva lista con los arcos de sol y el de extremos x e y descrito.
static <V> List<WDiEdge<V,Integer>>
                     addFlow(V x, V y, Integer p, List<WDiEdge<V,Integer>> sol)
```

1 punto

Los siguientes métodos se deberán implementar en el examen:

```
Crea una lista de arcos con los arcos de sol procesados mediante addFlow.
@param path  Camino que contiene los arcos de los flujos que se añadirán a sol
             pero con peso p.
@param p     Valor del peso con el que se añadirá cada flujo.
@param sol   Los arcos a procesar.
@return      Con los extremos de cada arco de path, peso p, y partiendo de la lista sol,
             se aplica addFlow de forma acumulativa y se devuelve la lista de arcos
             resultante.
static <V> List<WDiEdge<V,Integer>>
              addFlows(List<WDiEdge<V,Integer>> path, Integer p, List<WDiEdge<V,Integer>> sol)
```

0.50 puntos

El siguiente constructor deberá implementarse en el examen:

```
Algoritmo de Ford-Fulkerson

Se construye una lista vacía sol de arcos con peso
Sea wdg inicialmente el grafo g
Mientras haya un camino (path) de src a dst en wdg:
    Sea mf = el flujo máximo del camino path (mínimo peso de los arcos de path).
    Se crea una lista edges con los arcos del grafo wdg.
    Se usan los arcos de path y el peso -mf para actualizar edges con updateEdges.
    Se usan los arcos del camino inverso a path y el peso mf para actualizar edges con updateEdges.
    Se crea un nuevo grafo wdg con los mismos vértices pero con los arcos de edges.
    Se usan los arcos de path pero con peso mf para añadir flujo a la solución sol con addFlows.
sol será la lista de arcos del grafo de flujo máximo.

@param g    Grafo sobre el que obtener el flujo máximo.
@param src  Origen del flujo (fuente).
@param dst  Destino del flujo (sumidero).

Variables de instancia de la clase:
        g       Grafo original.
        src     Fuente.
        dst     Sumidero.
        sol     Tras la ejecución del algoritmo, almacena la lista de arcos del grafo de flujo máximo.
```

2 puntos

```
FordFulkerson(WeightedDiGraph<V,Integer> g, V src, V dst)
```

Las siguientes métodos se deberán definir en el examen:

```
Calcula el valor del flujo máximo de un grafo.
@return    El valor del flujo máximo partiendo de src.
int maxFlow()
```

0.75 puntos

```
Obtiene el flujo máximo del grafo aplicando el teorema MIN-CUT:
Si dividimos el conjunto de vértices del grafo de flujo máximo en dos conjuntos
K y K', exhaustivos y complementarios, de manera que la fuente quede en K y el
sumidero en K', el flujo máximo se calcula sumando el peso de todos los arcos
que parten de vértices de K y llegan a vértices de K' y restando los pesos de
todos los arcos que parten de vértices de K' y llegan a K.
@param set   Uno de los conjuntos (K o K'; el otro se debe calcular si hace falta).
@return      Flujo máximo
int maxFlowMinCut(Set<V> set)
```

1 punto

A partir de aquí,
SOLO para alumnos a tiempo parcial
sin evaluación continua

HASKELL

Las siguientes operaciones se deberán definir en el examen:

```
Comprueba si en todos los vértices intermedios (ni fuente ni sumidero)
del grafo de flujo máximo la cantidad de flujo que entra es igual
a la cantidad de flujo que sale.
@param g     Grafo sobre el que se calcula el flujo máximo.
@param src   Origen del flujo (fuente).
@param dst   Destino del flujo (sumidero).
@return      Cierto si se verifica la propiedad.
localEquilibrium :: (Ord a) => WeightedDiGraph a Integer -> a -> a -> Bool
```

0.75 puntos

```
Dado un grafo, devuelve un par con las listas de fuentes y sumideros.
@param g     Grafo al que queremos encontrar fuentes y sumideros.
@return      El par con las dos listas.
sourcesAndSinks :: (Eq a) => WeightedDiGraph a b -> ([a],[a])
```

0.75 puntos

Las siguientes operaciones se deberán definir en el examen:

```
Si el grafo tiene más de una fuente, agrega una fuente nueva (newSrc), y arcos de
ésta a cada una de las fuentes previas. El peso de cada arco nuevo será la suma de los
pesos de los arcos de la fuente destino del nuevo arco.
Si hay más de un sumidero, agrega un sumidero nuevo (newDst), y arcos de cada una
de los sumideros previos a srcDst. El peso de cada arco nuevo será la suma de los
pesos de los arcos del sumidero origen del nuevo arco.
@param g            Grafo sobre el que se unifican fuentes y sumideros.
@param newSrc       Nueva fuente del grafo.
@param newDst       Nuevo destino del grafo.
@return             Nuevo grafo con una sola fuente y sumidero.
unifySourceAndSink :: (Eq a) => WeightedDiGraph a Integer -> a -> a
                                -> WeightedDiGraph a Integer
```

1 punto

JAVA

Los siguientes métodos se deberán definir en el examen:

```
Comprueba si en todos los vértices intermedios (ni fuente ni sumidero)
del grafo de flujo máximo la cantidad de flujo que entra es igual
a la cantidad de flujo que sale.
@param g     Grafo sobre el que se calcula el flujo máximo.
@param src   Origen del flujo (fuente).
@param dst   Destino del flujo (sumidero).
@return      Cierto si se verifica la propiedad.
static <V> boolean localEquilibrium(WeightedDiGraph<V, Integer> g, V src, V dst)
```

0.75 puntos

```
Dado un grafo, devuelve un par con las listas de fuentes y sumideros.
@param g     Grafo al que queremos encontrar fuentes y sumideros.
@return      El par con las dos listas.
static <V,W> Tuple2<List<V>,List<V>> sourcesAndSinks(WeightedDiGraph<V, W> g)
```

0.75 puntos

Los siguientes métodos se deberán definir en el examen:

```
Si el grafo tiene más de una fuente, agrega una fuente nueva (newSrc), y arcos de
ésta a cada una de las fuentes previas. El peso de cada arco nuevo será la suma de los
pesos de los arcos de la fuente destino del nuevo arco.
Si hay más de un sumidero, agrega un sumidero nuevo (newDst), y arcos de cada una
de los sumideros previos a srcDst. El peso de cada arco nuevo será la suma de los
pesos de los arcos del sumidero origen del nuevo arco.
@param g            Grafo sobre el que se unifican fuentes y sumideros.
@param newSrc       Nueva fuente del grafo.
@param newDst       Nuevo destino del grafo.
static <V> void unifySourceAndSink(WeightedDiGraph<V, Integer> g, V newSrc, V newDst)
```

1 punto