

# Flujo máximo

Ariel Zylber

Training Camp Argentina

14 de julio de 2020

# Redes de flujo

Es importante saber flujo:

- En casi todas las regionales y WF hay un problema de flujo.

Suele ser fácil de implementar:

- La solución es armar el grafo y correr un algoritmo de notebook.

No muchos equipos resuelven el de flujo:

- Muchos no saben/manejan flujo.
- El de flujo no parece que sale con flujo.
- No me sale modelar el problema con flujo.

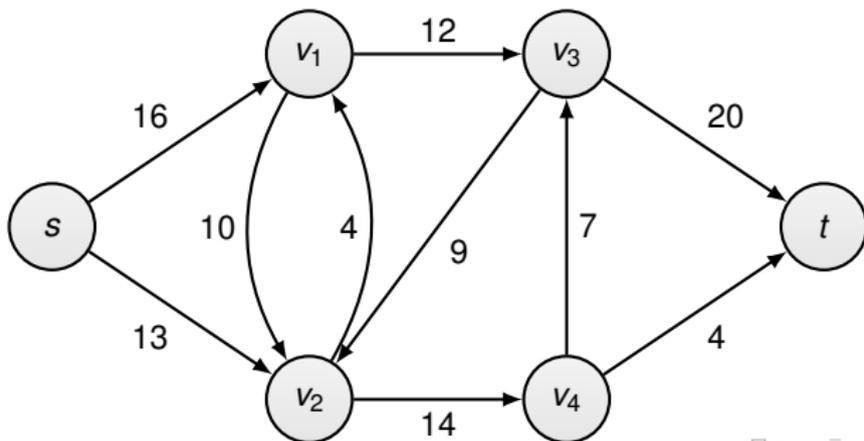
# Redes de flujo

Una red de flujo consiste en:

- Un grafo dirigido  $G$  con pesos en las aristas.
- Dos nodos distinguidos  $s$  (la fuente) y  $t$  (el sumidero).

Nuestro objetivo es mandar la mayor cantidad de unidades de flujo de  $s$  a  $t$ .

- los pesos de las aristas representan la cantidad máxima de flujo que soportan.



# Flujo en redes

Un flujo sobre una red es una asignación de un real no negativo a cada arista tal que:

- Por cada arista digo cuantas unidades de flujo pasan por ella.
- No asigno a una arista más de su capacidad máxima.
- En cada nodo (salvo  $s$  y  $t$ ) el flujo que entra tiene que ser igual al que sale.

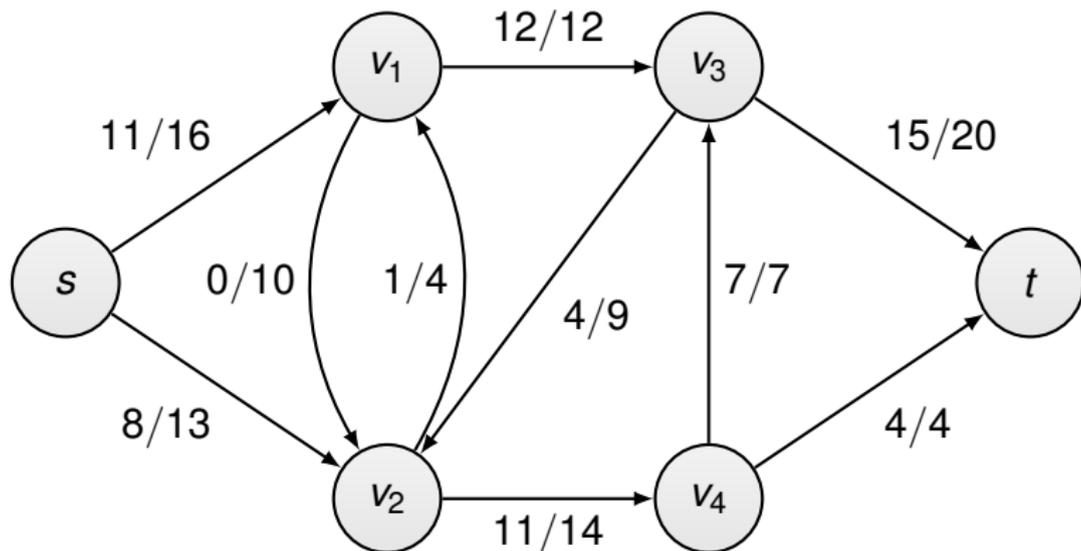
El valor del flujo es:

- La cantidad de flujo que sale de  $s$ .
- La cantidad de flujo que llega a  $t$ .
- Ambos números siempre coinciden.

Queremos encontrar cual es el mayor valor posible de un flujo sobre la red.

# Ejemplo

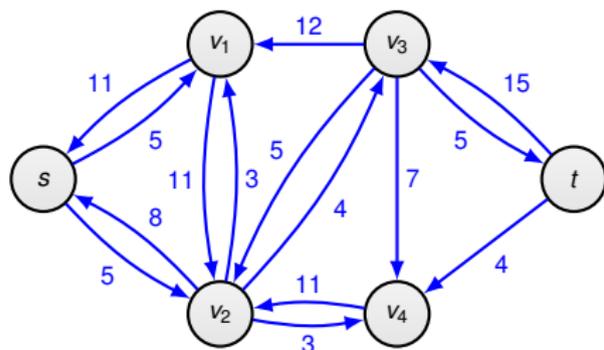
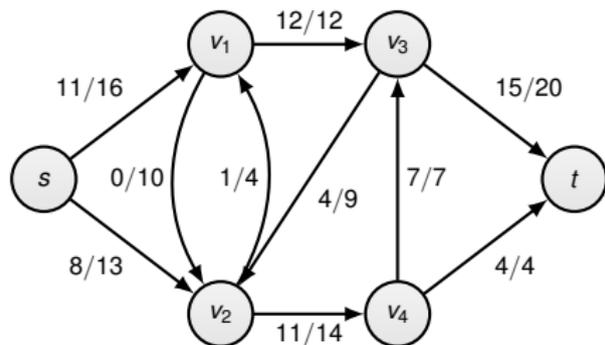
Este es un posible flujo sobre la red:



# Red residual

La red residual de un flujo es un grafo dirigido tal que:

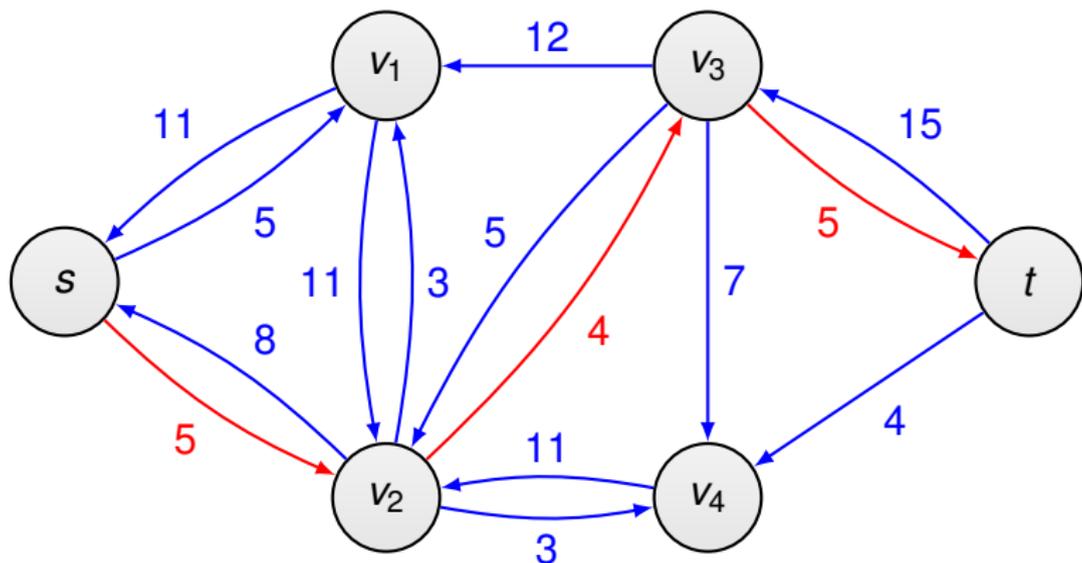
- Los nodos son los mismos que en la red original.
- Hay una arista de  $u$  a  $v$  con capacidad la suma de:
  - Lo que me falta para llenar la capacidad de  $u$  a  $v$  en la red original.
  - El flujo que estoy mandando de  $v$  a  $u$ .



# Camino de aumento

Un camino de aumento es un camino de  $s$  a  $t$  en la red residual. Dado un camino de aumento:

- Podemos conseguir un flujo de mayor valor mandando todo lo posible por este camino.



# Ford-Fulkerson

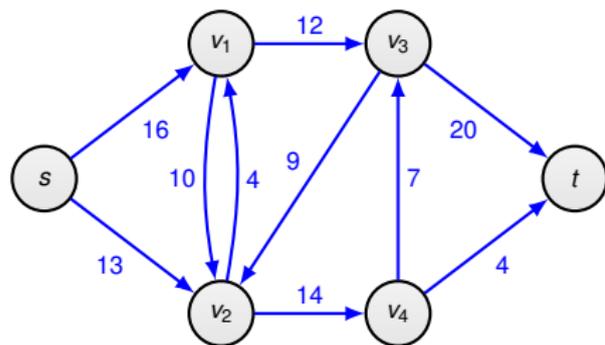
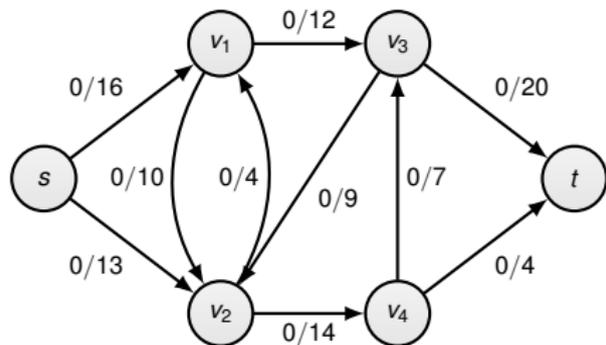
Esto induce el siguiente algoritmo de flujo máximo:

- Empiezo con el flujo vacío (todos 0).
- Construyo la red residual inicial para este flujo.
- Mientras haya camino de  $s$  a  $t$  en la red:
  - Busco el eje de menor peso ( $m$ ) del camino.
  - Actualizo el flujo para mandar  $m$  unidades de flujo por ese camino.
  - Actualizo la red residual para el nuevo flujo.

Devuelvo el flujo obtenido.

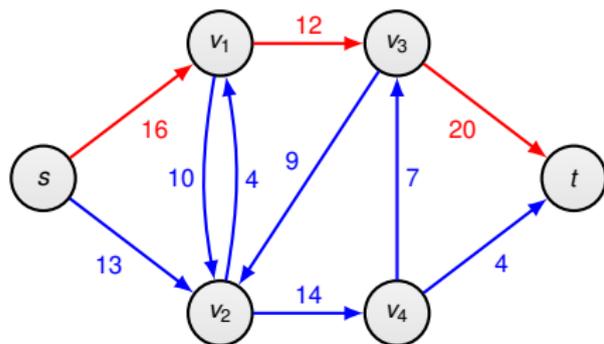
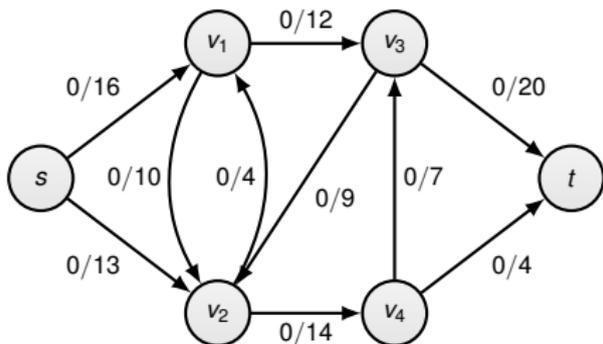
## Ejemplo 1)

- Empezamos con 0 en todos los ejes.



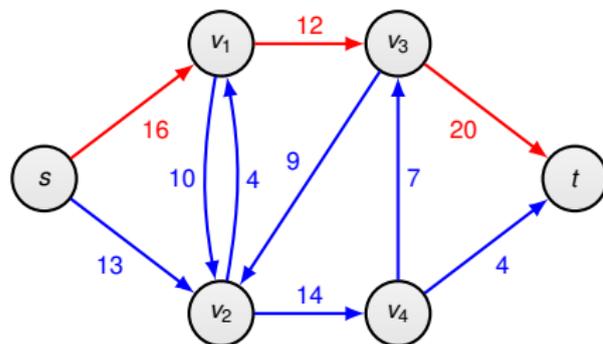
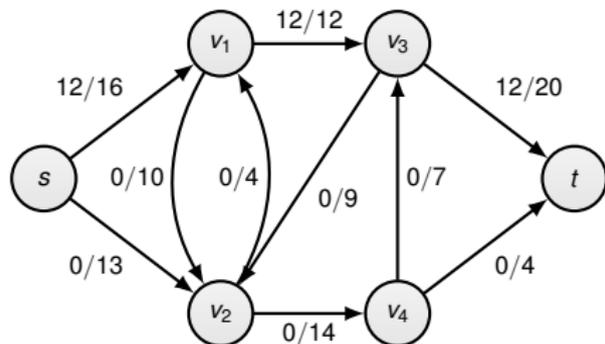
## Ejemplo 2)

- Buscamos un camino en la red residual.



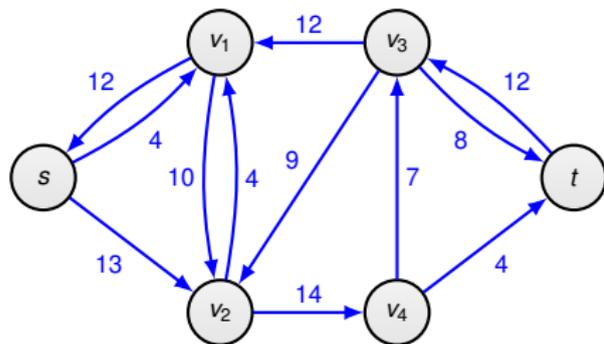
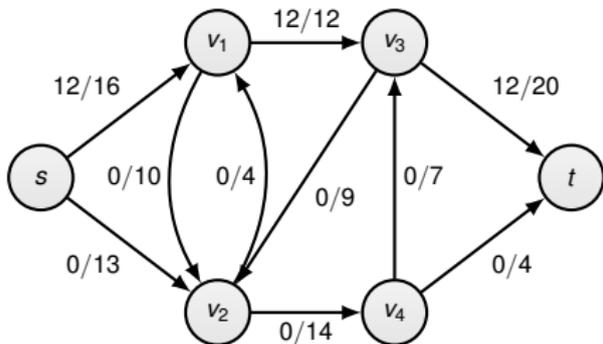
## Ejemplo 3)

- Mandamos todo lo que podemos por ese camino.



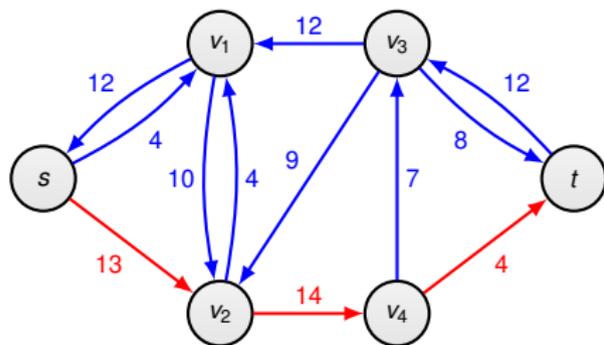
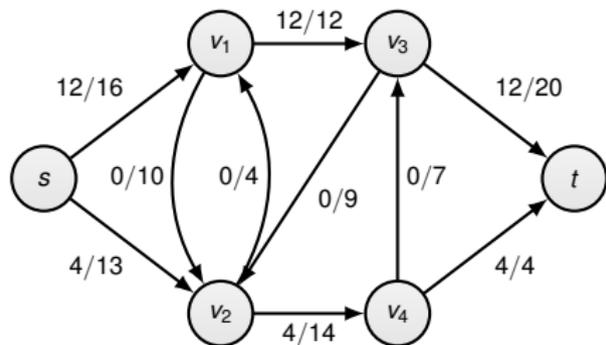
## Ejemplo 4)

- Actualizamos la red residual.



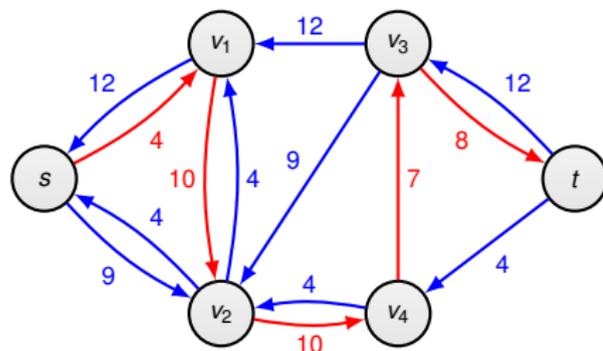
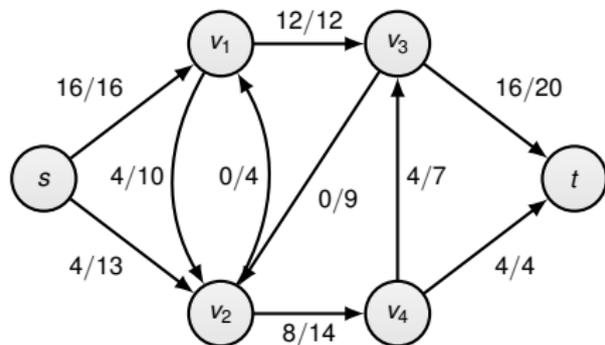
## Ejemplo 5)

- Volvemos a buscar un camino en la red residual y mandamos flujo por el.



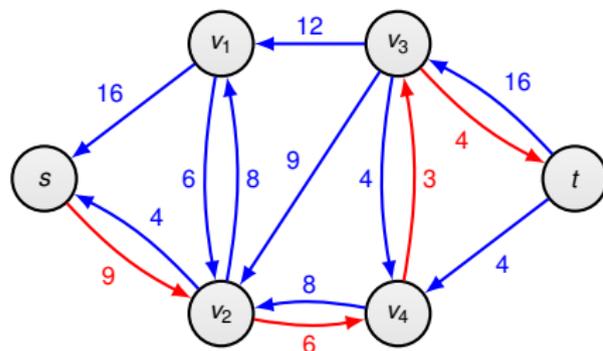
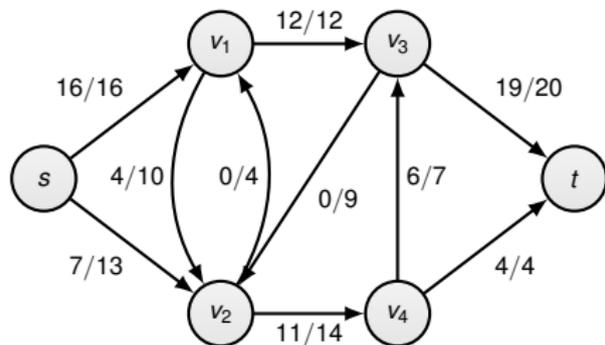
## Ejemplo 6)

- Actualizamos la red residual y repetimos.



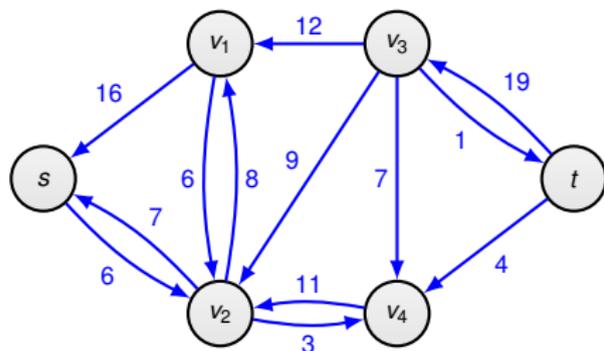
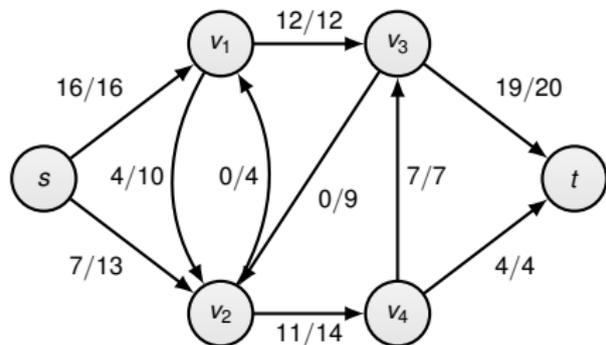
## Ejemplo 7)

- Actualizamos la red residual y repetimos.



## Ejemplo 8)

- No hay mas caminos en la red residual, paramos.



# E-K y Dinitz

Si además buscamos el camino de  $s$  a  $t$  con un BFS:

- El algoritmo queda  $O(nm^2)$  con  $n$  la cantidad de nodos y  $m$  la de aristas.
- Además, siempre tenemos la cota  $O(mF)$  donde  $F$  es el valor del flujo máximo.
- Este algoritmo se conoce como Edmond-Karp.

Otro algoritmo similar es el de Dinitz:

- Es  $O(n^2m)$  por lo que es más rápido que E-K.
- Es más largo de implementar que E-K.

Hay algoritmos mejores pero en competencias con estos dos alcanza. Son fundamentales en cualquier notebook.

# Aprovechar la red residual

Supongamos que nos piden saber si el flujo máximo en una red cambia si aumentamos en 1 la capacidad de una arista ¿Qué harían?

- Podemos calcular el flujo máximo para la red original.
- Luego calculamos el flujo máximo para la red nueva pero partiendo del flujo anterior.

Supongamos ahora que nos disminuyen la capacidad de una arista (por la que saturabamos la capacidad) en 1 ¿Qué harían?

- Buscamos un camino en la red residual de  $t$  a  $s$  que pase por la arista a disminuir.
- "Retrotraemos" 1 de flujo por ese camino y disminuimos la capacidad.
- Calculamos el flujo máximo partiendo de este flujo.

En resumen, pensar siempre al correr varios flujos si no es más eficiente reusar la información obtenida.

# Aprovechar la red residual

Supongamos que nos piden saber si el flujo máximo en una red cambia si aumentamos en 1 la capacidad de una arista ¿Qué harían?

- Podemos calcular el flujo máximo para la red original.
- Luego calculamos el flujo máximo para la red nueva pero partiendo del flujo anterior.

Supongamos ahora que nos disminuyen la capacidad de una arista (por la que saturabamos la capacidad) en 1 ¿Qué harían?

- Buscamos un camino en la red residual de  $t$  a  $s$  que pase por la arista a disminuir.
- "Retrotraemos" 1 de flujo por ese camino y disminuimos la capacidad.
- Calculamos el flujo máximo partiendo de este flujo.

En resumen, pensar siempre al correr varios flujos si no es más eficiente reusar la información obtenida.

# Aprovechar la red residual

Supongamos que nos piden saber si el flujo máximo en una red cambia si aumentamos en 1 la capacidad de una arista ¿Qué harían?

- Podemos calcular el flujo máximo para la red original.
- Luego calculamos el flujo máximo para la red nueva pero partiendo del flujo anterior.

Supongamos ahora que nos disminuyen la capacidad de una arista (por la que saturabamos la capacidad) en 1 ¿Qué harían?

- Buscamos un camino en la red residual de  $t$  a  $s$  que pase por la arista a disminuir.
- "Retrotraemos" 1 de flujo por ese camino y disminuimos la capacidad.
- Calculamos el flujo máximo partiendo de este flujo.

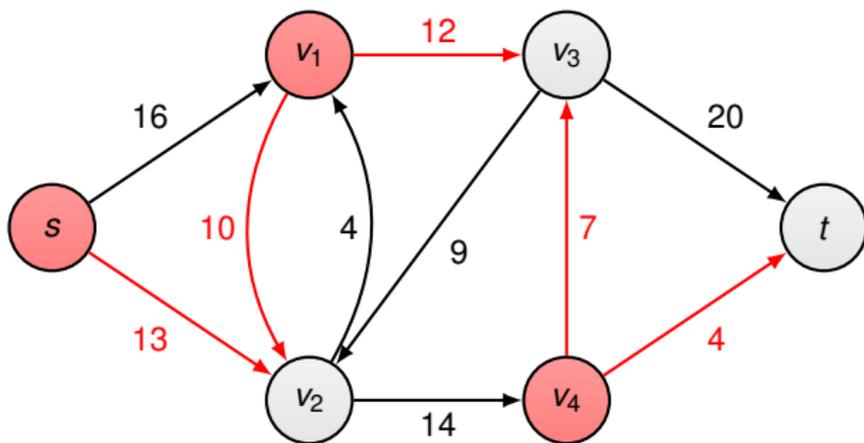
En resumen, pensar siempre al correr varios flujos si no es más eficiente reusar la información obtenida.

# Definición de corte

Un corte en una red es una partición de los nodos en dos conjuntos  $U$  y  $V$  tal que  $s \in U$  y  $t \in V$ .

Dado un corte en una red su valor es:

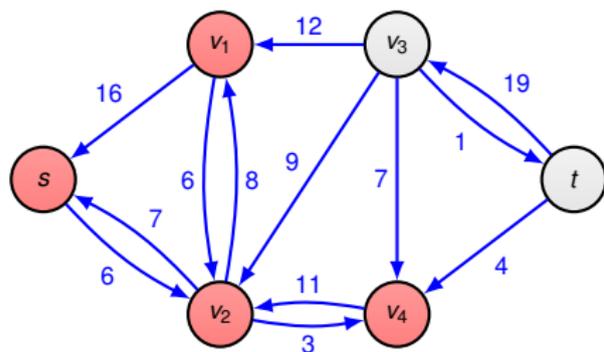
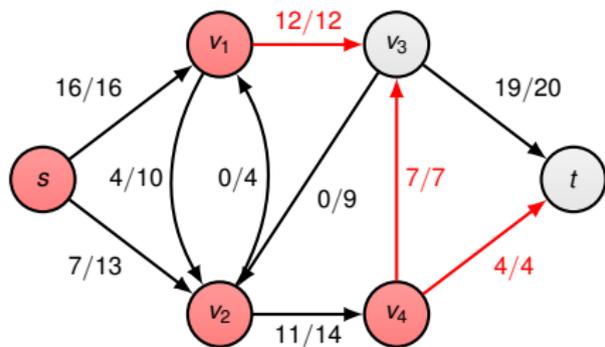
- La suma de las capacidades de todos los ejes que van de un nodo de  $U$  a uno de  $V$ .



# Teorema

El teorema de Max-Flow Min-Cut nos dice que:

- El valor del corte mínimo coincide con el del flujo máximo.
- Podemos encontrar un corte mínimo definiendo:
  - $U$  como los nodos alcanzables desde  $s$  en la red residual del flujo máximo.
  - $V$  como el resto de los nodos.



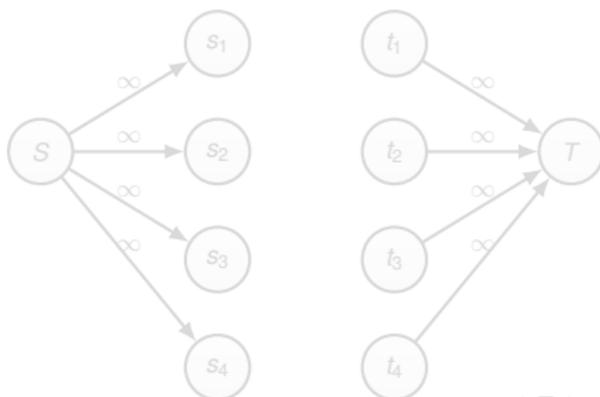
# Ejemplo: Pintar nodos

Dado un grafo  $G$ , queremos pintar cada nodo de rojo o azul de manera que la cantidad de aristas que unen nodos de distinto color sea mínima. Algunos nodos ya vienen pintados y no los podemos cambiar.

# Múltiples fuentes/sumideros

Supongamos que tenemos un problema donde hay varios nodos de donde "sale" flujo y/o varios nodos a los que tiene que llegar flujo  
¿Qué hacemos?

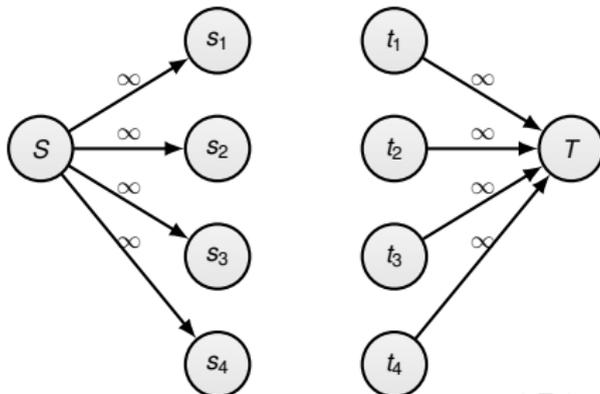
- Creamos dos nodos  $s$  y  $t$  nuevos.
- Conectamos  $s$  a todos los nodos de los que "sale" flujo con capacidad  $\infty$
- Conectamos todos los nodos a los que llega flujo con  $t$  con capacidad  $\infty$ .



# Múltiples fuentes/sumideros

Supongamos que tenemos un problema donde hay varios nodos de donde "sale" flujo y/o varios nodos a los que tiene que llegar flujo  
¿Qué hacemos?

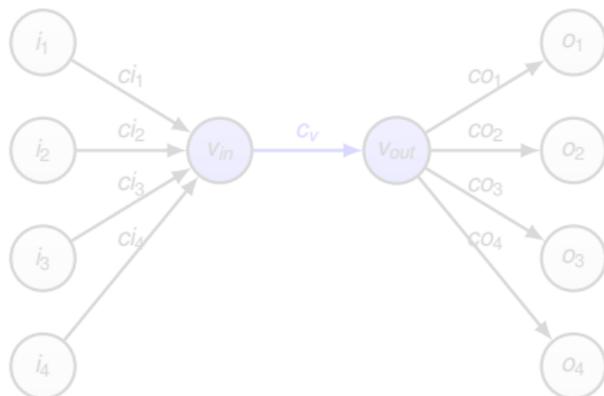
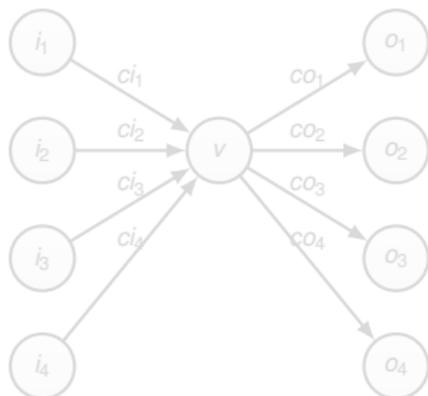
- Creamos dos nodos  $s$  y  $t$  nuevos.
- Conectamos  $s$  a todos los nodos de los que "sale" flujo con capacidad  $\infty$
- Conectamos todos los nodos a los que llega flujo con  $t$  con capacidad  $\infty$ .



# Restricciones sobre los nodos

¿Qué pasa si tenemos restricciones de flujo que pasa por un nodo  $v$  en lugar de por un eje?

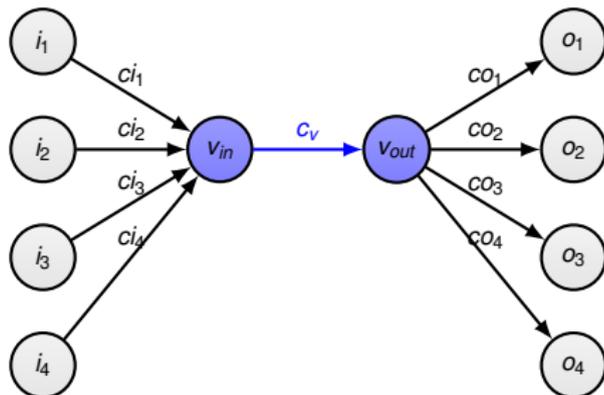
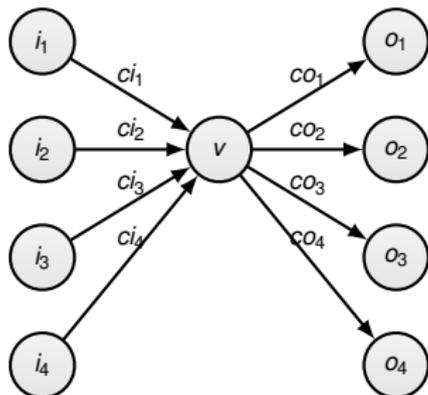
- Duplicamos el nodo en dos nodos  $v_{in}$  y  $v_{out}$ .
- Conectamos todas las aristas que entraban a  $v$  a  $v_{in}$ .
- Conectamos todas las aristas que salían de  $v$  a  $v_{out}$ .
- Conectamos  $v_{in}$  a  $v_{out}$  con un eje de la capacidad máxima del nodo.



# Restricciones sobre los nodos

¿Qué pasa si tenemos restricciones de flujo que pasa por un nodo  $v$  en lugar de por un eje?

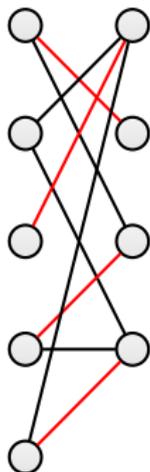
- Duplicamos el nodo en dos nodos  $v_{in}$  y  $v_{out}$ .
- Conectamos todas las aristas que entraban a  $v$  a  $v_{in}$ .
- Conectamos todas las aristas que salían de  $v$  a  $v_{out}$ .
- Conectamos  $v_{in}$  a  $v_{out}$  con un eje de la capacidad máxima del nodo.



# Matching máximo

Dado un grafo, un matching es un conjunto de ejes tal que no haya dos que incidan sobre el mismo vértice.

El problema de matching máximo pide el mayor conjunto de ejes que cumpla esto.

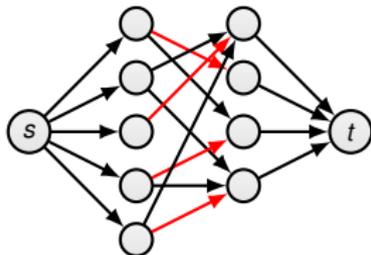


# Matching máximo bipartito

Si el grafo es bipartito se puede modelar con un flujo de la siguiente manera:

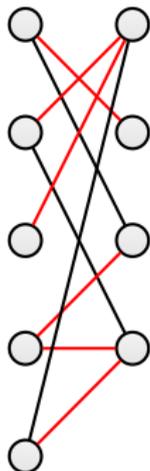
- Si  $P$  y  $Q$  son los conjuntos de nodos de cada lado.
- Creo un nodo  $s$  y lo conecto con capacidad 1 a cada nodo de  $P$ .
- Creo un nodo  $t$  y conecto cada nodo de  $Q$  a  $t$  con capacidad 1.
- Por cada arista  $(p, q)$  del grafo uno a  $p$  con  $q$  con capacidad  $\infty$ .

El flujo máximo sobre esta red nos da el matching máximo. Las aristas del medio por las que enviamos flujo forman el matching.



# Mínimo cubrimiento por aristas

Dado un grafo, un cubrimiento por aristas es un conjunto de ejes tal que todo nodo sea extremo de alguna arista del conjunto. Ahora nos interesa el cubrimiento más chico.

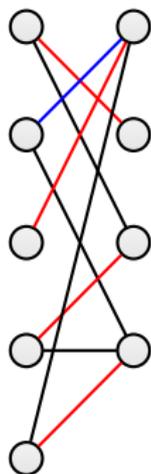


# Mínimo cubrimiento por aristas bipartito

Si el grafo es bipartito se puede obtener a partir del matching máximo:

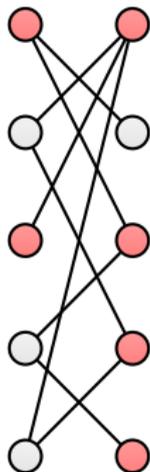
- Agrego al cubrimiento todas las aristas de un matching máximo.
- Por cada nodo que me quedó sin cubrir agrego una arista cualquiera que incida en el nodo.

De esta manera, el tamaño del cubrimiento mínimo es  $n - MM$  donde  $MM$  es el tamaño del matching máximo.



# Mínimo cubrimiento por nodos

Dado un grafo, un cubrimiento por nodos es un conjunto de nodos tal que todo eje tenga algún extremo en el conjunto. Nuevamente nos interesa el cubrimiento más chico.



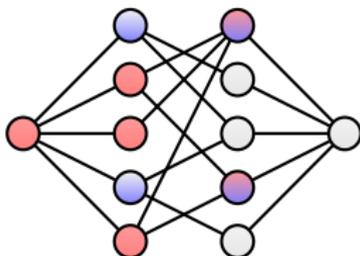
# Mínimo cubrimiento por nodos bipartito

Si el grafo es bipartito:

- El teorema de Konig nos dice que el tamaño del cubrimiento mínimo coincide con el del matching máximo.

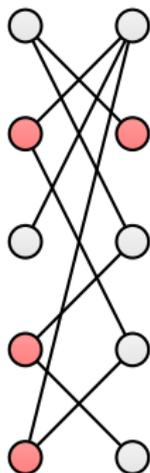
Para obtener el cubrimiento:

- Hallar el flujo máximo sobre la red del problema de matching máximo.
- Busco un corte mínimo.
- Un cubrimiento mínimo son los extremos de las aristas cortadas que no son  $s$  ni  $t$ .



# Máximo conjunto independiente

Dado un grafo, un conjunto independiente es un conjunto de nodos tal que no hay dos adyacentes. Queremos encontrar el conjunto independiente más grande.





# Sobre la complejidad

A la hora de calcular un matching máximo con flujo.

Si uso E-K:

- El flujo máximo es a lo sumo  $O(n)$ .
- Usando la cota  $O(mF)$  nos queda que matching máximo es  $O(mn)$ .

Si uso Dinitz:

- Se puede probar que en la red de matching máximo tarda  $O(m\sqrt{n})$ .

# Problemas sobre tableros

Es muy común tener alguno de estos problemas escondido en algún problema de tableros.

Conviene pensar el siguiente modelo:

- Creo un grafo bipartito.
- Creo un nodo en  $P$  por cada fila del tablero.
- Creo un nodo en  $Q$  por cada columna del tablero.
- Una arista entre un nodo fila y un nodo columna representa la casilla de esa fila y columna.

Muchas veces en este modelo el problema se reduce a alguno de los 4 que vimos.

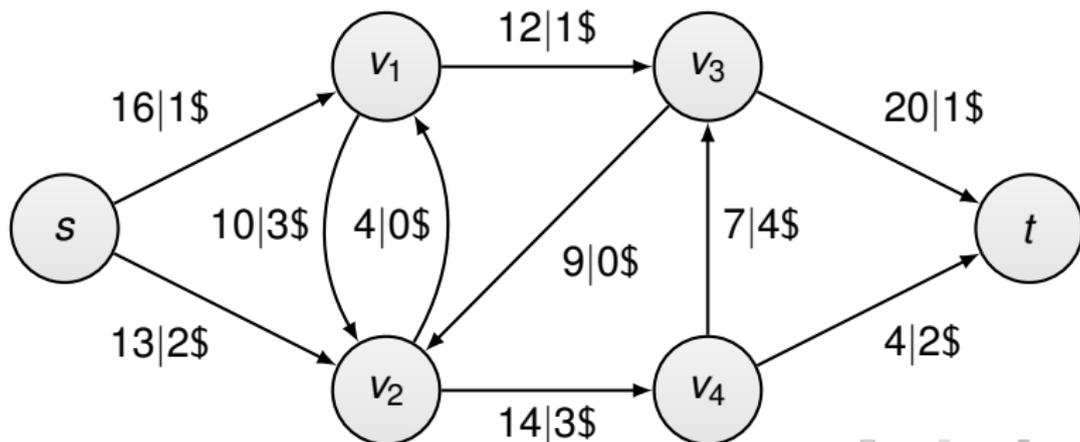
# Ejemplo

Costume Change

# Flujo máximo de costo mínimo

Una variante del problema de flujo máximo es la siguiente:

- A cada eje, le agregamos un costo de mandar una unidad de flujo por ese eje.
- El costo de un flujo es la suma de los costos de las aristas multiplicado por el flujo que mandamos por cada una.
- Queremos encontrar entre todos los flujos máximos, aquel que minimice el costo total.



# Algoritmos

Se puede resolver con ideas similares:

- Si corro el algoritmo usual pero busco un camino de aumento con Bellman-Ford en los costos obtengo uno de costo mínimo. Complejidad  $O(nmF)$ .
- Se puede hacer una modificación para que los costos no sean negativos y correr Dijkstra. Complejidad  $O(n^2F)$  o  $O(m\log(n)F)$ .

# Matching máximo de costo mínimo

La versión de matching máximo de costo mínimo es similar y se puede resolver así:

- Usando el Hungarian Algorithm que lo resuelve en  $O(n^3)$ .
- Modelándolo igual que antes pero con costos en las aristas y usar el algoritmo de Max Flow Min Cost.

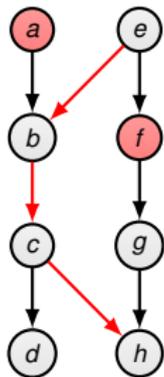
# Ejemplo

Direction Board

# Dilworth

Dado un grafo acíclico dirigido (DAG) tenemos las siguientes definiciones:

- Una cadena es una secuencia de nodos tal que de cada uno puedo llegar al siguiente.
- Una anticadena es un conjunto de nodos tal que no hay camino entre ningún par de ellos.



# Teorema

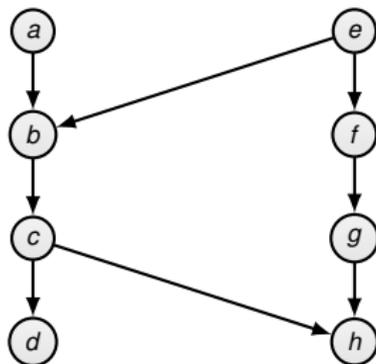
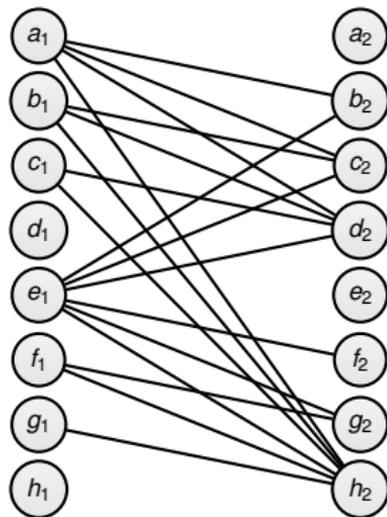
El teorema de Dilworth nos dice que los tamaños de estas cosas coinciden:

- La anticadena de mayor tamaño
- La menor cantidad de cadenas que cubren todos los nodos.

# Grafo auxiliar

Construimos el siguiente grafo bipartito:

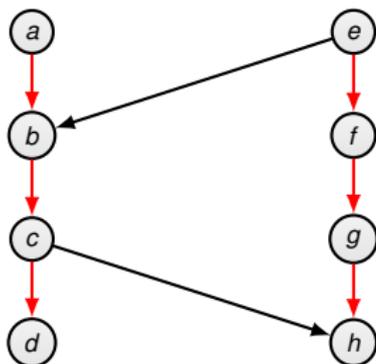
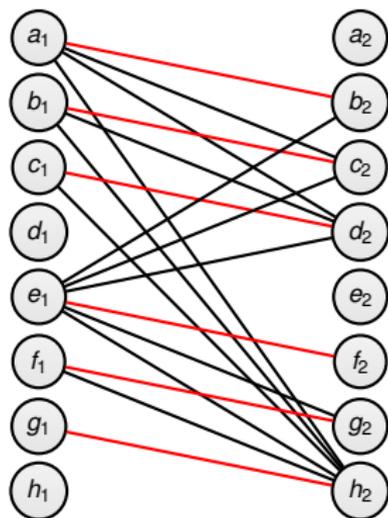
- Ponemos una copia de cada nodo del grafo de cada lado del bipartito.
- unimos un nodo del lado izquierdo  $v$  con uno del derecho  $w$  si hay camino de  $v$  a  $w$  en el grafo original.



# Cubrimiento por cadenas mínimo

Para el cubrimiento por cadenas:

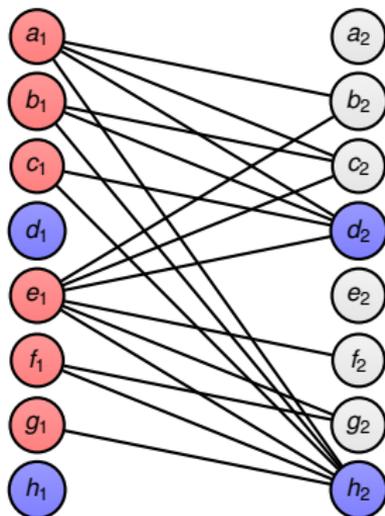
- Obtenemos un matching máximo del grafo.
- Si un eje de  $v$  a  $w$  esta en el matching, agregamos la cadena formada por el camino de  $v$  a  $w$  en el grafo.
- Si las dos copias de un nodo están en el matching, unimos las dos cadenas con extremos en este nodo.



# Anticadena máxima

Para la máxima anticadena:

- Obtenemos un cubrimiento por nodos del grafo.
- Tomamos los nodos que no tienen ninguna copia en el cubrimiento,



# Ejemplo

## Stock Charts

## Más problemas:

- **Inspection:** [goo.gl/iYho0F](https://goo.gl/iYho0F)
- **Super Watch:** [goo.gl/Y1h5zY](https://goo.gl/Y1h5zY)
- **Oil Company:** [goo.gl/XW3jSp](https://goo.gl/XW3jSp)
- **Shogui Tournament:** [goo.gl/KRz3Yi](https://goo.gl/KRz3Yi)
- **Graduation:** [goo.gl/hnCoOU](https://goo.gl/hnCoOU)
- **Parking:** [goo.gl/21ZJqC](https://goo.gl/21ZJqC)
- **"Shortest" pair of paths:** [goo.gl/ardYaY](https://goo.gl/ardYaY)
- **Angels and Devils:** [goo.gl/sCkd30](https://goo.gl/sCkd30)

## Los algoritmos en detalle (inglés):

- **Max flow:** [goo.gl/luDODH](https://goo.gl/luDODH)
- **Min-cost max-Flow:** [goo.gl/XfRWjS](https://goo.gl/XfRWjS)
- **Hungarian:** <https://tinyurl.com/ycto9f3n>