

Análisis y Diseño de Algoritmos

Modelar con flujo

Matching, caminos disjuntos, eliminación y selección de proyectos

Gerson Garrido

Semana 12

UNIVERSIDAD NACIONAL DE INGENIERÍA

Hoy

Sesión A

1. Qué es una reducción
2. Matching bipartito
3. Caminos disjuntos en aristas
4. Caminos disjuntos en nodos
5. El teorema de Menger

Sesión B

6. Asignación con cupos
7. Eliminación en un campeonato
8. Selección de proyectos
9. Reconocer un problema de flujo
10. Lo que no es una capacidad

> **Lectura**
>
> Apuntes de la semana 12. Código en `code/week-12`.

SECCIÓN 01

Qué es una reducción

01

La misma herramienta, otros problemas

Ya tenemos: flujo máximo, corte mínimo, y con capacidades enteras, flujo entero.

Hoy no aparece ningún algoritmo de flujo nuevo. Aparecen problemas que no mencionan redes:

- asignar estudiantes a secciones,
- saber si un equipo todavía puede ganar un campeonato,
- elegir qué proyectos conviene hacer.

La forma de trabajar

Construir una red a propósito, de modo que sus flujos o sus cortes *sean* las soluciones. El programa ya está escrito: lo difícil es diseñar la red y demostrar que representa lo que decimos.

Reducción

Reducción ***(reduction)***

Resolver un problema A usando, como caja negra, un algoritmo que ya existe para un problema B.

1. **Transformación:** de cada instancia *I* de A se construye una instancia *J* de B.
2. **Lectura:** de una solución de *J* se obtiene una solución de *I*.
3. **Demostración:** lo leído es correcto y óptimo.

Costo: construir *J*, más resolver *J*, más leer. En la semana 13 las reducciones vuelven con otro propósito.

Siempre las dos direcciones

Al maximizar:

- **De A hacia B.** Cada solución de I, con valor v, da una solución de J con valor v.
  La red no perdió ninguna solución: óptimo de $J \geq$ óptimo de I.
- **De B hacia A.** Cada solución de J que devuelve el algoritmo, con valor v, da una solución de I con valor v.
  La red no inventó ninguna solución: óptimo de $I \geq$ óptimo de J.

Error común

Probar solo una dirección. Con una sola, la red puede tener soluciones de más, o de menos.

SECCIÓN 02

Matching bipartito

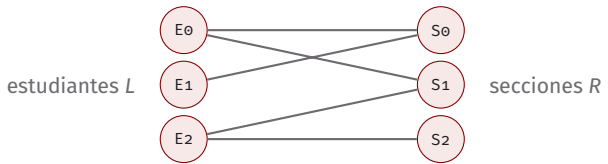
02

El problema

Cada sección admite, por ahora, a un solo estudiante. Cada estudiante puede ir a algunas.

Matching bipartito ***(bipartite matching)***

En un grafo con dos lados L y R, un conjunto de aristas sin nodos en común. Se busca uno de tamaño máximo.



Dos reglas que parecen buenas

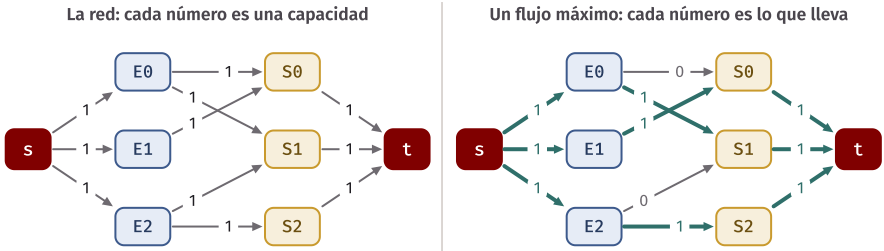
build/matching_demo, extracto

```
Cada uno a su primera opción libre falla:
  E0: S0 S1
  E1: S0
  primera opción libre:        1 (E0-S0)
  óptimo por fuerza bruta:     2

Menos opciones primero también falla:
  E0: S0 S1
  E1: S2 S0
  E2: S2
  menos opciones primero:      2 (E0-S0, E2-S2)
  óptimo por fuerza bruta:     3
```

Una decisión ya tomada estorba, y estas reglas no vuelven atrás. Ordenar mejor no cura eso. Lo cura poder decirle a E0: “muévete a S1”.

La red



Flujo máximo: 3. Matching: las aristas del medio con flujo 1, que son E0-S1, E1-S0 y E2-S2.

- Todas las capacidades son 1.
- s $\rightarrow$ cada estudiante. Cada sección $\rightarrow$ t. Cada opción, de izquierda a derecha.
- **Lectura:** el matching son las aristas del medio con flujo 1.

Teorema: dirección 1

Afirmación

El tamaño de un matching máximo es igual al flujo máximo de la red.

De un matching a un flujo. Sea M un matching de tamaño k.

1. Por cada arista $\{x, y\}$ de M se envía 1 por $s \to x \to y \to t$.
2. **Capacidades:** x está en una sola arista de M, igual que y. Entonces $s \to x$ e $y \to t$ se usan a lo sumo una vez.
3. **Conservación:** cada nodo recibe lo que entrega.
4. **Valor:** salen k unidades de s.

Por lo tanto, flujo máximo $\geq$ matching máximo.

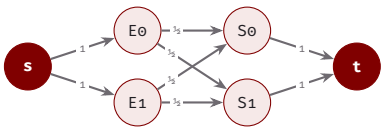
Teorema: dirección 2

De un flujo entero a un matching. Sea f un flujo entero de valor k.

1. Capacidades 1 y f entero: cada arista lleva 0 o 1. Sea M el conjunto de aristas del medio con flujo 1.
2. A cada x le entra a lo sumo 1, solo por $s \to x$. Por conservación, sale a lo sumo 1: aparece a lo sumo una vez en M.
3. De cada y sale a lo sumo 1, solo por $y \to t$. Entonces le entra a lo sumo 1.
4. **Tamaño:** el corte $\{s\} \cup L$ tiene como aristas justo las del medio, y su flujo neto es k. Entonces $|M| = k$.

Con la integralidad de la semana 11 hay un flujo máximo entero. Por lo tanto, matching máximo $\geq$ flujo máximo. ■

Por qué importa que sea entero

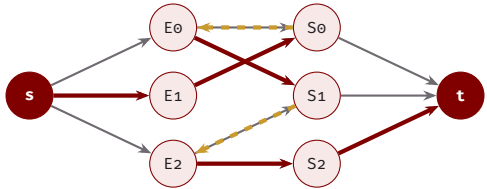


- Es un flujo válido, y máximo: vale 2.
- “Las aristas con flujo 1” son ninguna. Media silla no significa nada.
- La dirección 2 solo necesita que **exista** un flujo máximo entero.
- Ford-Fulkerson encuentra uno: cada aumento empuja un entero.

Error común

Razonar con cualquier flujo máximo como si fuera entero. No todo flujo máximo lo es: el que devuelve el programa, sí.

Paso a paso



	Camino	Matching
1	`s E0 S0 t`	E0-S0
2	`s E2 S1 t`	E0-S0, E2-S1
3	`s E1 S0 E0 S1 E2 S2 t`	E1-S0, E0-S1, E2-S2

Las líneas punteadas son aristas inversas: “E0, suelta S0” y “E2, suelta S1”. Nadie que tenía sección la pierde, aunque le puede cambiar.

En C++

matching.h

```cpp
inline MatchingNetwork matching_network(const Options& options, int right,
                                        long long source_capacity = 1) {
    int left = static_cast<int>(options.size());
    MatchingNetwork net{MaxFlow(left + right + 2), 0, left + right + 1, {}};
    for (int i = 0; i < left; i++) net.flow.add_edge(net.source, 1 + i,
     source_capacity);
    for (int j = 0; j < right; j++) net.flow.add_edge(1 + left + j, net.sink, 1);
    for (int i = 0; i < left; i++) {
        net.middle.emplace_back();
        for (int j : options[at(i)])
            net.middle.back().push_back(net.flow.add_edge(1 + i, 1 + left + j, 1));
    }
    return net;
}
```

Probar solo una dirección

“Total, las aristas del medio ya tienen capacidad 1.” Quitemos el límite en $s \to x$:

build/matching_demo, extracto

```
Sin el límite en s -> estudiante, E0 cuenta dos veces:
  E0: S0 S1
  E1:
  flujo con capacidad 1000 en s -> E: 2
  matching máximo real:               1
```

- **Dirección 1:** sigue valiendo. Todo matching da un flujo.
- **Dirección 2:** se rompe. E0 recibe 2 y entrega una unidad a cada sección.
- Lo simétrico también pasa: una red demasiado estrecha pierde soluciones, y la respuesta sale baja.

Dos preguntas para toda red: ¿puede representar cada solución? ¿Todo flujo entero representa una solución?

Cuánto cuesta

- Flujo $\leq \min(|L|, |R|) \leq V/2$. Cada búsqueda cuesta $O(V + E)$.
- Total: $O(V(V + E))$, es decir, $O(VE)$ sin nodos aislados. Sirve cualquier forma de buscar.

build/matching_bench

n	aristas	matching	aumentos EK	ms EK	aumentos DFS	ms DFS
2000	5995	1873	1873	34	1873	14
4000	11995	3755	3755	136	3755	76
8000	23998	7515	7515	662	7515	281
16000	47996	15024	15024	2683	15024	1259
32000	95997	30065	30065	13552	30065	4291

- Los aumentos son el tamaño del matching: eso no depende de la máquina.
- Los tiempos sí: una sola medición, con `-O2`, en el contenedor. Al duplicar n, BFS se multiplicó por algo entre 4 y 5.

SECCIÓN 03

Caminos disjuntos en aristas

03

El problema

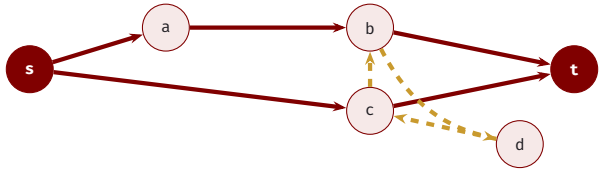
Enlaces dirigidos entre computadoras. ¿Cuántas rutas de s a t se pueden usar a la vez, sin que dos compartan un enlace?

> **Disjuntos en aristas** ***(edge-disjoint)***
> Dos caminos que no comparten ninguna arista. Pueden compartir nodos.

La red: **el mismo grafo, con capacidad 1 en cada arista**.

- k caminos disjuntos $\Rightarrow$ flujo k: inmediato.
- Flujo entero de valor k $\Rightarrow$ ¿k caminos? Un flujo es un montón de aristas con 1.

Un flujo puede tener ciclos



- Flujo de valor 2, con 1 en cada arista dibujada. Se cumple la conservación en todos los nodos.
- El ciclo $b \to d \to c \to b$ (punteado) no aporta nada al valor, y la definición de flujo lo permite.
- La demostración tiene que valer para cualquier flujo entero: no puede suponer que no hay ciclos.

Lema de descomposición

Afirmación

Las aristas con flujo 1 de un flujo entero de valor k se reparten en k caminos simples de s a t, más ciclos. Cada arista va en uno solo.

Inducción sobre las aristas con flujo. Si $k \geq 1$, caminamos desde s por esas aristas, hasta llegar a t o repetir un nodo.

- **No nos atascamos.** De s sale algo, porque $k \geq 1$. En $v \neq s, t$, entró una arista, y por conservación sale una que no se ha usado.
- **Llegamos a t:** camino simple. Al quitarlo, queda un flujo de valor $k - 1$.
- **Repetimos v:** ciclo. Al quitarlo, queda un flujo de valor k.
- **Si $k = 0$:** todo está balanceado. Desde cualquier arista se camina hasta repetir: otro ciclo.

Un ciclo puede pasar por s y por t: por eso se sacan caminos exactamente k veces.

El algoritmo y un ejemplo

pseudocódigo

```
algoritmo descomponer(flujo, s, t, k)
    salidas[v] ← aristas de flujo que salen de v
    para i desde 1 hasta k hacer
        recorrido ← [s]
        mientras el último v no sea t hacer
            w ← quitar una arista de salidas[v]
            si w ya está en recorrido entonces
                apartar el ciclo desde w
                dejar recorrido hasta w
            sino
                agregar w a recorrido
            fin si
        fin mientras
        guardar recorrido como camino
    fin para
fin algoritmo
```

build/disjoint_demo

```
Un flujo de valor 2 con
un ciclo, escrito a mano:
  caminos: 2
    s c t
    s a b t
  ciclo apartado: c b d c
```

Lo que queda en `salidas` son solo ciclos. Cuesta $O(V + F)$, con F las aristas con flujo.

El teorema, y el caso no dirigido

Teorema

Máximo de caminos de s a t disjuntos en aristas = flujo máximo con capacidades 1.

- Caminos $\Rightarrow$ flujo: se envía 1 por cada uno.
- Flujo $\Rightarrow$ caminos: integralidad, y el lema. Los ciclos se descartan.

No dirigido: cada arista $\{u, v\}$ son dos dirigidas, $u \to v$ y $v \to u$, con capacidad 1.

- Caminos $\Rightarrow$ flujo: cada camino, orientado de s a t, usa cada arista en una dirección.
- Flujo $\Rightarrow$ caminos: si un flujo usa las dos direcciones, **se cancelan**. Se mantienen la conservación y el valor. Luego, el lema.

Una arista en las dos direcciones

build/disjoint_demo, extracto

```
No dirigido: el flujo de esta red usa una arista en las dos direcciones.
  aristas canceladas: 1
  caminos: 3
    s t
    s b e t
    s d a c t
```

Esto no es un caso imaginario: el flujo que calcula el programa, en esta red, lo hizo.

Error común

Olvidar la cancelación. Sin ella, dos caminos pueden compartir una arista no dirigida.

SECCIÓN 04

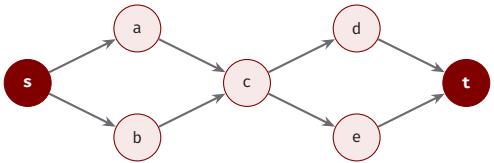
Caminos disjuntos en nodos

04

Pedir más: no compartir nodos

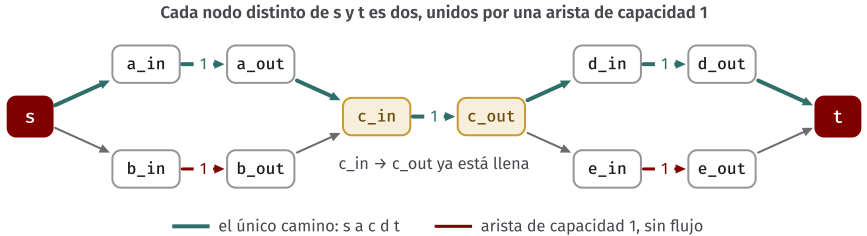
Disjuntos en nodos ***(vertex-disjoint)***

Dos caminos que no comparten ningún nodo, salvo s y t.



- Disjuntos en aristas: 2. Disjuntos en nodos: 1, porque todos pasan por c.
- Las capacidades de las aristas no pueden expresarlo: con 1 en cada arista, c deja pasar 2.

Dividir un nodo



- Cada $v \neq s, t$ se divide en $v_{in} \to v_{out}$, con capacidad 1.
- $u \to v$ pasa a ser $u_{out} \to v_{in}$. Lo que entra llega a v_{in}, lo que sale parte de v_{out}.
- **Así se limita a 1 el número de caminos que pasan por un nodo.** Con otra capacidad, a otro número.
- s **y** t **no se dividen:** son compartidos por todos, y dividirlos permitiría un solo camino.

Teorema

Afirmación

Si no hay arista $s \to t$: máximo de caminos disjuntos en nodos = flujo máximo de la red dividida.

1. **Caminos** $\Rightarrow$ **flujo.** Se envía 1 por cada uno. Como cada v está en un solo camino, cada arista lleva a lo sumo 1.
2. **Flujo** $\Rightarrow$ **caminos.** Flujo entero, y lema de descomposición.
3. Un camino que toca v pasa por $v_{in} \to v_{out}$: es la única salida de v_{in} y la única entrada de v_{out}.
4. Esa arista tiene capacidad 1: a lo sumo un camino toca v. ■

Si s y t son vecinos

build/disjoint_demo, extracto

```
Con una arista directa s -> t:
  caminos: 3
    s b t
    s a t
    s t
  flujo si las aristas tienen capacidad "sin límite" (aquí, 5): 7
```

- Con capacidad 1, $s \to t$ cuenta como un camino, sin nodos internos: 1 más que sin ella.
- Con “sin límite”, esa arista deja pasar 5, más 2 por a y b: 7.
- No es un error de la red. Si son vecinos, ningún conjunto de nodos los separa.

SECCIÓN 05

El teorema de Menger

05

El teorema de Menger

Máximo de cosas disjuntas = mínimo de lo que hay que romper. Sale como corolario.

Forma con aristas

En un grafo dirigido, con $s \neq t$: el máximo de caminos de s a t disjuntos en aristas es igual al mínimo de aristas que hay que quitar para que no quede ningún camino de s a t.

Forma con nodos

En un grafo dirigido, con $s \neq t$ y **sin arista** $s \to t$: el máximo de caminos de s a t disjuntos en nodos es igual al mínimo de nodos, distintos de s y t, que hay que quitar para que no quede ninguno.

Un conjunto de nodos así es un **separador** *(separator)*.

Demostración: forma con aristas

Máximo de caminos = flujo máximo = corte mínimo. Falta: **corte mínimo = mínimo que desconecta**.

1. **Un corte da un conjunto que desconecta.** Sea F el conjunto de aristas de S a T. Todo camino de s a t sale de S por una arista de F. Sin F no queda ninguno.
2. **Un conjunto que desconecta da un corte.** Sea F tal que, sin F, no hay camino. Sea S lo alcanzable desde s sin usar F.
3. t no está en S. Toda arista de S a T está en F: si no, su destino sería alcanzable.
4. La capacidad de ese corte es a lo sumo $|F|$. ■

En un grafo no dirigido vale igual: un corte cuenta una vez cada arista, porque solo una dirección va de S a T.

Demostración: forma con nodos

Red dividida, con capacidad "sin límite" en las aristas originales. Basta un número mayor que V: el corte de s y todos los v_{in} cuesta a lo sumo $V - 2$. Un corte mínimo tiene solo aristas $v_{in} \to v_{out}$.

1. **Corte $\Rightarrow$ separador.** Sea X el conjunto de esos v. Un camino de s a t que no toca X cruzaría el corte por una arista que no está en él. Contradicción.
2. **Separador $\Rightarrow$ corte.** Sea X un separador y A lo alcanzable sin X. Sea S: s, los v_{in} y v_{out} de A, y los x_{in} de X.
3. Una arista original que sale de u_{out}, con u en A, llega a un v de X o de A: se queda en S.
4. Solo cruzan las $x_{in} \to x_{out}$: a lo sumo $|X|$. ■

El programa lee las dos cosas

build/disjoint_demo, extracto

```
Dirigido, sin compartir aristas:
  caminos: 2
    s b c e t
    s a c d t
  corte mínimo: s->a s->b

El mismo grafo, sin compartir nodos:
  caminos: 1
    s a c d t
  separador mínimo: c
```

Idea clave

El corte es una prueba que se puede revisar. Si les muestro 2 caminos disjuntos y 2 aristas que los separan, nadie puede hacer 3.

`test_examples.cpp` lo compara con fuerza bruta en 1500 grafos al azar.

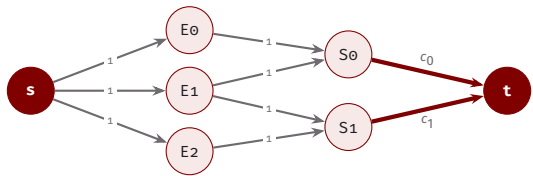
SECCIÓN 06

Asignación con cupos

06

La matrícula, ahora en serio

n estudiantes, m secciones, y cada sección j tiene un cupo $c_j \geq 0$.



- **Cada capacidad es una regla:** 1 en $s \to i$ es "una sola sección". c_j en $j \to t$ es "no más de su cupo".
- Costo: flujo $\leq n$. Con D la suma de los largos de las listas, $O(n(n+m+D))$. Un cupo enorme no agrega aristas.

Pseudocódigo y teorema

pseudocódigo

```
algoritmo asignar_con_cupos(opciones, cupo, n, m)
    red ← red vacía con fuente s y sumidero t
    para i desde 0 hasta n - 1 hacer
        agregar s → estudiante i, capacidad 1
        para cada sección j de opciones[i] hacer
            agregar estudiante i → sección j,
      capacidad 1
        fin para
    fin para
    para j desde 0 hasta m - 1 hacer
        agregar sección j → t, capacidad cupo[j]
    fin para
    k ← flujo máximo de red
    devolver k y las aristas del medio con flujo 1
fin algoritmo
```

Asignación $\Rightarrow$ flujo. Se envía 1 por $s \to i \to j \to t$ para cada par. $j \to t$ recibe a lo sumo c_j.

Flujo entero $\Rightarrow$ asignación. A i le entra a lo sumo 1: una sola sección. A j le entra lo que sale por $j \to t$: a lo sumo c_j.

Integralidad: si no, alguien va media hora a cada sección.

Los greedy tampoco mejoran con cupos

build/quota_demo, extracto

```
Seis estudiantes y tres secciones,
con cupos S0: 2, S1: 1, S2: 2
  E0: S0
  E1: S0
  E2: S2 S1
  E3: S0 S2
  E4: S2
  E5: S0

Los greedy y el árbitro:
  primera opción libre:   4
  menos opciones primero: 4
  fuerza bruta:           5
```

la red

```
Flujo máximo: 5 de 6
Aristas del medio con flujo:
  E0-S0 E1-S0 E2-S1 E3-S2 E4-S2
Estudiantes alcanzables desde s:
  E0 E1 E5
Secciones alcanzables desde s:
  S0
```

E0, E1 y E5 solo pueden ir a S0, con cupo 2. Se queda uno fuera, se asigne como se asigne.

La condición de Hall con cupos

Para un conjunto A de estudiantes: $N(A)$ son las secciones de sus listas, y $c(N(A))$ la suma de sus cupos.

Teorema

$k = \min_A \big(n - |A| + c(N(A))\big)$. Además, los estudiantes A y las secciones B alcanzables desde s en la red residual final cumplen $B = N(A)$ y alcanzan el mínimo: $|A| - c(N(A)) = n - k$.

La cota fácil. De los estudiantes de A se asignan a lo sumo $c(N(A))$, y de los demás a lo sumo $n - |A|$. Vale para todo A.

Corolario. Entran todos si y solo si $|A| \leq c(N(A))$ para todo A. Con cupos 1, es el teorema de Hall clásico.

La igualdad: la capacidad del corte de lo alcanzable

Sea R lo alcanzable. Por la semana 11, su corte vale k.

1. $N(A) \subseteq B$. Sea $i \in A$ y j en su lista. Si $i \to j$ lleva 0, tiene capacidad residual: $j \in R$.
2. Si lleva 1, $s \to i$ está saturada. A i solo se llega por una inversa $j' \to i$ con flujo en $i \to j'$. Como i lleva una unidad, $j' = j$.
3. $B \subseteq N(A)$. A j se llega por $i \to j$ o por la inversa $t \to j$. Como $t \notin R$, se llegó desde un estudiante de R.
4. **La capacidad:** salen $s \to i$ con $i \notin A$, que suman $n - |A|$, y $j \to t$ con $j \in B$, que suman $c(N(A))$. Por el paso 1, ninguna $i \to j$.

Entonces $k = n - |A| + c(N(A))$. ■

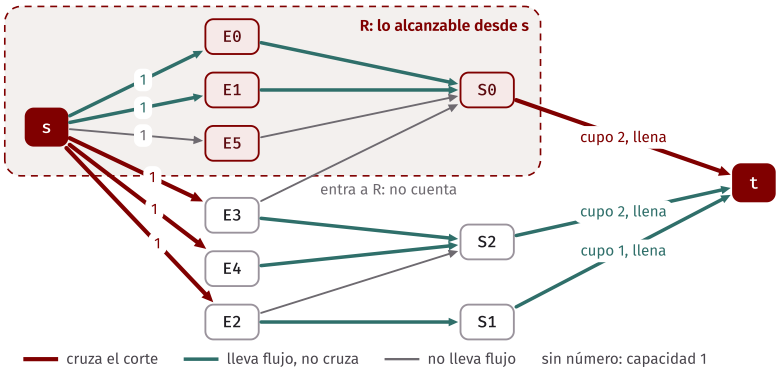
Dos estudiantes, una silla

build/quota_demo, extracto

```
Dos quieren S0, con cupo 1, y S1 queda libre porque ninguno puede ir:
  flujo: 1
  S0 -> t lleva 1, S1 -> t lleva 0
  alcanzables desde s: s E0 E1 S0
Lo mismo, pero E1 también puede ir a S1:
  flujo: 2
  S0 -> t lleva 1, S1 -> t lleva 1
  alcanzables desde s: s
```

- En el primero, S1 $\to$ t tiene capacidad de sobra, pero **a S1 no se llega desde** s. $A = \{\text{E0}, \text{E1}\}$, $N(A) = \{\text{S0}\}$: $2 - 1 = 1$.
- En el segundo, hay un camino hasta la sección libre, y entran los dos.

El grupo que limita, en la red



Corte: s → E2, s → E3, s → E4 y S0 → t, con capacidad 1 + 1 + 1 + 2 = 5, el flujo máximo.

A = {E0, E1, E5}, N(A) = {S0}: |A| − c(N(A)) = 3 − 2 = 1 = 6 − 5.

Lo saturado no es lo alcanzable

En el ejemplo de seis estudiantes, las tres secciones quedaron llenas:

- S0 tiene a E0 y E1, y S2 tiene a E3 y E4: llenas.
- S1 tiene a E2: llena.
- Sin embargo, E2, E3 y E4 caben perfectamente. Lo que falla es $A = \{\text{E0, E1, E5}\}$ con $N(A) = \{\text{S0}\}$: $3 - 2 = 1 = 6 - 5$.

Error común

Decir que limitan “las secciones llenas”. Es el error de la semana pasada: confundir aristas saturadas con aristas del corte. El grupo se lee de lo alcanzable, no de lo saturado.

Más capas: elegir de a varios

Un horario de exámenes: cada curso necesita una hora y un aula.



- Cada camino de s a t es una tupla (curso, hora, aula). Un nodo con límite se divide.
- La red solo ve **pares de capas consecutivas**. No puede decir “este curso no puede usar esta aula”.
- En el caso general no hay arreglo fácil: elegir tríos sin elementos en común de una lista arbitraria es 3-dimensional matching, NP-completo. Se ve en la semana 13.

SECCIÓN 07

Eliminación en un campeonato

07

¿Todavía puede quedar primero?

Cada partido lo gana un equipo. **Quedar primero** incluye empatar. Lo más que x alcanza es $W = w_x + r_x$.

build/baseball_demo

```
equipo   ganados  faltan  máximo  ¿eliminado?
Lima     27       4       31      no (flujo 5 de 5)
Cusco    26       4       30      no (flujo 5 de 5)
Ica      26       4       30      no (flujo 5 de 5)
Piura    25       3       28      sí (flujo 5 de 6)
Callao   20       3       23      sí, a simple vista: Lima ya lo pasó
```

- Lima, Cusco e Ica tienen pendientes 2 partidos por par: 6 en total. Los de Piura son contra Callao.
- Con 28 de tope: Lima gana a lo sumo 1, Cusco 2, Ica 2. Total 5.
- Pero 6 partidos se juegan, y alguien los gana. Sobra una victoria.

Dos simplificaciones

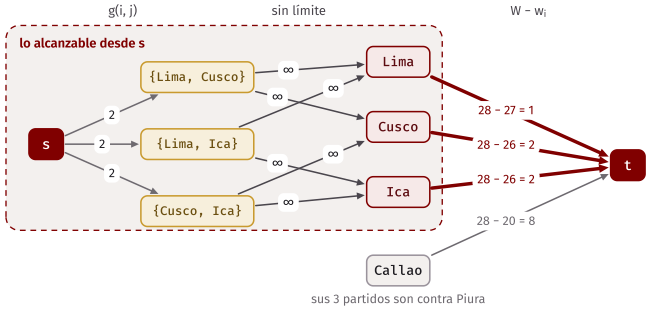
1. **Eliminación trivial.** Si algún $w_i > W$, x está eliminado. Hay que comprobarlo **antes** de construir la red: las capacidades $W - w_i$ serían negativas.
2. x **gana todo lo que le falta.** Si en un resultado donde x queda primero pierde un partido, cambiarlo suma a x y resta a su rival: x sigue primero. Entonces esos partidos desaparecen.

Queda repartir las victorias de los demás partidos sin que nadie pase de W. Por eso es un flujo.

Error común

Mirar solo si W alcanza al líder. Importa lo que los demás se quitan entre sí.

La red



De s tienen que salir G = 6 victorias, y solo pueden cruzar hacia t 1 + 2 + 2 = 5: Piura está eliminado.

- $s \to \{i, j\}$ con $g(i, j)$. $\{i, j\} \to i$ y $\{i, j\} \to j$ sin límite. $i \to t$ con $W - w_i \geq 0$.
- Solo equipos distintos de x, y solo pares con $g(i, j) > 0$. Sea G la suma de esos g. Para Piura, $W = 28$: de s tendrían que salir $G = 6$, y por Lima, Cusco e Ica solo pasan 5.

Teorema

Afirmación

Si ningún w_i supera W: x puede quedar primero si y solo si el flujo máximo vale G.

1. **Resultado $\Rightarrow$ flujo.** Tomemos un resultado donde x gana todo y queda primero. De cada par se envía a i lo que ganó i. De i sale lo que ganó en total, a lo sumo $W - w_i$. Vale G.
2. **Flujo $\Rightarrow$ resultado.** Un flujo entero de valor G satura s. Lo que $\{i, j\}$ envía a i es lo que gana i, y j gana el resto.
3. i termina con w_i más lo que le entra: a lo sumo W. Nadie pasa a x.

Integralidad: no se puede ganar medio partido.

Lo que dice el corte

Sea R los equipos alcanzables desde s, y $g(R)$ los partidos que faltan entre ellos.

Teorema

Si el flujo es menor que G: R no es vacío, y $\sum_{i \in R} w_i + g(R) > |R| \cdot W$.

1. Si un nodo de partido está en S, sus aristas sin límite tienen residual: sus dos equipos están en R.
2. Salen de S las $s \to \{i, j\}$ fuera de S, y las $i \to t$ con $i \in R$. Su capacidad es menor que G:
$$G - \sum_{\{i,j\} \in S} g(i,j) + \sum_{i \in R} (W - w_i) < G$$
3. Y $\sum_{\{i,j\} \in S} g(i,j) \leq g(R)$. De ahí sale la desigualdad.
4. Esos partidos los ganan equipos de R: en cualquier resultado, el promedio de R pasa de W. ■

Una prueba que se revisa con una cuenta

build/baseball_demo, extracto

```
Piura: el grupo que lo elimina es Lima Cusco Ica
  victorias ya ganadas: 79
  partidos entre ellos: 6
  máximo para que nadie pase a Piura: 3 x 28 = 84
  fuerza bruta, ¿puede quedar primero?: no
```

$79 + 6 = 85$ victorias, y solo caben 84. Alguien llega a 29.

- Decisión y certificado comparados con todos los resultados posibles, en 2500 campeonatos al azar.
- Con e equipos, hay $O(e^2)$ nodos y aristas. Con Ford-Fulkerson: a lo sumo G aumentos de $O(e^2)$.

SECCIÓN 08

Selección de proyectos

08

Ahora importa el corte

Cada proyecto tiene una ganancia, tal vez negativa. Si p requiere q, hay que hacer q.

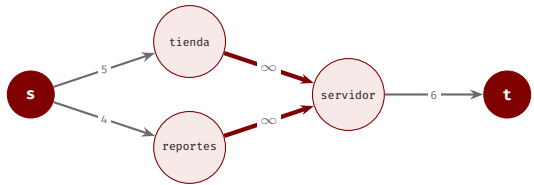
> **Conjunto cerrado** *(closed)*
>
> Si p está en X y p requiere q, entonces q está en X. Se busca uno de máxima ganancia.

datos y salida de build/projects_demo, resumidos

```
tienda 5, reportes 4, app 9, servidor -6, api -8, auditoria 3, robot 2, sensor -5
requisitos: tienda, reportes y app piden servidor; app y auditoria piden api; robot pide sensor
Greedy, paquete por paquete: (ninguno)
```

Tienda con servidor pierde 1, y reportes con servidor pierde 2. **Juntos** pagan el servidor una vez.

La red



- $s \to p$ con la ganancia, si es positiva. $q \to t$ con el costo, si es negativa.
- $p \to q$ con capacidad **infinita** si p requiere q.
- Un corte reparte los proyectos en dos lados: **los de S son los elegidos**.

Idea clave

Un corte es una decisión binaria por nodo. Con costos, y con aristas infinitas que prohíben lo incompatible, un corte mínimo elige lo mejor.

Teorema

Afirmación

Los cortes finitos son los conjuntos cerrados, con $X = S - \{s\}$. Su capacidad es $P -$ ganancia(X), con P la suma de las ganancias positivas.

1. **Cerrado $\Rightarrow$ corte finito.** Una arista infinita que salga de S tendría p en X y q fuera. No puede ser.
2. **Corte finito $\Rightarrow$ cerrado.** Si $p \in X$ requiere $q \notin X$, la arista infinita saldría de S.
3. **La capacidad.** Salen $s \to p$ con $p \notin X$, lo que *dejamos de ganar*, y $q \to t$ con $q \in X$, lo que *pagamos*.
4. ganancia(X) = positivas dentro $-$ costos dentro $= P-$ positivas fuera $-$ costos dentro $= P-$ capacidad. ■

Ganancia máxima $= P-$ corte mínimo. Los elegidos son lo alcanzable desde s.

En el ejemplo

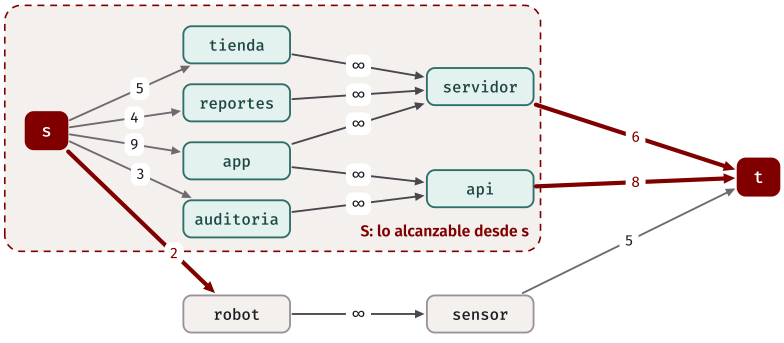
build/projects_demo, extracto

```
Corte mínimo:
  suma de ganancias positivas: 23
  corte mínimo:                16
  ganancia máxima:             7
  elegidos: tienda reportes app
    servidor api auditoria
  fuerza bruta:                7
```

Arista		Significa
servidor $\to$ t	6	se paga
api $\to$ t	8	se paga
s $\to$ robot	2	se renuncia

- $5 + 4 + 9 + 3 - 6 - 8 = 7$.
- Robot queda fuera: ganar 2 no paga un sensor de 5.

El corte, en la red completa



Corte: servidor → t (6) + api → t (8) + s → robot (2) = 16. Ganancia: P − corte = 23 − 16 = 7.

Elegidos, S sin contar a s: tienda reportes app servidor api auditoria.

Por qué infinito, y cuánto es infinito

build/projects_demo, extracto

```
Si los prerrequisitos tuvieran capacidad 1:
  la cuenta dice:              17
  elegidos: tienda reportes app auditoria robot
  ¿respetan los requisitos?    no
```

- Un requisito no se puede pagar para saltárselo. Con capacidad finita, un corte lo hace.
- **Cuánto:** el corte $\{s\}$ cuesta P, así que el mínimo es a lo sumo P. Basta $P + 1$.
- El flujo no pasa de P. Esas aristas nunca se saturan, y por eso **lo leído es cerrado**.
- `INT_MAX` es mala idea: al sumarle flujo, da overflow. Usen `long long`.

SECCIÓN 09

Reconocer un problema de flujo

09

Cómo leer la solución

Problema	Se calcula	Dónde está la respuesta
Matching, asignación	Flujo	Aristas del medio con flujo 1
Caminos disjuntos	Flujo	Aristas con flujo 1, descompuestas, sin ciclos
Menger: lo que desconecta	Corte	De R hacia afuera, en la red original
Hall: el grupo que limita	Corte	Estudiantes y secciones de R
Eliminación: ¿vive?	Valor	¿Es igual a G?
Eliminación: el grupo	Corte	Equipos de R
Proyectos	Corte	Proyectos de R

R es lo alcanzable desde s en la red residual **al final**. No toda arista saturada cruza el corte.

Señales

A favor

- Un recurso que se reparte: cupos, victorias, rutas.
- Elecciones de a uno: $s \to$ persona con capacidad 1.
- Límites de “a lo sumo”: una capacidad.
- Una decisión binaria por objeto, con costos: un corte.

En contra

- Una regla que une tres cosas a la vez: la red no recuerda de dónde viene una unidad.
- Costos por combinación: algunos caben, otros no.
- Un grafo que no es bipartito: esta red no sirve.
- Una restricción de “al menos”.

Ante la duda

Escriban las dos direcciones. Si una no sale, la demostración dice qué parte del modelo está mal.

“Al menos dos de cada grupo”

build/lower_bound_demo, extracto

```
Grupo A: 4 personas. Grupo B: 2 personas. Turnos: T1 con cupo 3, T2 con cupo 2.
Sin restricción:
  total 5, grupo A 4, grupo B 1
Capacidad 2 en s -> grupo, que significa "a lo sumo dos":
  total 4, grupo A 2, grupo B 2
Cota inferior 2 en s -> grupo:
  total 5, grupo A 3, grupo B 2
```

Error común

Una capacidad dice “a lo sumo”. El flujo cero las respeta todas, y no cumple la regla.

Cotas inferiores

Cada arista $u \to v$ debe llevar entre ℓ y c. Se quita lo obligatorio y se contabiliza aparte.

1. $u \to v$ queda con capacidad $c - \ell$.
2. Fuente nueva s^* con $s^* \to v$ de capacidad ℓ. Sumidero nuevo t^* con $u \to t^*$ de capacidad ℓ.
3. Una arista $t \to s$ sin límite.
4. **Hay solución si y solo si un flujo máximo de s^* a t^* satura todas las aristas de s^*.**
5. Si la hay, se quita $t \to s$ y se sigue aumentando de s a t.

No se demuestra aquí: es un ejercicio. Sí se comprueba: coincide con una fuerza bruta en 1500 redes pequeñas, incluida la decisión de si hay solución.

SECCIÓN 10

Cierre

10

Laboratorio

Asignar estudiantes a secciones, en `lab/section_assignment`.

- Cada estudiante puede ir a ciertas secciones, y cada sección tiene un cupo. Asignen a la mayor cantidad posible.
- Un verificador revisa la asignación: nadie dos veces, solo opciones de su lista, y cupos respetados.
- Implementen Edmonds-Karp, o reutilicen su código de flujo.

Extensión opcional, en `lab/section_bottleneck`.

- Cuando no entran todos, digan qué grupo limita el resultado.
- El verificador recalcula $N(A)$, y comprueba que $|A| - c(N(A))$ sea justo lo que falta.

> **Antes de empezar**
>
> Lean los límites. No toda representación de un grafo cabe en ellos.

Lo que hay que llevarse

- **Reducción:** transformar, leer, y demostrar **las dos direcciones**.
- **Matching:** capacidades 1, y las aristas del medio con flujo 1. $O(VE)$ sin nodos aislados.
- **Integralidad:** la lectura necesita un flujo entero, y Ford-Fulkerson lo da.
- **Descomposición:** un flujo entero son caminos y ciclos.
- **Nodos:** se dividen en $v_{in} \to v_{out}$. s y t no.
- **Menger:** máximo de caminos disjuntos = mínimo que desconecta.
- **Hall con cupos:** lo alcanzable explica lo que falta. Lo saturado, no.
- **Eliminación:** primero lo trivial, x gana todo, y se reparte el resto.
- **Proyectos:** ganancia $= P -$ corte. Infinito es $P + 1$, en `long long`.
- **Una capacidad es "a lo sumo".** "Al menos" necesita una cota inferior.

Para la próxima clase

- Leer los apuntes de la semana 12, y hacer sus ejercicios. La semana 13 trata de P, NP y reducciones.
- **Laboratorio:** `lab/section_assignment`. Opcional: `lab/section_bottleneck`.

Preguntas

Dibujen la red para un matching bipartito de 3 por 3 y den un flujo máximo.
Demuestren las dos direcciones de la reducción de matching bipartito a flujo.
¿Cómo se limita a 1 el número de caminos que pasan por un nodo?
En la selección de proyectos, ¿por qué las aristas de prerrequisito tienen capacidad infinita?
Dos estudiantes quieren la misma sección con cupo 1, y otra sección queda libre. ¿Cómo se ve eso en la red?