









Flows III: Flow Applications

Contenido

1. Vertex/Edge Cut	
2. Menger's Theorem	
3. Matchings, Coverings and Independent Sets	

Contenido

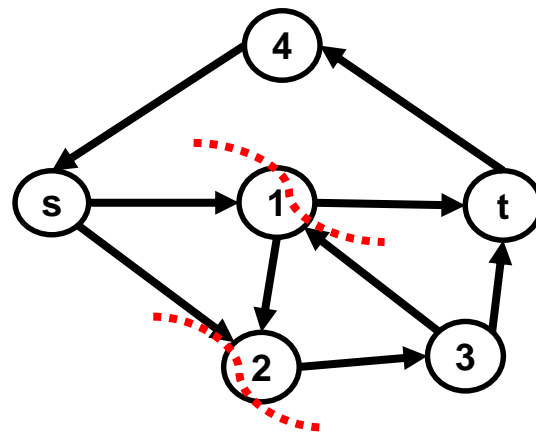
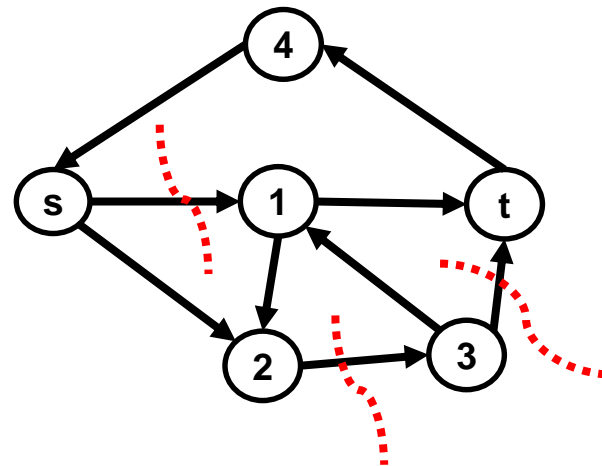
1. Vertex/Edge Cut	
2. Menger's Theorem	
3. Matchings, Coverings and Independent Sets	

Min Edge cut / Min Vertex Cut

❑ **Definición (Edge cut):** En un grafo $G(V, E)$ con nodos s y t , un $(s - t)$ edge cut X es un conjunto de aristas de E tal que cualquier path que va desde s hasta t tiene que pasar por alguna arista de X . Asimismo, el mínimo edge cut es aquel que tenga menor cantidad de aristas $|X|$.

❑ **Definición (Vertex cut):** En un grafo $G(V, E)$ con nodos s y t , un $(s - t)$ vertex cut X es un conjunto de nodos de $V - \{s, t\}$ tal que cualquier path que va desde s hasta t tiene que algún nodo de X . Asimismo, el mínimo vertex cut es aquel que tenga menor cantidad de nodos $|X|$.

❑ **Observación:** Un edge cut siempre existe, pero un vertex cut puede no existir y se da solo en caso haya una arista $s \rightarrow t$



Min Edge cut

□ **Lemma 1:** Sea un grafo G (dirigido o no dirigido) con capacidades unitarias. Sea (S^*, T^*) el min cut y sea X^* el min edge cut. Entonces $c(S^*, T^*) = |X^*|$

Demostración:

- **Demostremos** $|X^*| \leq c(S^*, T^*) \dots (1)$

En cualquier cut (S, T) , las aristas del corte forma un edge cut X de tamaño $c(S, T)$. Por contradicción supongamos que no, entonces existe un path desde s a t que no pasa por ninguna arista del corte. Sin embargo, como $s \in S$ y $t \in T$, en alguna parte del path debe aparecer una arista (u, v) tal que $u \in S$ y $v \in T$ (contradicción).

En particular, eso quiere decir que para un min cut (S^*, T^*) , el min edge cut $|X^*| \leq c(S^*, T^*)$

- **Demostremos** $c(S^*, T^*) \leq |X^*| \dots (2)$

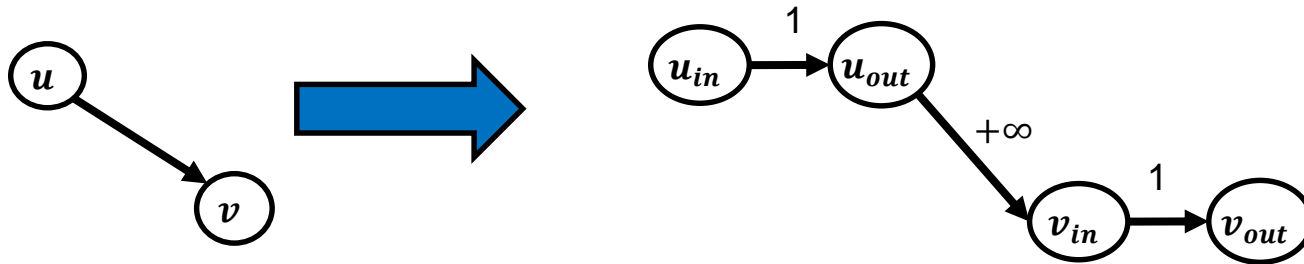
Si tengo un edge cut X cualquier, demostremos que puedo construir un corte (S, T) con $c(S, T) \leq |X|$. Para ello, definimos el conjunto S como todos los nodos que s puede alcanzar sin pasar por aristas de X (y $T = V - T$). Entonces, por contradicción, todas las aristas del corte deben ser parte de X , ya que, de lo contrario, existiría una arista $(u, v) \notin X$ tal que $u \in S$ y $v \in T$, pero esto implicaría que existe un path $s \rightsquigarrow v$ que no usa ninguna arista de X , por lo que v debió pertenecer a S (contradicción). En particular si aplico esto para el min edge cut, tengo que $c(S^*, T^*) \leq |X^*|$

Finalmente, juntamos (1) y (2) para llegar a $c(S^*, T^*) = |X^*|$ ■

Min Vertex Cut

Para obtener el min vertex cut tendremos que hacer unos ajustes al grafo $G(V, E)$. Construiremos $G'(V', E')$ de la siguiente forma.

- ☐ Para cada nodo u del grafo original G construiremos 2 copias u_{in} y u_{out}
- ☐ $\forall u \in V$ crearemos una arista (u_{in}, u_{out}) en G' con capacidad 1
- ☐ $\forall (u, v) \in E$ crearemos una arista (u_{out}, v_{in}) en G' con capacidad $+\infty$
- ☐ Obtendremos el min vertex cut al utilizar Max Flow desde s_{out} hasta t_{in}



Min Vertex cut

□ **Lemma 2:** Sea un grafo $G(V, E)$ donde no exista una arista $s \rightarrow t$ y sea G' el grafo construido con el procedimiento anterior con copias de vértices. Sea (S^*, T^*) el min cut en G' y sea X^* el min vertex cut en G . Entonces $c(S^*, T^*) = |X^*|$

Demostración:

Como no existe la arista $s \rightarrow t$ en el grafo original, entonces todo path desde s a t en el grafo original, debe pasar por un nodo $u \neq s, t$, por lo que, si cogemos cualquier flujo y hacemos un flow decomposition, este debe pasar por al menos una arista (u_{in}, u_{out}) en G' con capacidad 1. Por lo que el max flow (min cut) es finito.

- **Demostremos** $|X^*| \leq c(S^*, T^*) \dots (1)$

Partiendo del min cut (S^*, T^*) , como su capacidad es finita, solo puede tener aristas (u_{in}, u_{out}) que están ligadas a un solo nodo u en G , por lo que para todo $(u, v) \in E$, $u_{in}, v_{out} \in S$ o $u_{in}, v_{out} \in T$.

Podemos demostrar que el conjunto de todos estos nodos, forma un vertex cut X en G . Por contradicción, supongamos que no, eso significa que existe un path en G desde s a t que no pasa por ningún nodo de X .

Considerando que $s \in S$ y $t \in T$, en alguna parte del path debe cumplirse que exista una arista que pasa por el corte y solo pueden ser aristas $(u_{in}, u_{out}) \Rightarrow |X^*| \leq c(S^*, T^*)$

Min Vertex cut

□ **Lemma 2:** Sea un grafo $G(V, E)$ donde no exista una arista $s \rightarrow t$ y sea G' el grafo construido con el procedimiento anterior con copias de vértices. Sea (S^*, T^*) el min cut en G' y sea X^* el min vertex cut en G . Entonces $c(S^*, T^*) = |X^*|$

Demostración:

- **Demostremos** $c(S^*, T^*) \leq |X^*| \dots (2)$




Partiendo de un vertex cut X cualquiera, demostremos que puedo construir un corte (S, T) en G' con $c(S, T) \leq |X|$. Para ello, definimos el conjunto S como todos los nodos que s puede alcanzar sin pasar por aristas (u_{in}, u_{out}) para todo nodo $u \in X$ (y $T = V - T$). Entonces, por contradicción, todas las aristas del corte deben estar ligadas a un nodo que es parte de X , ya que, de lo contrario, existiría una arista $(u_{in}, u_{out}) \notin X$ tal que $u_{in} \in S$ y $u_{out} \in T$, pero esto implicaría que existe un path $s \rightsquigarrow u_{out}$ que no usa ninguna arista de relacionadas a algún nodo de X , por lo que u_{out} debió pertenecer a S (contradicción). En particular si aplico esto para el min edge cut, tengo que $c(S^*, T^*) \leq |X^*|$

Finalmente, juntamos (1) y (2) para llegar a $c(S^*, T^*) = |X^*|$ ■

Problemas

❏ CSES - Police Chase

Contenido

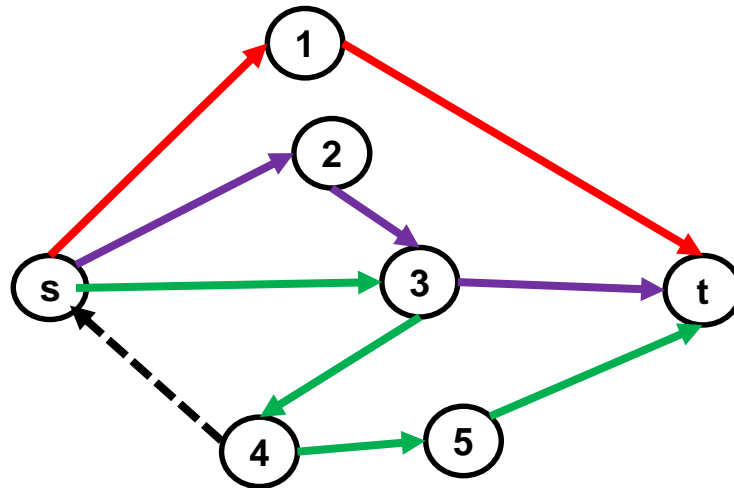
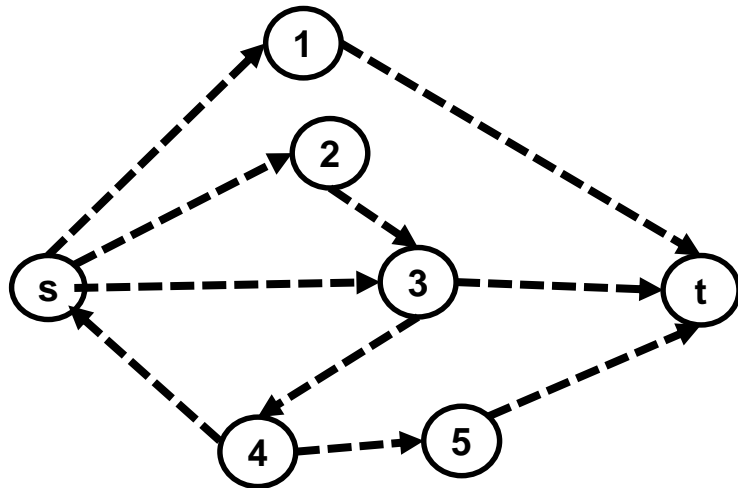
1. Vertex/Edge Cut	
2. Menger's Theorem	
3. Matchings, Coverings and Independent Sets	

Menger's Theorem – Edge Version

❑ **Definición (Edge Disjoint Paths):** Son paths que, entre sí, no tienen ninguna arista en común.

Un problema clásico es hallar la máxima cantidad de edge disjoint paths desde s a t .

Ejemplo: [CSES - Distinct Routes](#)



Menger's Theorem – Edge Version

□ **Menger's Theorem (Edge Version):** Sea un grafo $G(V, E)$. El máximo número de edge-disjoint paths desde s a t es igual al $\min(s - t)$ edge cut.

Demostración:

Definamos $G'(V, E)$ como el grafo G con capacidades unitarias en cada arista. Sea Q el conjunto de edge-disjoint paths con máxima cardinalidad y sea X el min edge cut. Además sea f el max flow en G' y (S, T) el min cut. Por el teorema de max flow min cost y por el **lemma 1** sabemos que $|f| = c(S, T) = |X|$. Así que solo hace falta demostrar que $|f| = |Q|$.

- **Demostremos $|Q| \leq |f| \dots (1)$**

Si a cada uno de los paths de Q le ponemos flujo 1 en cada arista, la unión de los $|Q|$ paths forma trivialmente una función de flujo válida en G' con valor de flujo $|Q|$.

$\Rightarrow |Q| \leq |f|$ (por definición de max flow)

- **Demostremos $|f| \leq |Q| \dots (2)$**

El flujo f se puede descomponer en $|f|$ flow paths (y algunos flow cycles) por el Flow Decomposition Theorem. Además, por ser G' un grafo de capacidades unitarias, estos paths deben ser edge-disjoints

$\Rightarrow |f| \leq |Q|$

Finalmente, juntamos (1) y (2) para llegar a $|f| = |Q|$ ■

Menger's Theorem — Edge Version

Debido al teorema de Menger, podemos fácilmente obtener un algoritmo para obtener el conjunto máximo de disjoint-edge paths.

❑ Max flow part

Corremos el algoritmo de Dinic para hallar el max flow en el grafo con capacidades unitarias en $O(E * \min(\sqrt{E}, V^{2/3}))$.

❑ Flow Decomposition part

Luego podemos usar el algoritmo de la demostración del **Flow Decomposition Theorem**, para descomponer el max flow en varios **flow paths**. El valor del max flow $|f|$ estará acotado por E debido a las capacidades unitarias, o también por V en caso el grafo no tenga multi-edges. Por cada unidad de flujo utilizaremos podemos encontrar un path de la respuesta con un dfs en $O(E)$, dando una complejidad total de $O(EV)$ para el caso de simple graph.

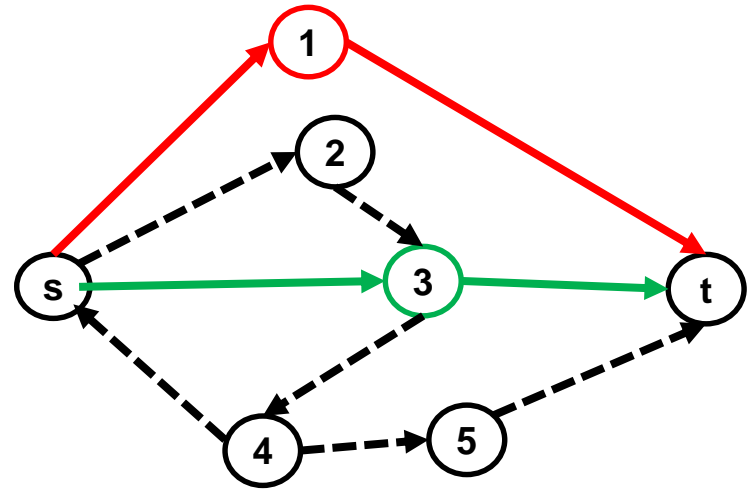
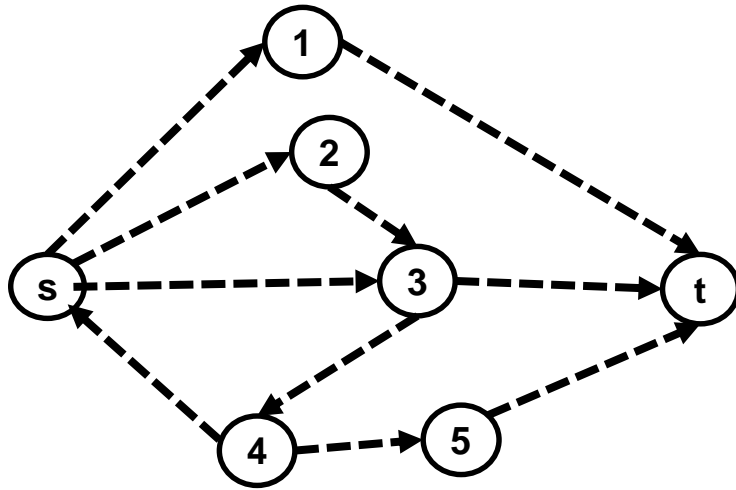
⇒ **Complejidad total:** $O(EV)$ (predomina flow decomposition part)

Optimization: Esta última parte se puede incluso mejorar un poco más, guardando la última arista visitada, similar al procedimiento para hallar un blocking flow, lo cuál daría una complejidad de $O(E)$ para descomponerlo en paths, siendo la complejidad total igual a la de Dinic por ser la predominante.

Menger's Theorem – Vertex Version

□ **Definición (Vertex Disjoint Paths):** Son paths que, entre sí, no tienen ningún nodo en común, a excepción de s y t .

Un problema clásico es hallar la máxima cantidad de vertex disjoint paths desde s a t .



Menger's Theorem – Vertex Version

□ **Menger's Theorem (Vertex Version):** Sea un grafo $G(V, E)$ donde no existe la arista $s \rightarrow t$. El máximo número de vertex-disjoint paths desde s a t es igual al min $(s - t)$ vertex cut.

Demostración:

Definamos $G'(V, E)$ el grafo construido para hallar el vertex cut (con nodos duplicados). Sea Q el conjunto de vertex-disjoint paths con máxima cardinalidad y sea X el min vertex cut. Además sea f el max flow en G' y (S, T) el min cut. Por el teorema de max flow min cut y por el **lemma 2** sabemos que $|f| = c(S, T) = |X|$. Así que solo hace falta demostrar que $|f| = |Q|$.

• Demostremos $|Q| \leq |f| \dots (1)$

Partiendo de Q , generaremos una función f' de la siguiente forma. Para cada path $P \in Q$, colocaremos una arista (u_{in}, u_{out}) con flujo 1 por cada nodo $u \in P$ y una arista (u_{out}, v_{in}) con flujo 1 por cada arista $(u, v) \in P$. f' cumple trivialmente *flow conservation* por ser unión de paths.

Además cumple también con las *capacity constraints* debido a que las aristas (u_{out}, v_{in}) tienen capacidad infinita en G' y las aristas (u_{in}, u_{out}) solo aparecerán una vez en f' debido a que cada path es vertex-disjoint. $\Rightarrow f'$ es una función válida de flujo.

Además $|f'| = |Q|$ por ser combinación de paths.

$\Rightarrow |f'| = |Q| \leq |f|$ (por definición de max flow)

Menger's Theorem – Vertex Version

□ **Menger's Theorem (Vertex Version):** Sea un grafo $G(V, E)$ donde no existe la arista $s \rightarrow t$. El máximo número de vertex-disjoint paths desde s a t es igual al $\min(s - t)$ vertex cut.

Demostración:

- **Demostremos** $|f| \leq |Q| \dots (2)$

El flujo f se puede descomponer en $|f|$ flow paths (y algunos flow cycles) por el Flow Decomposition Theorem. Además, como G' tiene capacidades unitarias en cada arista que pertenece a un nodo, estos paths deben ser vertex-disjoint.

$$\Rightarrow |f| \leq |Q|$$



Finalmente, juntamos (1) y (2) para llegar a $|f| = |Q|$ ■

- ❖ El algoritmo para construir el conjunto de vertex-disjoint paths es muy similar al del edge-disjoint paths.
- ❖ No es necesario utilizar capacidades infinitas, podemos utilizar capacidades unitarias y la demostración sigue siendo válida.
- ❖ Hay que tener en cuenta que si el grafo tiene una arista $s \rightarrow t$, si bien es cierto que ya no existe vertex cut, sí existe un set de vertex disjoint paths. Para hallar el máximo podemos ver que la arista $s \rightarrow t$ no puede compartir ningún nodo intermedio con ningún otro path por lo que siempre podemos agregarlo al set de respuesta sin empeorar la situación.

Problemas

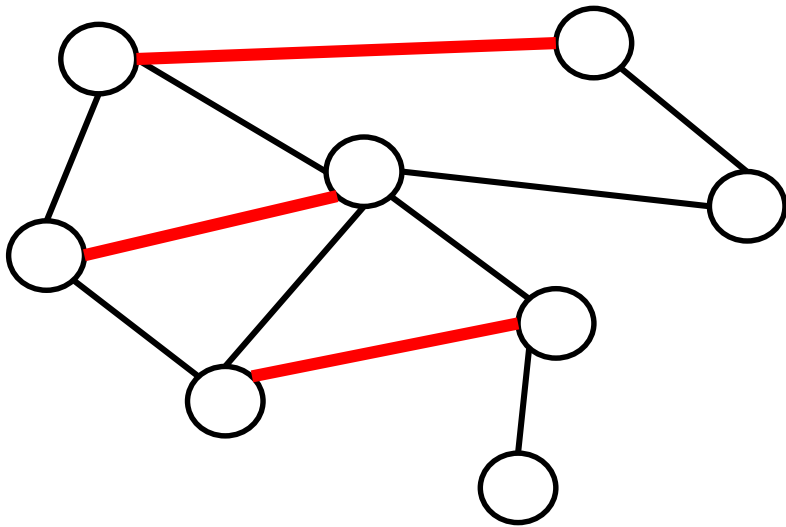
❑ CSES - Distinct Routes

Contenido

1. Vertex/Edge Cut	
2. Menger's Theorem	
3. Matchings, Coverings and Independent Sets	

Matching

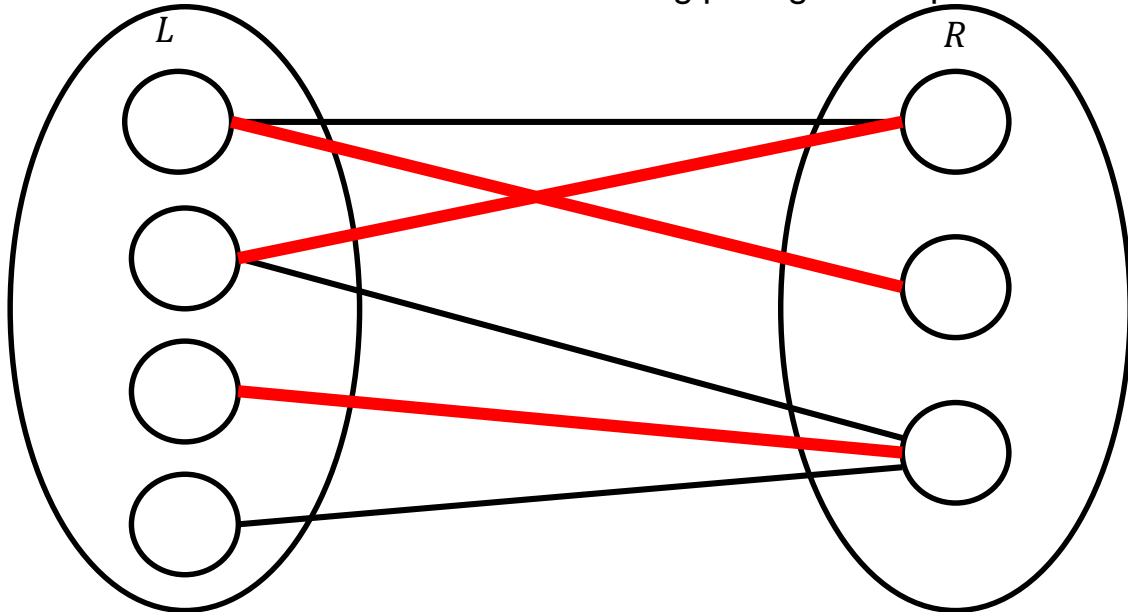
- ❑ **Definición (Matching):** Para un grafo $G(V, E)$ **no dirigido**, un matching es un conjunto $M \subset E$ de aristas tal que no exista ningún par de aristas que compartan algún nodo.



- ❑ **Observación:** Trivialmente el conjunto vacío de aristas es un matching.
- ❑ Estamos interesados en encontrar el matching con máxima cardinalidad ($|M|$), el maximum matching.

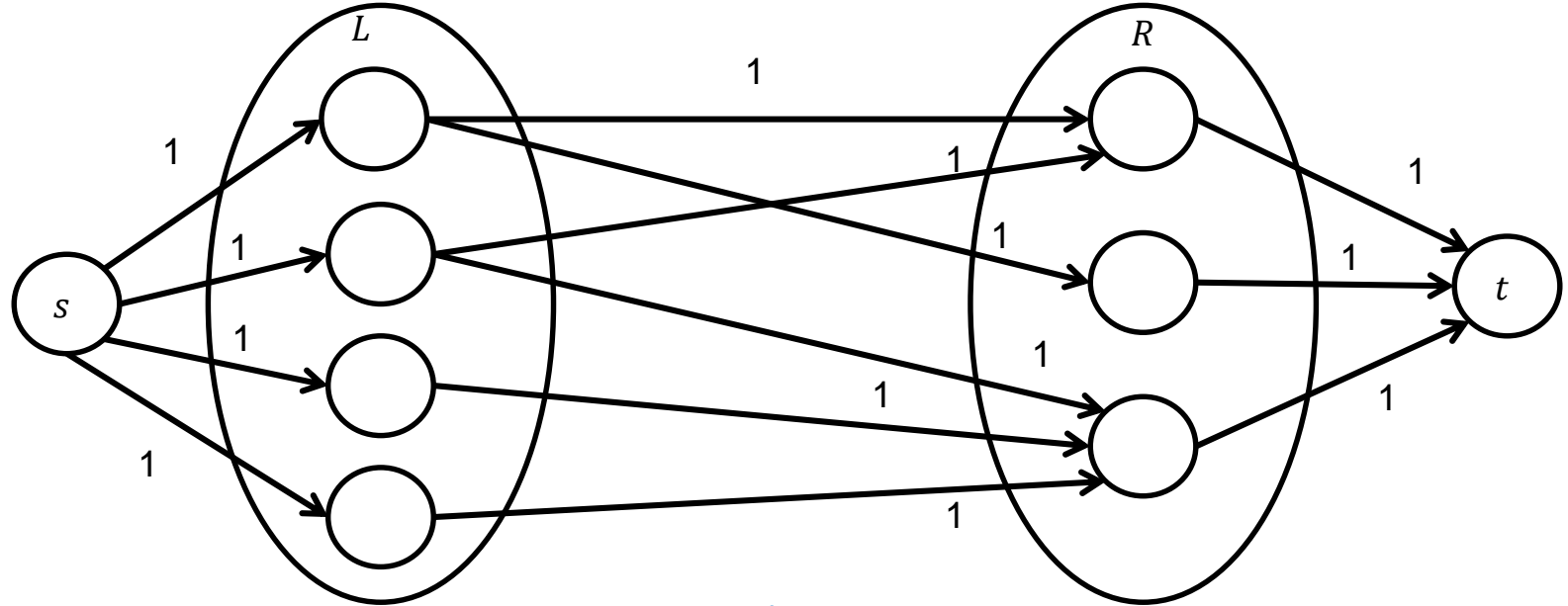
Maximum Bipartite Matching

- ❑ Recordemos que un grafo es **bipartito** si es que no tiene ningún ciclo impar. Además todo grafo bipartito se puede dividir en dos conjuntos L y R tal que no existe ninguna arista entre dos nodos del mismo conjunto. Es decir, el grafo se puede bicolorar.
- ❑ **Grafos bipartitos clásicos:** árboles, bosques, chess-board graphs.
- ❑ El problema de encontrar el maximum matching para grafos bipartitos es mucho más sencillo.



Maximum Bipartite Matching

□ Transformemos en un problema con max flow, usando capacidades unitarias. Sea el grafo $G(V = L \cup R,$



Maximum Bipartite Matching

❑ **Lemma 3:** Sea un **grafo bipartito no dirigido** $G(V = L \cup R, E)$, construimos el grafo dirigido $G'(V', E')$ con capacidades unitarias con el procedimiento anterior. Entonces el valor del max flow f en G' es igual al tamaño del maximum matching M de G .

Demostración:

• **Demostremos** $|M| \leq |f| \dots (1)$

Partamos del maximum matching M . Entonces para cada arista $(u, v) \in M$ tal que $u \in L$ y $v \in R$, creamos los siguientes valores de flujo

❖ $f'(s, u) = 1$

❖ $f'(u, t) = 1$

❖ $f'(u, v) = 1$

Como todos los flujos son unitarios y solo suceden sobre un subset de E' , entonces f' cumple con las *capacity constraints*.

Además el *flow conservation* también se mantiene trivialmente.

$\Rightarrow f'$ es función de flujo válida en G'

Además $|f'| = |M|$ debido a que cada arista de M aumenta el flujo neto en s

$\Rightarrow |M| = |f'| \leq |f|$ (debido a que f es máximo)

Maximum Bipartite Matching

□ **Lemma 3:** Sea un **grafo bipartito no dirigido** $G(V = L \cup R, E)$, construimos el grafo dirigido $G'(V', E')$ con capacidades unitarias con el procedimiento anterior. Entonces el valor del max flow f en G' es igual al tamaño del maximum matching M de G .

Demostración:

- **Demostremos** $|f| \leq |M| \dots (2)$

Dado el max flow f , podemos formar un matching $M' = \{(u, v) \mid f(u, v) = 1 \wedge u \neq s, t \wedge v \neq s, t\}$.

Podemos demostrar que M' es matching debido a que para cada arista $(u, v) \in M'$ sabemos que u solo tiene una arista entrante en G' (s, u), por lo tanto, usando *flow conservation*, solo puede tener una arista saliente con flujo 1, que debe ser (u, v) .

De igual forma, v solo tiene una arista saliente (v, t), por lo tanto, solo puede tener una arista entrante con flujo 1 y debe ser (u, v) .

\Rightarrow No existe otra arista en M' que contenga a u o v

$\Rightarrow M'$ es matching en G

Además $|f| = |M'|$ debido a que cada arista de M' está relacionada con exactamente una arista que parte desde s por lo que contribuye en 1 al flujo neto de s y al valor del flujo.

$\Rightarrow |f| = |M'| \leq |M|$ (debido a que M es máximo)

Finalmente, el lemma se demuestra combinando (1) y (2) ■

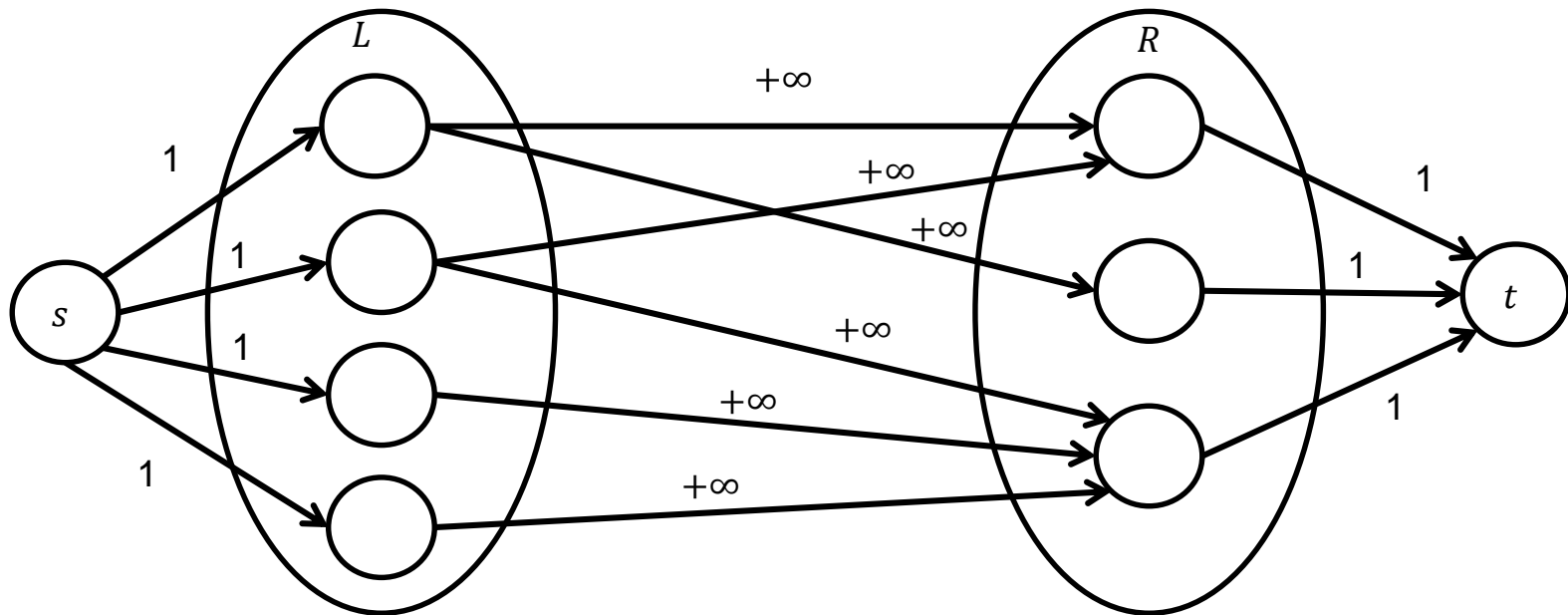
Maximum Bipartite Matching

Asumiendo un grafo bipartito

- ❑ **Cardinalidad del maximum bipartite matching:** Solo tenemos que correr el algoritmo de max flow para hallar el valor del maximum matching
- ❑ **Algoritmo para obtener las aristas del matching:** La demostración anterior nos da un algoritmo para obtener las aristas del matching: basta con ver las aristas con flujo 1 (que no sean las de s o t)
- ❑ **Complejidad:** El grafo de flujo del bipartite matching cumple con tener **capacidades unitarias** y tener 1 arista entrante o 1 arista saliente para todos los nodos distinto de s, t , entonces es un **unit network** y Dinic funcionará en $O(E'\sqrt{V'})$. Considerando que $|V'| = |V| + 2$ y que $|E'| = E + 2|V| = O(E)$ entonces la complejidad total también es $O(E\sqrt{V})$.
- ❑ **Detalle de implementación:** Ten cuidado de especificar que el grafo de flujo generado sea **dirigido**, de lo contrario la demostración ya no aplica.
- ❑ **General Maximum Matching:** Si el grafo no es bipartito, entonces esta misma idea no funciona, pero existe otro algoritmo llamado **Blossom Algorithm** que halla el maximum matching para cualquier grafo en $O(V^3)$ o $O(EV^2)$ dependiendo de la implementación, pero es muy poco común en los concursos.

Maximum Bipartite Matching

- ❑ **Permisividad en las capacidades:** Si en lugar de poner capacidad 1 a las aristas (u, v) del matching (las intermedias que van de L a R), le ponemos cualquier otra capacidad positiva, seguirá siendo correcto, debido a que el flujo está restringido por el 1 de flujo entrante en L y el 1 de flujo saliente de R . Podemos incluso poner capacidades infinitas. Se puede demostrar que la complejidad se mantiene en $O(E\sqrt{V})$

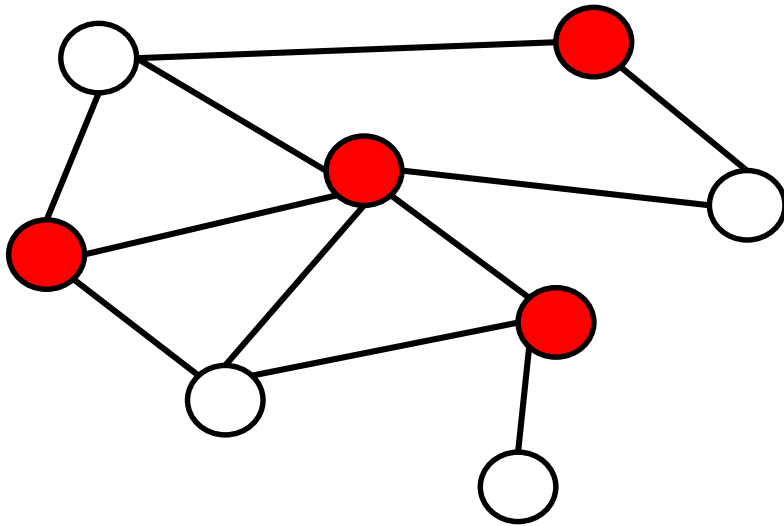


Problemas

- ❑ [CSES - School Dance](#)
- ❑ [Library Checker - Matching On Bipartite Graph](#) – To test efficiency

Vertex Cover

- ❑ **Definición (Vertex Cover):** Para un grafo $G(V, E)$ **no dirigido**, un vertex cover es un conjunto $C \subset V$ de nodos tal que cualquier arista $e \in E$ es incidente a algún nodo de C .



- ❑ **Observación:** Trivialmente V es un vertex cover
- ❑ Estamos interesados en encontrar el vertex cover con mínima cardinalidad ($|C|$), el minimum vertex cover.

Minimum Vertex Cover

□ **Lemma 4:** En cualquier grafo **no dirigido** $G(V, E)$, el tamaño del maximum matching es menor o igual al tamaño del minimum vertex cover.

Demostración:

Sea M' cualquier matching, entonces cualquier vertex cover C' necesita tener al menos 1 nodo por cada arista de M' . Además como las aristas de M' no tienen ningún nodo en común, cada nodo puede cubrir a lo más una arista del matching M' .

$$\Rightarrow |M'| \leq |C'|.$$

En particular, si usamos a M como el maximum matching, y denotamos a C como el minimum vertex cover

$$\Rightarrow |M| \leq |C| \blacksquare$$

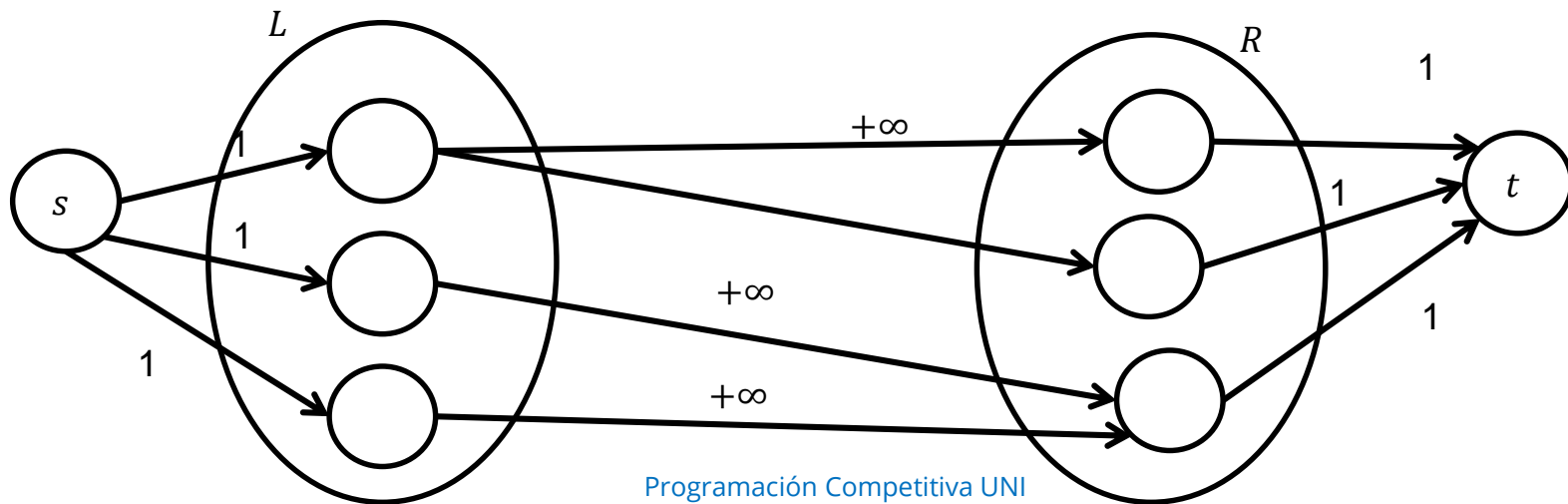
Minimum Vertex Cover on Bipartite Graphs

□ **Konig's Theorem:** Sea un **grafo bipartito no dirigido** $G(V = L \cup R, E)$, entonces el tamaño del maximum matching es igual al tamaño del minimum vertex cover.

Demostración:

Sea M el maximum matching y C el minimum vertex cover. Por el lemma 4 sabemos que $|M| \leq |C|$ (sin necesidad de ser bipartito). Por lo tanto, solo falta demostrar que $|C| \leq |M|$.

Partamos del maximum matching M y demostremos que podemos formar un vertex cover C' de la siguiente forma. Formemos el grafo dirigido de flujo G' con el que se obtiene el maximum matching del lemma 3, pero usando capacidades infinitas en las aristas que van de algún nodo de L a un nodo de R



Minimum Vertex Cover on Bipartite Graphs

❑ **Konig's Theorem:** Sea un **grafo bipartito no dirigido** $G(V = L \cup R, E)$, entonces el tamaño del maximum matching es igual al tamaño del minimum vertex cover.

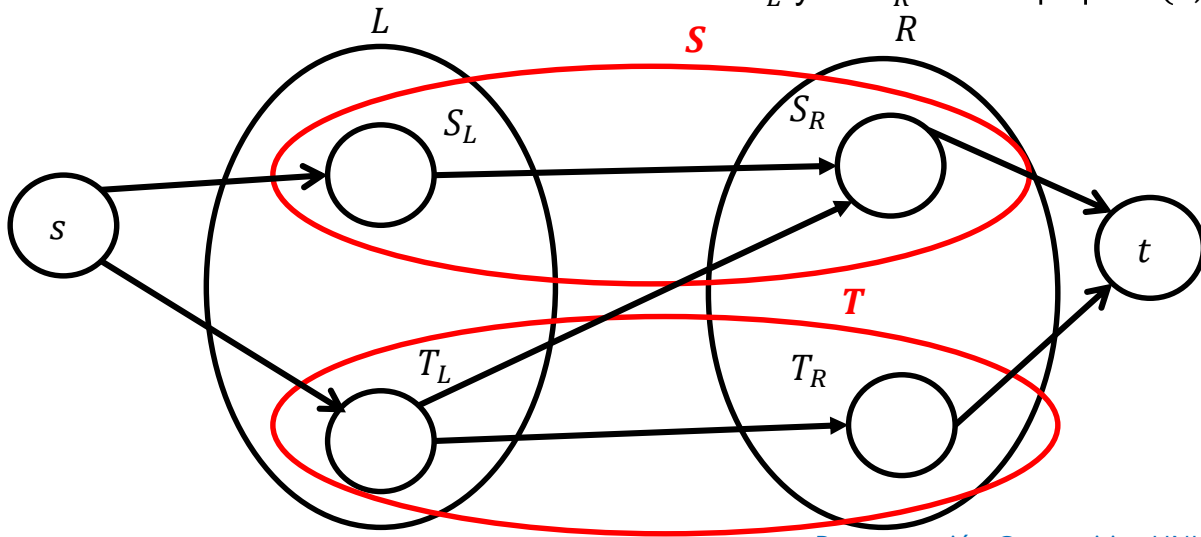
Demostración:

Sea (S, T) un min cut en G' . Por el Max Flow Min Cut theorem y por el lemma 3 sabemos que $c(S, T) = |M|$.

Sean $S_L = S \cap L$, $S_R = S \cap R$, $T_L = T \cap L$, $T_R = T \cap R$. Entonces podemos formar $C' = T_L \cup S_R$.

Además en G' no pueden existir aristas de S_L hacia T_R ya que sino el corte tendría capacidad infinita.

\Rightarrow Las únicas aristas del corte son de s a T_L y de S_R a $t \Rightarrow |M| = c(S, T) = |T_L| + |S_R| = |C'|$.



Por último C' es un vertex cover debido a que las únicas aristas que no podrían estar cubiertas serían las que hay desde S_L a T_R , pero estas no existen por el hecho de que el corte es finito.

$\Rightarrow |C| \leq |C'| = M$ ■

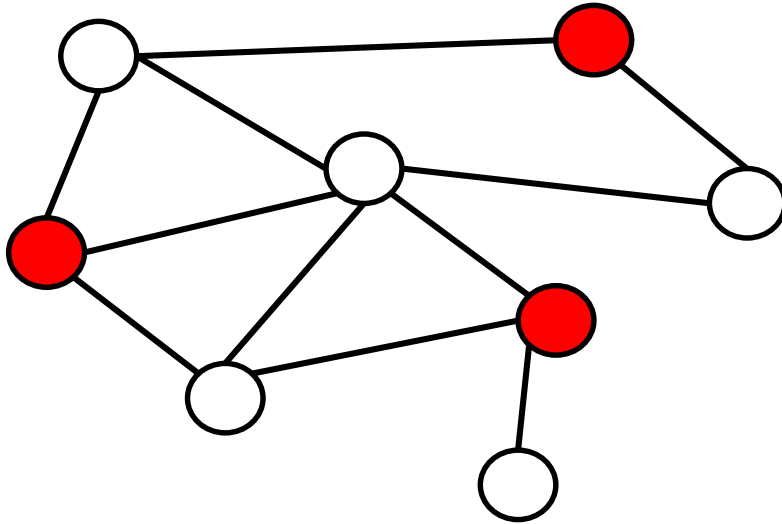
Minimum Vertex Cover on Bipartite Graphs

Asumiendo un grafo bipartito

- ❑ **Cardinalidad del min vertex cover:** Basta con hallar el max flow en G' .
- ❑ **Algoritmo para reconstruir el min vertex cover:** La demostración anterior nos da un algoritmo constructivo para poder hallar un minimum vertex cover. $T_L \cup S_R$
- ❑ **Complejidad:** Correr el algoritmo de max flow puede salir en $O(E\sqrt{V})$ y hallar un min cut puede hacerse en $O(E)$ al igual que obtener T_L y S_R . Por lo que conseguir el min vertex cover sigue siendo dominado por $O(E\sqrt{V})$.
- ❑ **Permisividad de las capacidades:** La demostración anterior es directa cuando utilizamos las capacidades infinitas. Si utilizáramos full capacidades unitarias, no podríamos utilizar *cualquier* min cut para completar la prueba, pero podemos utilizar uno en particular, el que es formado con S igual al conjunto de todos los nodos alcanzables desde s en la red residual final (el clásico min cut). Se puede demostrar que tampoco pueden existir aristas entre S_L y T_R en este corte, por lo que también es válido usarlo de esa forma.
- ❑ **General Minimum Vertex Cover:** Si el grafo no es bipartito, hallar el minimum vertex cover es NP-Hard. Es decir, no se conoce algún algoritmo polinomial para resolverlo. De todas formas es fácil utilizar fuerzabruta para encontrar uno en $O(E * 2^V)$.

Independent Set

- ❑ **Definición (Independent Set):** Para un grafo $G(V, E)$ **no dirigido**, un independent set es un conjunto de nodos $I \subset V$ tal que no exista ningún par de nodos $u, v \in I$ tal que $(u, v) \in E$.



- ❑ **Observación:** Trivialmente el conjunto vacío de nodos es un independent set
- ❑ Estamos interesados en encontrar el independent set con máxima cardinalidad ($|I|$), el maximum independent set.

Maximum Independent Set

□ **Gallai's Theorem (Primera parte):** Sea $G(V, E)$ un grafo no dirigido cualquiera. Sea I su maximum independent set y sea C su minimum vertex cover. Entonces $|C| + |I| = |V|$.

Demostración:

- **$V - C$ es un independent set**

Partiendo del minimum vertex cover C , podemos formar $I' = V - C$. Por contradicción supongamos que existen dos nodos $(u, v) \in I'$ tal que exista una arista $(u, v) \in E$, entonces esta arista no estaría cubierta por ningún nodo de C , lo cual sería una contradicción.

$\Rightarrow I'$ es independent set $\Rightarrow |I'| = |V| - |C| \leq |I| \dots (1)$

- **$V - I$ es un vertex cover**

Partiendo del maximum independent set I , podemos formar $C' = V - I$. Por contradicción, supongamos que exista una arista $(u, v) \in E$ tal que no sea cubierta por C' , entonces esto significaría que ni u ni v pertenecen a C' , lo que significa que pertenecen a I . Sin embargo esto iría en contradicción con la definición de independent set de I .

$\Rightarrow C'$ es vertex cover $\Rightarrow |C'| = |V| - |I| \geq |C| \dots (2)$

Finalmente combinamos (1) y (2) para llegar a $|C| + |I| = |V|$ ■

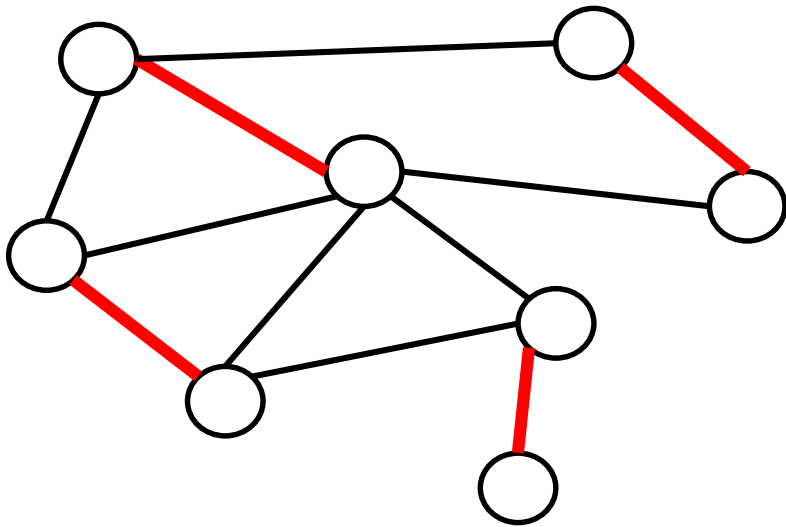
Maximum Independent Set on Bipartite Graphs

Asumiendo un grafo bipartito

- ❑ **Cardinalidad del max independent set:** Basta con hallar el max flow en G' (el grafo de flujo de matching). La respuesta es $|V| - \max flow$.
- ❑ **Algoritmo para reconstruir el max independent set:** La demostración anterior nos da un algoritmo constructivo para poder hallar un max independent set. Basta con encontrar un minimum vertex cover con el algoritmo de la sección anterior y luego cogemos todos los nodos restantes.
- ❑ **Complejidad:** Hallar un minimum vertex cover en un grafo bipartito es $O(E\sqrt{V})$ y hallar los nodos restantes toma $O(V)$. Por lo que conseguir el max independent set sigue siendo dominado por $O(E\sqrt{V})$.
- ❑ **General Maximum Independent Set:** Si el grafo no es bipartito, si bien el teorema de Gallai sigue siendo válido, no podemos hallar el minimum vertex cover en tiempo polinomial por lo que este método ya no se podría usar. Peor aún, hallar el maximum independent set es NP-Hard. Es decir, no se conoce algún algoritmo polinomial para resolverlo. De todas formas es fácil utilizar fuerzabruta para encontrar uno en $O(E * 2^V)$.

Edge Cover

- ❑ **Definición (Edge Cover):** Para un grafo $G(V, E)$ **no dirigido**, un edge cover es un conjunto A de aristas tal que cualquier nodo $u \in V$ es incidente a alguna arista de A



- ❑ **Observación:** Si el grafo tiene nodos aislados, no existe edge cover.
- ❑ **Observación:** Si no existen nodos aislados, trivialmente el conjunto E es un edge cover
- ❑ Estamos interesados en encontrar el edge cover con mínima cardinalidad ($|A|$), el minimum edge cover

Minimum Edge Cover

□ **Lemma 5:** En cualquier grafo **no dirigido** $G(V, E)$, el tamaño del maximum independent set es menor o igual al tamaño del minimum edge cover.

Demostración:

Sea I' cualquier independent set, entonces cualquier edge cover A' necesita cubrir cada nodo de I' como mínimo. Además como los nodos en I' no pueden ser adyacentes, entonces cada arista de A' puede cubrir a lo más un nodo del independent set I' .

$$\Rightarrow |I'| \leq |A'|.$$

En particular, si usamos a I como el maximum independent set, y denotamos a A como el minimum edge cover.

$$\Rightarrow |I| \leq |A| \blacksquare$$

Minimum Edge Cover

□ **Gallai's Theorem (Segunda parte):** Sea $G(V, E)$ un grafo no dirigido **sin vértices aislados**. Sea M su maximum matching y sea A su minimum edge cover. Entonces $|A| + |M| = |V|$.

Demostración:

- $|M| + |A| \geq |V| \dots (1)$

Sea I el maximum independent set y C el minimum vertex cover, entonces por el lemma 5 tenemos $|I| \leq |A|$ y por el teorema de Gallai (primera parte) tenemos que $|I| + |C| = |V|$.

Además por el teorema de König tenemos que $|C| = |M|$. Juntando todo tenemos que $|M| + |A| \geq |V|$

- $|A| + |M| \leq |V| \dots (2)$

Partiendo del maximum matching M , sabemos que este matchea a $2 * |M|$ vértices. Además, como no hay vértices aislados, el resto de $|V| - 2 * |M|$ nodos tiene al menos una arista adyacente. Para cada uno de esos nodos u toma cualquiera de estas aristas, y sea v el otro nodo de esa arista, entonces v tiene que haber sido matcheado por M , porque, de lo contrario significaría que podríamos agregar (u, v) al matching y ya no sería máximo.

Sea K el conjunto de todas estas aristas formadas con el paso anterior ($|K| = |V| - 2 * |M|$), entonces podemos formar $A' = M \cup K$.

$\Rightarrow A'$ es edge cover porque M cubre todos los nodos matcheados y K los no matcheados.

Además $|A'| = |M| + |V| - 2 * |M| = |V| - |M|$. $\Rightarrow |A| \leq |A'| = |V| - |M| \Rightarrow |A| + |M| \leq |V|$

Finalmente combinamos (1) y (2) para llegar a $|A| + |M| = |V|$ ■

Minimum Edge Cover

□ **Corolario 6 (Gallai + Konig):** Sea $G(V, E)$ un grafo bipartito no dirigido sin vértices aislados. Sea I su maximum independent set y sea A su minimum edge cover. Entonces $|A| = |I|$.

Demostración:

Sea M el maximum matching y C el minimum vertex cover. Por el teorema de Konig sabemos que $|M| = |C|$. Por el teorema de Gallai, asumiendo que no hay nodos aislados, tenemos que $|C| + |I| = |V| = |M| + |A|$. Combinando ambas ecuaciones, tenemos que $|A| = |I|$. ■

Minimum Edge Cover on Bipartite Graphs

Asumiendo un grafo bipartito

- ❑ **Cardinalidad del min edge cover:** Basta con hallar el max flow en G' (el grafo de flujo de matching). La respuesta es $|V| - \max flow$.
- ❑ **Algoritmo para reconstruir el min edge cover:** La demostración anterior nos da un algoritmo constructivo para poder hallar un min edge cover. Basta con encontrar un maximum matching M y para todos los nodos no matcheados cogemos una de sus aristas y formamos K siendo el minimum edge cover $M \cup K$
- ❑ **Complejidad:** Hallar un maximum matching en un grafo bipartito es $O(E\sqrt{V})$ y hallar las aristas restantes toma $O(|E|)$. Por lo que conseguir el minimum edge cover sigue siendo dominado por $O(E\sqrt{V})$.
- ❑ **General Minimum Edge Cover:** Si el grafo no es bipartito, el teorema de Gallai sigue siendo válido, y sabemos que podemos hallar el maximum matching con el algoritmo Blossom en $O(EV^2)$ o en $O(V^3)$ y a partir de ahí podemos hallar fácilmente el minimum edge cover.

Resumen – 4 fantásticos

Todas la propiedades solo están definidas en grafos no dirigidos

Restricciones	Propiedad	Fórmula
Ninguna	Inecuaciones	$ maximum\ matching \leq minimum\ vertex\ cover $
		$ maximum\ independent\ set \leq minimum\ edge\ cover $
Ninguna	Gallai	$ maximum\ independent\ set + minimum\ vertex\ cover = V $
Sin nodos aislados		$ maximum\ matching + minimum\ edge\ cover = V $
Grafo bipartito		$ max\ flow = maximum\ matching $
Grafo bipartito	Konig	$ maximum\ matching = minimum\ vertex\ cover $
Grafo bipartito sin nodos aislados	Konig + Gallai	$ maximum\ independent\ set = minimum\ edge\ cover $

Resumen – 4 fantásticos

Algoritmos para encontrar los matchings, cover e independent sets en **grafos bipartitos** $G(V = L \cup R, E)$.

Objetivo	Algoritmo
Maximum Maching	Construye G' como el grafo dirigido de capacidades unitarias. Las aristas con flujo 1 forman las aristas del matching
Minimum Vertex Cover	Construye G' , el grafo de flujo del matching. Halla el min cut (S, T) usando el algoritmo de dfs desde la fuente en el grafo residual. Halla $S_R = S \cap R$, $T_L = T \cap L$, y el minimum vertex cut es $S_R \cup T_L$
Maximum Independent Set	Halla cualquier minimum vertex cover C . Forma $I = V - C$ y será el maximum independent set
Minimum Edge Cover	Halla cualquier maximum matching M . Por cada nodo que no esté matcheado, coge 1 de sus aristas y forma en total el conjunto K . Forma el minimum edge cover $M \cup K$

Referencias

- ❑ [1] Cornell University. Supplementary lecture notes on flow.
<https://www.cs.cornell.edu/courses/cs6820/2019fa/handouts/maxflow-mincut.pdf>
- ❑ [2] Rich Schwartz. Applications of Max Flow Min Cut.
<http://www.math.brown.edu/reschwar/M1230/maxflow.pdf>
- ❑ [3] Frank de Zeeuw. Cover and independent sets.
<https://archiveweb.epfl.ch/dcg.epfl.ch//wp-content/uploads/2018/10/GT-4-Covers.pdf>
- ❑ [4] Adamant. König's and Hall's theorems through minimum cut in bipartite graphs.
<https://codeforces.com/blog/entry/105697>