Discreta II: Demos final

Mansilla, Kevin Gaston*

July 26, 2023

1) **Cual es la complejidad del algoritmo de Edmonds-Karp? Probarlo (Nota: en la prueba se definen unas distancias, y se prueba que esas distancias no disminuyen en pasos sucesivos de EK. Ud. puede usar esto sin necesidad de probarlo.)**

Diremos que un lado se vuelve critico al pasar de f_k a f_{k+1} si se **satura** o **vacia**. Entonces cuántas veces puede un lado volverse crítico?

Supongamos que un lado se vuelve critico en el paso k y luego en el paso j, donde $k < j$. Entonces:

$$s \dots \underbrace{xz}_{critico} \dots t$$

pasa de f_k a f_{k+1}.

Si $\overrightarrow{xz} \in E$ se satura tenemos que $d_k(z) = d_k(x) + 1$, para volverse critico otra vez, o bien se vacia o bien se vuelve a saturar, pero para esto último primero tiene que devolver flujo.

En cualquier caso, $\exists l : k \leqslant l \leqslant j$ en donde devolvio flujo, es decir, tal que para pasar de f_l a f_{l+1}, devolvimos flujo (lado backward), como se muestra en el siguiente camino:

$$s \dots \overleftarrow{zx} \dots t, f_l \to f_{l+1}$$

por EK sabemos que $d_l(x) = d_l(z) + 1$ (distancias no disminuyen).

Si lo que tenemos es $\overleftarrow{zx} \in E$ (backward), con un camino $s \dots \overleftarrow{zx} \dots t \Rightarrow d_k(x) = d_k(z) + 1$ el analisis es similar, se volvio critico porque se vacio, entonces para volver a ser critico debe saturarse o volverse a vaciar, para esto último debe haber enviado flujo.

Entonces, $\exists l : k \leqslant l \leqslant j$ para pasar de f_l a f_{l+1}, enviamos flujo, como se muestra en el siguiente camino:

$$s \dots \overrightarrow{xz} \dots t, f_l \to f_{l+1}$$

Entonces, por EK sabemos: $d_l(z) = d_l(x) + 1$.

En cualquier caso, tenemos en terminos de f_k y f_{k+1}

$$d_k(z) = d_k(x) + 1 \text{ (fordward)}$$
$$d_k(x) = d_k(z) + 1 \text{ (backward)}$$

*kevingston47@gmail.com

Entonces

$$\begin{aligned} d_l(t) = d_l(x) + b_l(x) &= d_l(z) + 1 + b_l(x) \\ &= \{\text{Como las distancias no disminuyen}\} \\ &\geqslant d_k(z) + b_k(x) + 1 = d_k(x) + 1 + b_k(x) + 1 = d_k(x) + b_k(x) + 2 \\ &\Rightarrow d_l(t) = d_k(t) + 2 \end{aligned}$$

Conclusión: Una vez que un lado se vuelve crítico solo puede volver a ser crítico si la distancia entre s y t aumenta en por lo menos 2. Un lado puede volverse crítico a lo sumo $\frac{n-1}{2}$ veces entonces es $O(n)$.

1. Para pasar de f_k a f_{k+1} al menos un lado se vuelve crítico.

2. Hay m lados.

3. Cada lado se vuelve critico $O(n)$ veces.
   Por lo tanto el número total de pasos es $O(nm)$.
   La complejidad de hallar un camino aumentate es $O(m)$ por BFS.
   Entonces la complejidad total es $O(m) * O(nm) = O(nm^2)$.

2) **Probar que si, dados vértices x, z y flujo f definimos a la distancia entre x y z relativa a f como la longitud del menor $f-$camino aumentante entre x y z, si es que existe tal camino, o infinito si no existe o 0 si $x = z$, denotandola por $d_f(x, z)$, y definimos $d_k(x) = d_{f_k}(s, x)$, donde f_k es el $k-$ésimo flujo en una corrida de Edmonds-Karp, entonces $d_k(x) \leqslant d_{k+1}(x)$**

Vamos a demostrar que $d_k(x) \leqslant d_{k+1}(x)$.

Sea $A = \{y : d_{k+1}(y) < d_k(y)\}$. Queremos ver que $A = \emptyset$ vamos a suponer que no es $\emptyset$ y llegar a un absurdo.

Suponemos $A \neq \emptyset$, es decir, que tiene 1 o muchos elementos, y de entre todos los elementos elegimos $x \in A$ tal que:

$$d_{k+1}(x) \leqslant d_{k+1}(y) \ \forall y \in A \tag{1}$$

Como $x \in A$, entonces $d_{k+1}(x) < d_k(x) \Rightarrow d_{k+1} < \infty$, por definición de A. Entonces, existe $f_{k+1}-$ca entre s y x, tomemos uno de menor longitud, es decir,

$$\underbrace{s \ldots . x}_{longitud = d_{k+1}(x)}$$

Observación importante: $d_k(s) = d_{k+1}(s) = 0, s \notin A \ldots s \neq x$.

Sea z el vertice inmediatamente anterior a x en ese camino $(s \ldots zx)$. Como el camino es de **longitud minima**, entonces:

$$d_{k+1}(x) = d_{k+1}(z) + 1 \tag{2}$$

En particular, $d_{k+1}(z) < d_{k+1}(x) \Rightarrow_{Por(1)} z \notin A$. (tendría que ser $\leq$)

Como $z \notin A$, entonces se da lo contrario:

$$d_k(z) \leqslant d_{k+1}(z) \Rightarrow d_k(z) < \infty \tag{3}$$

Como $d_{k+1}(z) < d_{k+1}(x) < \infty$. Por (3) existe un f_k-ca entre s y z, y de ellos voy a tomar uno de longitud minima.

Primero supongamos que $\overrightarrow{zx} \in E$. Entonces, si suponemos primero que esta saturado ($f_k(\overrightarrow{zx}) = c(\overrightarrow{zx})$), pero $\underbrace{s \dots z}_{f_{k+1}} x$ es un ca relativo. Entonces, $f_{k+1}(\overrightarrow{zx}) < c(\overrightarrow{zx})$. Esto implica que para pasar de f_k a f_{k+1} el lado se uso backward.

Entonces, para pasar de f_k a f_{k+1} debe haber un ca de la forma $s \dots \overleftarrow{xz} \dots t$ y como estamos usando EK este camino es de longitud minima.

$$d_k(z) = d_k(x) + 1 \tag{4}$$

Entonces:

$$\begin{aligned} d_k(x) =_{(4)} d_k(z) - 1 \leqslant_{(3)} d_{k+1}(z) - 1 = (d_{k+1}(x) - 1) - 1 = d_{k+1}(x) - 2 \\ < d_k(x) - 2 \\ \Rightarrow d_k(x) < d_k(x) - 2 \\ \Rightarrow 0 < 2 \text{ Absurdo} \end{aligned}$$

Llegamos a un absurdo por suponer que $f_k(\overrightarrow{zx}) = c(\overrightarrow{zx})$. Ahora veamos el otro caso, $f_k(\overrightarrow{zx}) < c(\overrightarrow{zx})$ (no esta saturado). Pero teniamos el $f - k$ ca, $s \dots z$ entonces existe un f_{k+1} ca del tipo $s \dots \overrightarrow{zx}$ (no se si es de longitud minima). Entonces:

$$\begin{aligned} d_k(x) &\leqslant d_k(z) + 1 \\ &\leqslant d_{k+1}(z) + 1 = d_{k+1}(x) \\ &< d_k(x) \Rightarrow 0 < 0 \text{ Absurdo} \end{aligned}$$

Por lo tanto en cualquier caso, si $\overrightarrow{zx} \in E$, llegamos a un absurdo.

Ahora supongamos que $\overrightarrow{xz} \in E$, en este caso también llegamos a un absurdo dependiendo si $f_k(\overrightarrow{xz}) = 0$ o no.

Entonces, si $f_k(\overrightarrow{xz}) = 0$ como $\underbrace{s \dots z}_{f_k} x$ es ca. Entonces, zx es backward, $s \dots \overleftarrow{zx}$, entonces $f_{k+1}(\overrightarrow{xz}) > 0$. Para pasar de f_k a f_{k+1} usamos un ca, $s \dots \overleftarrow{xz} \dots t \Rightarrow_{EK} d_k(z) = d_k(x) + 1$ y es un absurdo como antes.

Por otro lado, si $f_k(\overrightarrow{xz}) > 0$ entonces tenemos que $\underbrace{s \dots \overleftarrow{zx}}_{f_{k+1}}$ es un ca, por lo que $d_k(x) \leqslant d_k(z) + 1$ y llegamos a un absurdo otra vez.

Entonces hemos llegado a un absurdo en todos los casos, por lo tanto, $A = \emptyset$ y $d_n < d_{n+1}$ (las distancias no disminuyen).

3) **Cual es la complejidad del algoritmo de Dinic? Probarla en ambas versiones: Dinitz original y Dinic-Even. (no hace falta probar que la distancia en networks auxiliares sucesivos aumenta)**

Corolario 1 *La complejidad de cualquier algoritmo que use NA, es*

$$\underbrace{O(n)}_{\text{Num de NA}} \left(\underbrace{O(m)}_{\text{Contruir NA con BFS}} + \text{Complejidad de hallar un FB} \right)$$

Por el corolario anterior, basta ver que la complejidad de hallar un **flujo bloqueante** es $O(nm)$.

En la versión original, cada camino entre s y t se encuentra usando DFS, pero el NA tiene la propiedad extra que se garantiza que toda búsqueda DFS nunca va a tener que hacer backtranking, pues cada vértice con lado entrante tiene lado saliente. Pero esta propiedad tiene el costo de tener que mantenerla entre camino y camino, revisando el NA. A esta operación se la llama **podar**.

Primero veamos la complejidad de encontrar todos los caminos. La construcción de cada camino es $O(r)$ donde r es el número de niveles. Como $r < n$ podemos decir que esta parte es $O(n)$, luego de cada camino se borran del NA los lados saturados, pero esto es simplemente recorrer una vez más el camino, así que es $O(n)$

Cada camino satura al menos un lado, y luego borramos ese lado (del NA), por lo tanto hay a lo sumo $O(\#\,\text{lados del NA}) = O(m)$ caminos.

Por lo tanto la complejidad de hallar todos los caminios es $O(n)O(m) = O(nm)$, pero falta la complejidad de todos los "Podar", pero cómo funciona podar?

Se recorre en los niveles desde t a s mirando cada vertice y sino tiene lados lo borramos a él y a todos los lados que llegaban a él.

Podar tiene dos partes:

1. Revisa los vertices.
2. Si es necesario borrar los lados.

En 1) es para cada vertice, entonces es $O(1)$ y como hay n vertices, el costo total de un podar es $O(n)$. Cuántos podar hay? Hay uno por camino y había $O(m)$ caminos, por lo tanto hay $O(m)$ podar y la complejidad de revisar los vertices es $O(n)$, por lo tanto esta parte también es $O(nm)$.

Por otra parte en 2) no queremos la complejidad de un podar sino la de todos (la de un podar puede ser muy grande y va reduciendo a medida que se borran lados). Como la segunda parte cada vez que se llama borra al menos un lado, la suma total sobre todos los podar es $O(m)$.

$$\text{Complejidad Total} = O(nm) + O(nm) + O(m) = O(nm)\square$$

Dinitz-Even (Versión Occidental): La diferencia de la versión de Even es que el NA no esta depurado y por lo tanto un DFS puede demorar mas de $O(n)$, pero va depurando a medida que se corre DFS, borrando los lados por los cuales tenemos que retroceder.

Algorithm 1 Calculo del flujo bloqueante del NA

```
1: g=0
2: flag=1
3: while flag do
4:     k = s (pivote)
5:     Inicializar el camino p en s
6:     while x ≠ t and flag do
7:         if Γ+(x) ≠ ∅ then AVANZAR(x, p, g, NA)
8:         else if x ≠ s then RETROCEDER(x, p, g, NA)
9:         elseflag=0
10:        if x == t then INCREMENTAR(g)
11: return g
```

Una vez que se tiene creado el camino para aumentar el flujo, el INCREMENTAR es $O(n)$ pues la longitud del camino es a la sumo n, e INCREMENTAR aumenta el flujo a lo largo de ese camino y borra todos los lados saturados. Además la complejidad de AVANZAR es $O(1)$ ya que consiste en buscar el primer vértice en la lista de $\Gamma^+(x)$, agregar un lado al camino y cambiar x. Por último, RETROCEDER también es $O(1)$ pues simplemente borramos un lado y cambiamos quien es x.

Teorema 1 *Complejidad de Dinitz-Even es* $O(n^2m)$

Prueba

Como antes basta ver que la complejidad del **flujo bloqueante** es $O(nm)$. Una corrida es una "palabra" que se obtiene con DFS de la forma:

$$IA \dots AAAIAARARR \dots AA \dots IA \dots$$

donde:

- $A \to AVANZAR \Rightarrow O(1)$
- $R \to RETROCEDER \Rightarrow O(1)$
- $I \to INCREMENTAR \Rightarrow O(r) = O(n)$

Calcular la complejidad de la palabla? Analizo un solo DFS hasta que no pueda AVANZAR, es decir, una subpalabra.

$A \dots AX$ con $X = I$ ó $X = R$, entonces la complejidad de cada palabra? Como hay r niveles y cada A incrementa el nivel del pivote, hay a lo sumo r $A's$ seguidas, entonces:

$$\text{Complejidad}(A \dots AX) = O(r) + complejidad(X)$$

X puede ser I o R, entonces si es I es $O(r)$ y si es R es $O(1)$. Entonces

$$\text{Complejidad}(A \dots AX) = \begin{cases} O(r) + O(1) & \to \quad X = R \\ O(r) + O(r) & \to \quad X = I \end{cases}$$

Entonces $= O(r) \Rightarrow O(n)$.

Cuantas palabras hay? Si $X = R$, R borra un lado y si $X = I$, I borra al menos un lado. Entonces, cada $A \dots Ax$ borra al menos un lado por lo que hay a los sumo $O(m)$ palabras (# lados del NA).

Entonces son $O(m)$ palabras con complejidad $O(n)$ cada una, entonces $O(nm)$ total.□

4) **Cual es la complejidad del algoritmo de Wave? Probarla. (no hace falta probar que la distancia en networks auxiliares sucesivos aumenta).**

La complejidad del algoritmo de Wave es $O(n^3)$, pero como trabaja con NA's por el corolario 1 basta ver que la complejidad de hallar un **flujo bloqueante** es $O(n^2)$.

Para poder encontrar un flujo bloqueante, se hace una serie de ciclos de olas hacia adelante y olas hacia atrás. En cada ola hacia adelante hacemos una serie de forwardbalance (fwb) y en cada ola hacia atrás hacemos una serie de backwardbalance (bwb). En cada ola hacia adelante (salvo en la última) al menos un vertice se bloquea, si los fwb no bloquean ningún vertice entonces todos quedan balanceados y es la última ola.

Como los vertices nunca se desbloquean, hay a lo sumo $O(n)$ olas. Entonces tenemos que analizar dos casos, los fwb y los bwb y ver su complejidad.

En cada **fwb** vamos saturando lados salvo quizas uno. Entonces dividamos la complejidada en dos:

1. S = complejidad total de los fwb saturado.
2. P = complejidad de fwb parcial.

P es facil, en cada fwb solo se ejecuta una vez, entonces es $O(1)$ pero en una ola se pueden realizar $n - 2$ fwb, entonces:

$$\begin{aligned} P &= \#\,\text{total de fwb} \\ &= \#\,\text{de fwb en una ola} * \text{olas} \\ &= O(n) * O(n) \end{aligned}$$

En cuanto a S, cada vez que se ejecuta uno de esta parte, se borra un vecino de $\Gamma^+(x)$. Por lo tanto, será a lo sumo $O(d(x))$ en cada x, entonces:

$$s = \sum_x O(d(x)) = O(m)$$

.

Con **bwb** también lo dividimos en casos que vaciamos un lado V y los que parcialmente vaciamos Q. El analisis de Q es igual que el de P, es $O(1)$ en el bwb, por lo tanto

$$\begin{aligned} Q &= \#\,\text{total de bwb(x)} \\ &= \#\,\text{de bwx(x) en una ola} * \text{olas} \\ &= O(n) * O(n) \end{aligned}$$

Y con respecto a V, cada vez que lo hacemos en un $bwb(x)$ borramos un vertice de $M(x)$ (vecinos hacia atrás). Por lo que esta acotado por:

$$O(\#\,\text{número de elementos máximos de } M(x)) = O(d(x)) = O(m)$$

Un detalle importante es que $\Gamma^+(x)$ es fijo y le voy sacando elementos y $M(x)$ al inicio es vacio, pero le voy agregando y luego removiendo vertices.

Supongamos que $\overrightarrow{yx}$ se vacía, entonces y se borra de $M(x)$, pero más adelante podría ser que lo vuelva a poner en $M(x)$. Pero $\overrightarrow{yx}$ solo se puede vaciar si x está bloqueado, pues de otra forma no pudiera devolverle flujo a y. Pero si x esta bloqueado el vértice y nunca más puede mandarle flujo. Entonces, si $\overrightarrow{yx}$ nunca más puede recibir flujo, menos aún va a poder volver a vaciarse. Entonces una vez que removó a y no lo voy a poder agregar nuevamente a $M(x)$.

Entonces, la complejidad final:

$$\text{Complejidad} = P + V + S + Q = O(n^2) + O(m) + O(n^2) + O(m) = O(n^2)\square$$

5) **Probar que la distancia en networks auxiliares sucesivos aumenta.**

Teorema 2 *(Dinitz) La cantidad de NA usados es* $O(n)$

Prueba:

Sea NA un network auxiliar y NA' el siguiente network auxiliar. Sean $d_i(x) = d_f(s, x)$ las distancias de FF usadas para construir NA y d' para NA'. Vamos a demostrar que $d(t) < d'(t)$.

Sabemos por EK que $d \leqslant d'$, acá queremos ver solo el $<$. Entonces, si $d'(t) = \infty$, ya está.

Suponiendo, $d'(t) < \infty$, entonces existe al menos un camino aumentante (ca), entre s y t en el network original, por lo tanto existe un camino dirigido de s a t en NA'.

Sea $s = x_0, x_1, \ldots, x_n = t$ un caminio dirigido entre s y t en NA', como NA' es un network por niveles, entonces $d(x_i) = i, i = 0, \ldots, r$, pues mando flujo, ese camino no pude estar en NA, porque para pasar de NA a NA' se bloquean todos lo caminos de NA por lo tanto si ese camino estuviera en NA se hubiera bloqueado y no estaría en NA'. Sino es camino en NA entonces puede suceder:

a) Falta un vertice

b) Falta un lado

Veamos el a) entonces si falta un vertice, tomamos un x cualquiera por lo que $x_i \notin NA \Rightarrow d(t) \leqslant d(x_i)$.

Pero por EK, sabemos que $d \leqslant d'$:

$$\begin{aligned} d(t) \leqslant d(x_i) \leqslant_{EK} d'(x_i) &= i \\ &<_{x \neq t} r = d'(t) \\ &\Rightarrow d(t) < d'(t) \end{aligned}$$

Si falta al menos un lado (parte b) tengo dos posibles casos. Sea i el primer indice tal que $\overrightarrow{x_i x_{i+1}} \notin NA$, con esto me aseguro que todos los anteriores esten.

Caso 1. $d(x_{i+1}) < i+1 \Rightarrow d(x_{i+1}) < d'(x_{i+1})$ Entonces hacemos algo similar al caso a). pero con x_{i+1}.

$$\begin{aligned} d(t) &= d(x_{i+1}) + b(x_{i+1}) \\ &\leqslant d(x_{i+1}) + b'(x_{i+1}) \\ &< i+1+b'(x_{i+1}) \\ &\text{por hip} \\ &= d'(x_{i+1}) + b'(x_{i+1}) \\ &= d'(t) \Rightarrow d(t) < d'(t) \end{aligned}$$

Caso 2. $d(x_{i+1}) \not< i+1$. Pero $d(x_{i+1}) \leqslant i+1$, entonces $d(x_{i+1}) = i+1$. Como i es el primer indice con $\overrightarrow{x_i x_{i+1}} \notin NA$, entonces $\overrightarrow{x_j x_{j+1}} \in NA \forall j < i$, como NA es por niveles nos dice que la porción $x_0 x_1 \dots x_j$ esta en NA. $\Rightarrow d(x_j) = j, \forall j < i$. Es decir, que vale $\forall j \leqslant i+1$.

En particular $d(x_i) = i, d(x_{i+1}) = i+1$. Entonces, x_i esta en el nivel i y x_{i+1} en el nivel $i+1$. Entonces, $\overrightarrow{x_{i+1} x_i} \notin NA$ por lo que vimos antes (los niveles no son legales).

Como sabemos que $\overrightarrow{x_{i+1} x_i} \notin NA$ entonces tenemos que no está ninguno de los dos, es decir, que $\overrightarrow{x_i x_{i+1}} \notin NA$ y $\overrightarrow{x_{i+1} x_i} \notin NA$.

Pero x_i pertenece al nivel i y x_{i+1} al nivel $i+1$, entonces $\overrightarrow{x_i x_{i+1}}$ podría estar, pero no puede mandar ni devolver flujo. Como no está, caso fordward saturado o caso backward vacio. Pero $\overrightarrow{x_i x_{i+1}} \in NA'$, entonces se tiene que haber des-saturado o llenado un poco según el caso al ir de NA a NA'.

Pero entonces, para pasar de NA a NA', tengo que haber usado el lado al revés, pero no lo puedo haber usado porque $x_i x_{i+1}$ no existe.

□

6) **Probar que el valor de todo flujo es menor o igual que la capacidad de todo corte y que si f es un flujo, entonces las siguientes afirmaciones son equivalentes:**

   i) f **es maximal**

   ii) Existe un corte S **tal que** $v(f) = cap(S)$ **(y en este caso,** S **es minimal)**

   iii) No existen $f-$**caminos aumentantes. (puede usar sin necesidad de probarlo que si** f **es flujo y** S **es corte entonces** $v(f) = f(S, \overline{S}) - f(\overline{S}, S)$**)**

Prueba 3) $\Rightarrow$ 2) $\Rightarrow$ 1) $\Rightarrow$ 3).

2) $\Rightarrow$ 1) (si existe un corte S tal que $v(f) = cap(S)$ entonces f es maximal)

Dados S corte, f y g dos flujos, entonces sabemos que el valor de todo flujo es menor o igual a la capacidad de todo corte. Entonces $v(g) \leqslant cap(S)$.

Por hipotesis tenesmos que $v(f) = cap(S)$ entonces $v(g) \leqslant cap(S) = v(f)$, por lo que $v(g) \leqslant v(f)$ entonces f es maximal. Además, si T es un corte, $cap(T) \geqslant v(f) = cap(S)$, entonces S es minimal.

1) $\Rightarrow$ 3) (si f es maximal entonces no existen $f-$caminos aumentantes)

$\neg\, 3) \Rightarrow \neg\, 1)$ (contrareciproca)

Entonces, existe un $f-$camino aumentante entre s y t, entonces usando ese camino aumentante podemos construir un nuevo flujo con valor f^* tal que $v(f^*) = v(f) + \epsilon$ esto implica que f no es maximal, debido a que $v(f^*) > v(f)$.

3) $\Rightarrow$ 2), la idea es $\nexists \Rightarrow \exists$ (parte dificil)

Definamos:

$$S = \{s\} \cup \{x : \exists \text{ un } f \text{ ca entre } s \text{ y } x\}$$

Como f es flujo y S es corte, entonces:

$$v(f) = f(S, \overline{S}) - f(\overline{S}, S)$$

Entonces, veamos primero

$$f(S, \overline{S}) = \sum_{x \in S, y \in \overline{S}, xy \in E} f(\overrightarrow{xy})$$

Y tomemos un par (x, y) cualquiera con $x \in S, y \notin S, xy \in E$, entonces existe un $f-$camino aumentante entre $s \dots x$ y como $y \notin S$ no existe $f-$ca entre $s \dots y$.

Supongamos que $f(\overrightarrow{xy}) < c(\overrightarrow{xy})$, no se satura, entonces el lado $\overrightarrow{xy}$ puede usarse en algún ca. Como $s \dots x$ es $f-$ca, entonces $s \dots xy$ es $f-$ca, entonces $y \in S$, lo que es un absurdo.

$f(\overrightarrow{xy}) \not< c(\overrightarrow{xy}) \Rightarrow f(\overrightarrow{xy}) = c(\overrightarrow{xy})$ esto vale $\forall x \in S, y \notin S, \overrightarrow{xy} \in E$. Por lo tanto:

$$f(S, \overline{S}) = \sum_{x \in S, y \notin S, xy \in E} f(\overrightarrow{xy}) = \sum_{x \in S, y \notin S} c(\overrightarrow{xy}) = c(S, \overline{S}) = cap(\overline{S})$$

Por otro lado, si es un lado backward:

$$f(\overline{S}, S) = \sum_{x \notin S, y \in S, xy \in E} f(\overrightarrow{xy})$$

tomemos un par (x, y) cualquiera con $\overrightarrow{xy} \in E, x \notin S, y \in S$, entonces $\exists f-$ca, $s \dots y$

Supongamos que $f(\overrightarrow{xy}) > 0$, entonces podemos devolver flujo por $\overrightarrow{xy}$ por lo que:

$$\underbrace{s \dots y}_{ca} \overleftarrow{x}$$

también es ca, entonces $x \in S$ lo que es un absurdo, entonces $f(\overrightarrow{xy}) = 0, \forall x \notin S, y \in S, \overrightarrow{xy} \in E$

Entonces:

$$f(\overline{S}, S) = \sum_{x \notin S, y \in S, xy \in E} f(\overrightarrow{xy}) = 0$$

Por lo tanto:

$$v(f) = f(S, \overline{S}) - f(\overline{S}, S) = cap(S) - 0 = cap(S)$$

$\square$

7) **Probar que 2−COLOR es polinomial.**

Asumimos G un grafo conexo. Sea $x \in V$ corramos $BFS(x)$, si al correr $BFS(x)$, cuando un vertice z agrega a un w agregrando al lado zw, tenemos un arbol.

Sea $N(z) =$ Longitud del único camino entre x y z en el arbol BFS.

Definimos:

$$C(z) = N(z) \mod 2 = \begin{cases} 0 \rightarrow N(z)\ par \\ 1 \rightarrow N(z)\ impar \end{cases}$$

Si es propio $\Rightarrow$ **return** $\chi(G) \leqslant 2$.

Si no propio $\Rightarrow$ **return** $\chi(G) > 2$.

Fin del algoritmo.

Entonces, hay que demostrar 1) el algoritmo es polinomial, 2) el algoritmo es correcto.

Veamos 1) sabemos que $BFS(x)$ es $O(m)$ y chequear que el coloreo es propio es $O(m)$ entonces $O(m) + O(m) = O(m)$.

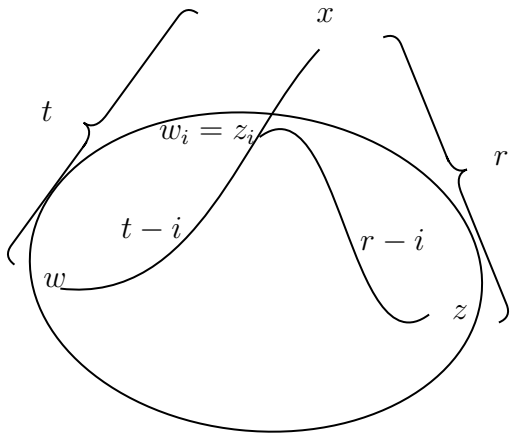
2) Supongamos que la respuesta es "no es 2−colorable", es decir, que ningún otro coloreo con 2 colores es propio lo que veremos es que G tiene un ciclo impar.

Entonces con C no propio. $\exists z, w : C(z) = C(w) \wedge zw \in E$.

Sea $x = z_0, z_1, \ldots, z_r = z$ un camino de x a z en el arbol BFS y $x = w_0, w_1, \ldots, w_t = w$ otro camino de x a w en el arbol BFS. Y $w_0 = z_0$ y $z_r \neq w_t$.

Además $\exists i : w_i = z_i \wedge w_{i+1} \neq z_{i+1}$

Consideremos $w_i w_{i+1}, \ldots\ wz\, z_{r-1} \ldots z_i$, como se muestra en la siguiente figura.



Como $w_i = z_i$ es un ciclo con

$$t - i + r - i + \underbrace{1}_{zx} = t + r - 2i + 1 \text{ lados}$$

Pero $2i$ es par y $t+r$ también es par pues $C(z) = r \mod 2$ y $C(w) = t \mod 2$ y como $C(z) = C(w)$ entonces $r \mod 2 = t \mod 2$. Entonces la suma de los lados es impar. $\square$

8) **Enunciar y probar el Teorema de Hall.**

Teorema 3 *(Hall) Si $G = (\bar{X} \cup \bar{Y}, E)$ es bipartito con partes $\bar{X}$ e $\bar{Y}$, entonces existe matching completo de $\bar{X}$ en $\bar{Y}$ si y solo si $|S| \leqslant |\Gamma(S)|$ $\forall S \subseteq \bar{X}$.*

Prueba:

($\Rightarrow$) Si existe un matching completo de $\bar{X}$ en $\bar{Y}$, el matching induce una función inyectiva de $\bar{X} \cap \bar{Y}$ tal que $\psi(x) \in E$, por lo tanto:

$$\psi(S) \subseteq \Gamma(S)$$

por lo tanto, $|\Gamma(S)| \geqslant |\psi(S)| = |S|$

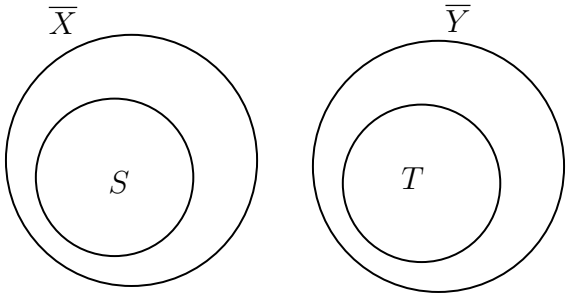
($\Leftarrow$) Queremos ver que si $\underbrace{|S| \leqslant |\Gamma(S)|}_{P} \Rightarrow \underbrace{\exists \text{ un matching completo de } \bar{X} \cap \bar{Y}}_{Q}$.

Lo veremos de la siguiente forma $[P \Rightarrow Q] = [\neg Q \Rightarrow \neg P]$ por contrareciproca.

Si $\nexists$ matching completo de $\bar{X}$ en $\bar{Y}$, entonces al correr el algoritmo llegamos a un matching maximal que no cubre a $\bar{X}$. Que es equivalente a hallar un flujo maximal entero f cuyo valor no es $|\bar{X}|$.

Al hallar f, también hallamos un corte minimal que vamos a denotar por c (seria la última cola, al correr E-K).

Sea $S = c \cap \bar{X}$, $T = c \cap \bar{Y}$, como en la siguiente imagen ($T \neq t$).



Entonces, T forma parte de c por lo tanto forma parte de la última cola. Entonces todos sus elementos fueron agregados por alguien (pues $s \notin T$), ese alguien debe ser vecino, y como el grafo es bipartito y $T \subseteq \bar{Y}$, esos vecinos deben estar en $\bar{X}$.

Pero además deben haber estado en la cola, es decir, que están en c. Entonces el vecino estaba en S. En resumen:

$\forall y \in T, \exists x \in S : xy \in E \Rightarrow$ si xy es lado $\Rightarrow$ y es vecino x con $x \in S$. Entonces, $y \in \Gamma(x) \subseteq \Gamma(S) \Rightarrow$

$$T \subseteq \Gamma(S) \tag{5}$$

Pero además:

$$\Gamma(S) \subseteq T \tag{6}$$

Ahora probemos 6, para ver si se cumple la igualdad. Sea $y \in \Gamma(S) \Rightarrow \exists x \in S : xy \in E$ (x esta en la cola).

Supongamos que si $f(\overrightarrow{xy}) = 0$ entonces x puede agregar a y a la cola entonces $y \in T$.

Supongamos que si $f(\overrightarrow{xy}) = 1$ entonces x no puede agregar a y a la cola pero $x \in S$ entonces algún vertice z agrego a x a la cola.

Como $f(\overrightarrow{xy}) = 1$ entonces $out_f(x) = 1 \Rightarrow In_f(x) = 1 \Rightarrow f(\overrightarrow{sx}) = 1$. Entonces, s no agrego a x a la cola por lo que $z \neq s$.

Como z debe ser un vecino de x y $z \neq s$ entonces $z \in \bar{Y}$. Y para agregar a x, lo debe haber hecho backward, entonces $f(\overrightarrow{xz}) = 1$ sino no puede. Entonces: $f(\overrightarrow{xy}) = 1$ y $f(\overrightarrow{xz}) = 1$, por lo que $y = z$.

Pero si $y = z$, agrego a x a la cola, y esta en c, entonces $y \in c \cap \bar{Y} = T$, con esto 6 queda demostrado.

Además, 5 y 6 $\Rightarrow$

$$T = \Gamma(S) \tag{7}$$

Sea $S_0 = \left\{x \in \bar{X} : In_f(x) = 0\right\}$, entonces s agrega a los vertices de S_0 a la cola, entonces:

$$S_0 \subseteq S \tag{8}$$

Como estamos suponiendo que $v(f) \neq |\bar{X}|$, entonces:

$$S_0 \neq \emptyset \tag{9}$$

(me queda al menos un vertice sin matchear).

Queremos comparar $S - S_0$ con T.

$y \in T \Leftrightarrow y$ es puesto en la cola por alguien pero como $t \notin c$, y no puede poner a t. Entonces, $f(\overrightarrow{yt}) = 1 \Rightarrow out_f(y) = 1$ y $In_f(y) = 1$. Entonces $\exists x : f(\overrightarrow{xy}) = 1$.

Como $f(\overrightarrow{xy}) = 1$ entonces, y agrega a x a la cola y $x \in S$ además $out_f(x) = 1 \Rightarrow In_f(x) = 1 \Rightarrow x \notin S_0$. Entonces, podemos concluir que $x \in S - S_0$. Y hay un solo x con $f(\overrightarrow{xy}) = 1$.

Entonces, tengo una función $y \to x$ y T a $S - S_0$ inyectiva, pero además si $x \in S - S_0$ entonces $in_f(x) = 1 \Rightarrow out_f(x) = 1 \Rightarrow \exists y : f(\overrightarrow{xy}) = 1$.

Entonces $y \in \Gamma(S) = T$ entonces tengo una biyección entre

$$T \text{ y } S - S_0 \tag{10}$$

Finalmente:

$$|\Gamma(S)| \underbrace{=}_{7} |T| \underbrace{=}_{10} |S - S_0| \underbrace{=}_{8} |S| - |S_0| < |S| \tag{11}$$

La última desigualdad vale pues $S_0 \neq \emptyset$. Hemos probado que $|\Gamma(S)| < |S|$, lo que contradice la hipótesis de que $|S| \leqslant |\Gamma(S)|$. para todo $S \subseteq \bar{X}$.

9) **Enunciar y probar el teorema del matrimonio de Konig**

Teorema 4 *(matrimonio de konig) Todo grafo bipartito regular tiene un matching perfecto (todos los vertices forman parte del matching).*

Prueba: Dado un conjunto de vertices definimos:

$$E_w = \{zw \in E : w \in W\}$$

Sean $\bar{X}$ e $\bar{Y}$ las partes de G y supongamos $W \subseteq \bar{X}$ (completamente contenido en $\bar{X}$). Entonces:

$$|E_w| = |\{zw : w \in W\}|$$

Como $W \subseteq \bar{X}$ y no hay lados entre vertices de $\bar{X}$,

cada lado que aparece en E_w, aparece una sola vez

$$= \sum_{w \in W} |\Gamma(w)|$$

$$= \sum_{w \in W} \delta(w)$$

Como G es regular

$$\sum_{w \in W} \Delta = \Delta|W|$$

Si $W \subseteq \bar{Y}$ vale el mismo analisis y tambien tenemos que $|G_w| = \Delta|W|$.

Primero demostraremos que hay un matching completo (basta demostrar la condición de Hall).

Sea $S \subseteq \bar{X}$ y sea $l \in E_s$, entonces l es de la forma $l = xy$ con $x \in S$ y $y \in \bar{Y}$. Como l es lado, entonces y es vecino de x entonces $y \in \Gamma(x)$. Pero $x \in S \Rightarrow y \in \Gamma(S)$. Como $l = xy$ entonces si $y \in \Gamma(S) \Rightarrow l \in E_{\Gamma(S)}$.

Conclusión: $E_S \subseteq E_{\Gamma(S)} \Rightarrow |E_s| \leqslant |E_{\Gamma(S)}|$. Y como $S \subseteq \bar{X} \Rightarrow |E_S| = \Delta|S|$. A su vez, $\Gamma(S) \subseteq \bar{Y} \Rightarrow |E_{\Gamma(S)}| = \Delta|\Gamma(S)|$. Entonces podemos decir que $\cancel{\Delta}|S| \leqslant \cancel{\Delta}|\Gamma(S)| \Rightarrow |S| \leqslant |\Gamma(S)|$.

Entonces se satisface la condición de Hall, podemos afirmar que existe matching completo de x a y. Para ver que ese matching es perfecto basta ver que $|\bar{X}| = |\bar{Y}|$.

Como G es bipartito los unicos lados son entre x e y. Entonces:

$$E = E_{\bar{X}} + E_{\bar{Y}}$$

Por lo tanto

$$|E_{\bar{X}}| = |E_{\bar{Y}}|$$

Como G es regular

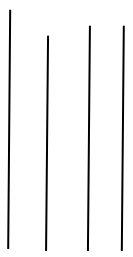
$$\Delta|\bar{X}| = \Delta|\bar{Y}|$$

$$|\bar{X}| = |\bar{Y}|$$

10) **Probar que si G es bipartito entonces $\chi'(G) = \Delta(G)$**

Corolario 2 *(tambien de koring) G bipartito entonces $\chi'(G) = \Delta(G)$.*

Donde $\chi'(G)$ es el indice cromatico (lados del un grafo), osea, la menor cantidad de colores necesarios para colorear los lados de un grafo de forma tal que lados con un vertice en común tengan colores distintos. Es obvio que $\Delta \leqslant \chi'(G)$.



Prueba: Supongamos G regular, por el teorema del matrimonio G tiene un matching perfecto, puedo colorear a todos con un color sin ningún problema, ej. puedo usar el color 1.

Luego remuevo los lados, entonces $\widetilde{G} = G -$esos lados, cada vertice disminuye su grado en 1. Entonces $\widetilde{G}$ sigue siendo regular. Por lo que tiene un matching perfecto. Coloreo esos lados con el color 2, los remuevo, etc. Y así sucesivamente.

Siempre va a ser regular hasta el final y terminamos coloreando con Δ colores.

Lemma 1 *G bipartito, entonces existe H bipartito regular tal que $G \subseteq H$. Entonces $\Delta(G) = \Delta(H)$.*

Prueba: Si G es un grafo bipartito, entonces podemos encontrar un grafo bipartito regular H tal que $G \subseteq H$. Esto se debe a que un grafo bipatito regular es un grafo en el que todos los vertices tienen el mismo número de vecinos. Para encontrar un grafo que contenga a G, podemos comenzar por elegir cualquier conjunto de vértices de G que formen un conjunto completo. Luego, podemos agregar vértices adicionales al grafo uno por uno, asegurandonos de que todos los nuevos vértices estén conectados a todos los vértices existentes. El grafo resultante será un grafo bipartito tegular que contiene a G.

Una vez que hemos encontrado un grafo bipartito regular H que contenga a G, podemos mostrar que $\Delta(G) = \Delta(H)$. Esto se debe a que todos los vértices de G tienen el mismo número de vecinos en H. Por lo tanto, el grado de cualquier vertice en G debe ser igual al grado de cualquier vertice de H. Esto significa que $\Delta(G) = \Delta(H)$.

Entonces $\chi^{'}(H) = \Delta$. $G \subseteq H \Rightarrow \chi^{'}(G) \leqslant \Delta$. Como $\Delta \leqslant \chi^{'}(G) \Rightarrow \chi^{'}(G) = \Delta$.

11) **Probar la complejidad $O(n^4)$ del algoritmo Hungaro y dar una idea de como se la puede reducir a $O(n^3)$.**

Teorema 5 *La complejidad del algoritmo Hungaro como lo vimos en clase es $O(n^4)$.*

Prueba: Se corre EK sobre la matriz y, crucialmente, no tenemos que cambiar ningún network al cambiar la matriz, y vimos que al cambiar la matriz se pueden perder algunos ceros pero ninguno del matching parcial que tenemos hasta el momento.

Como no se pierde ningún cero del matching parcial que tenemos, podemos continuar con la busqueda como veniamos, simplemente revisando las columnas donde hay nuevos ceros si lo hacemos completamente con el sistema de matriz o bien agregando esas columnas a la cola si lo hacemos con colas explícitas.

De esta forma continuaremos con varios posibles cambios de matriz en el medio, pero terminaremos agregando UN lado extra al matching.

Restar el *min* de una fila es $O(n)$, entonces restar *min* de cada fila es $O(n^2)$. Es el mismo análisis para las columnas es $O(n^2)$.

Hallar matching inicial es $O(n^2)$ pues hay que revisar n filas y para cada una de las n columnas. Este matching se puede extender a lo sumo $O(n)$ veces. Por lo tanto:

Complejidad Hungaro $= O(n^2) + O(n) *$ (Complejidad de extender matching en un lado)

Para extender el matching, hay que revisar n filas buscando ceros, entonces cada busqueda es $O(n)$, por lo tanto en el peor de los casos si revisamos todas las filas hasta poder extender el matching es $O(n^2)$.

Luego de revisar a lo sumo n filas, si o si extendemos el matching pero en el medio quizas debamos hacer cambios de matrices, si seguimos con el matching parcial que teniamos, no hace falta re-escanear las filas de S (esta dentro del cambio de matriz).

Y además por el lema sabemos que hay $\leqslant O(n)$ cambios de matries antes de extender el matching. Por lo tanto:

$$\text{Complejidad de extender el matching en un lado} = \underbrace{O(n^2)}_{\text{escanear filas}} + \underbrace{O(n)}_{\#\text{ cambio de matrices}} * CCM$$

donde CCM es la complejidad de hacer un cambio de matriz (esto pude cambiar al programarlo). Entonces, cambiar la matriz requiere:

1. Calcular $m = \min\left\{S \times \overline{\Gamma(S)}\right\}$ (esto es $O(|S \times \overline{\Gamma(S)}|) = O(n^2)$).
2. Restar m de S (filas) esto es $O(n) * |S| = O(n^2)$. Sumar m a $\Gamma(S)$ (columnas) eso es $O(n) * |\Gamma(S)| = O(n^2)$.

Entonces:

$$CCM = O(n^2) + O(n^2) + O(n^2) = O(n^2)$$

Por lo tanto:

$$\text{Complejidad hungaro} = O(n^2) + O(n) * (O(n^2) + O(n) * O(n^2)) = O(n^4)$$

Teorema 6 *El hungaro se puede coficar en $O(n^3)$.*

Prueba:

Hay que tratar de hacer que $CCM = O(n)$, entonces:

$$\text{Complejidad hungaro} = O(n^2) + O(n) * (O(n^2) + O(n) * O(n)) = O(n^3)$$

Debemos:

1. Hallar un min de $O(n^2)$ elementos en $O(n)$.
2. Debemos cambiar $O(n^2)$ elementos en $O(n)$.

Para hacer 1. parte del costo de hallar cada m se traslada a la parte donde escaneamos filas, entonces:

$$\begin{aligned} m &= \min\left\{S \times \overline{\Gamma(S)}\right\}, \text{ Sea } C \text{ la matriz de costos} \\ &= \min\{C_{x,y} : x \in S, y \notin \Gamma(S)\} \\ &= \min_{y \notin \Gamma(S)} \left(\min_{x \in S}\{C_{x,y}\}\right) \text{ (esto de puede calcular en } O(n) \text{ haciendo)} \\ &= \min_{y \notin \Gamma(S)} M_y \end{aligned}$$

donde $M_y = \min_{x \in S}\{C_{x,y}\}$, siempre hay que tenerlo precalculado. Precalcular los M_y demanda $O(n)$ pero los podemos hacer cuando escanemos una fila al buscar cero, en los no cero actualizamos M_y.

Para 2. hacemos una resta y suma virtual (indicar cuanto se le deberia restar y sumar). Entonces usamos $RF(x)$ que indique cuanto restarle a la fila x y $SC(y)$ que indique cuanto sumarle a la columna y.

Restar m de S es simplemente hacer $RF(x) -= m$, $\forall x \in S$ esto es $O(n)$.
Sumar m a $\Gamma(S)$ es simplemente hacer $SC(y) += m$, $\forall y \in \Gamma(S)$ esto es $O(n)$.

El problema es el chequeo de ceros. Entonces, en vez de hacer

```
if (0 == C(x)(y))
```

se chequea haciendo:

```
if (0 == C(x)(y) - RF(x) + SC(y))
```

que es $O(1)$.

12) **Enunciar el teorema de la cota de Hamming y probarlo**

Teorema 7 *(cota de hamming) Sea C código de longitud n, $t = \lfloor \frac{\delta-1}{2} \rfloor$, entonces:*

$$|C| \leqslant \frac{2^n}{1 + n + \binom{n}{2} + \cdots + \binom{n}{t}}$$

Prueba:

Sea $A = \bigcup_{x \in C} D_t(x)$. Como C corrige t errores, $D_t(x) \cap D_t(y) = \emptyset$, $\forall x, y \in C, x \neq y$. Entonces esa unión es disjunta por lo que $|A| = \sum_{x \in C} |D_t(x)|$.

Queremos calcular $|D_t(x)|$, para ello definimos $S_r(x) = \{y \in \{0,1\}^n : d_H(x,y) = r\}$ (parto al disco en circulos de radio r).

Entonces, $D_t(x) = \bigcup_{r=0}^{t} S_r(x)$ y la unión es disjunta. Entonces

$$|D_t(x)| = \sum_{r=0}^{t} |S_r(x)|$$

$y \in S_r(x) \iff y$ difiere de x en exactamente r lugares.

$y \in S_r(x) \to r$ lugares (r posiciones del conjunto $\{1, 2, \ldots, n\}$), es un biyección.

Cada $y \in S_r(x)$ determina r lugares y el conjunto de r lugares determina un $y \in \delta(x)$. Entonces:

$$|S_r(x)| = |\{L \subseteq \{1, \ldots, n\} : |L| = r\}| = \binom{n}{r}$$

(es cantidad de conjuntos de subconjuntos). Por lo tanto:

$$\begin{aligned} |A| &= \sum_{x \in C} |D_t(x)| = \sum_{x \in C} \sum_{r=0}^{t} |S_r(x)| \\ &= \sum_{x \in C} \sum_{r=0}^{t} \binom{n}{r} \\ &= \left(\sum_{r=0}^{t} \binom{n}{r} \right) \cdot |C| \end{aligned}$$

Despejando C:

$$|C| = \frac{|A|}{\sum_{r=0}^{t} \binom{n}{r}} \leqslant \frac{2^n}{\sum_{r=0}^{t} \binom{n}{r}}$$

la desiguandad sale de que $A = \bigcup_{x \in C} D_t(x) \subset \{0,1\}^n$. $\square$

13) **Probar que si H es matriz de chequeo de C, entonces**

$$\delta(C) = \min\{j : \exists \textbf{ un conjunto de } j \textbf{ columnas LD de } H\}$$

(LD es linealmente dependiente)

Un código es lineal si es un subespacio vectorial de $\{0,1\}^n$.

Definición 1 *El peso de Hamming es $|x| = d_H(x,0) = \#$ de 1's de x.*

Propiedad C lineal $\Rightarrow \delta(C) = \min\{|v| : v \in C, v \neq 0\}$.

Prueba:

Teorema 8 *Si H es matriz de chequeo de C, entonces:*

$$\begin{aligned}\delta(C) &= \min\ \{\textit{número de columnas de } H \textit{ que son LD}\} \\ &= \min\{j : \exists j \textit{ columnas LD de } H\}\end{aligned}$$

Prueba:

Sea $m = \min\{j : \exists j \text{ columnas LD de } H\}$. Probaremos $\delta(C) \leqslant m$ y luego $\delta(C) \geqslant m$. Denotaremos la $j-$ésima columna de H por $H^{(j)}$. Por definición de m existe $j_1, \ldots j_m$ tal que $H^{(j_1)}, \ldots, H^{(j_m)}$ son LD.

Entonces existen $c_1, \ldots, c_m$ no todos 0 tales que:

$$c_1 H^{(j_1)} + \cdots + c_m H^{(j_m)} = 0$$

Sea $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ con 1 en la posición i. Entonces:

$$He_i^t = \begin{bmatrix} & i & \\ & \vdots & \\ & i & \\ & \vdots & \\ & i & \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = H^{(i)}$$

Es la columna $i-$ésima de H.

Sea $x = c_1 e_{j_1} + c_2 e_{j_2} + \ldots + c_m e_{j_m}$, como no todos los c_j son 0, entonces $x \neq 0$, por lo que:

$$\begin{aligned}Hx^t &= H(c_1 e_{j_1}^t + c_2 e_{j_2}^t + \ldots + c_m e_{j_m}^t) \\ &= c_1 He_{j_1}^t + c_2 He_{j_2}^t + \ldots + c_m He_{j_m}^t \\ &= c_1 H^{(j_1)} + c_2 H^{(j_2)} + \ldots + c_m H^{(j_m)} = 0 \\ &\Rightarrow Hx^t = 0\end{aligned}$$

$\Rightarrow x \in C$ pues $C = Nu(H)$, pero su peso es $\leqslant m$, pues es suma de a lo sumo m vectores de peso 1. y vimos que $x \neq 0$ entonces sabemos que:

$$\delta(C) = \min\{|v| : v \in C, v \neq 0\}$$

Por lo tanto como $x \in C$ y $x \neq 0$, entonces x está en ese conjunto, así que tenemos:

$$\delta(C) = \min\{|v| : v \in C, v \neq 0\} \leqslant |x|$$

Pero su peso es $\leqslant m$, entonces:

$$\delta(C) = \min\{|v| : v \in C, v \neq 0\} \leqslant |x| \leqslant m$$

Entonces, $\delta(C) \leqslant m$.

La otra parte de la desigualdad. Sea $x \neq 0 : \delta(C) = |x|$, entonces:

$$\begin{aligned} x &= c_1 e_{i_1} + \ldots + c_{\delta(C)} e_{i_{\delta(C)}} \\ &= \{\text{como } x \in C, \text{ entonces } x \neq 0\} \\ 0 = Hx^t &= H^{(i_1)} + \ldots + H^{(i_{\delta(C)})} \\ &\Rightarrow \left\{H^j, \ldots, H^{(i_{\delta(C)})}\right\} \text{ son LD} \Rightarrow m \leqslant \delta(C) \end{aligned}$$

□

14) **Sea C un código cíclico de dimensión k y longitud n y sea $g(x)$ su polinomio generador. Probar que:**

i) C **esta formado por los multiplos de** $g(x)$ **de grado menor que** n**.**

$$C = \{p(x) : gr(p) < n \ \& \ g(x)|p(x)\}$$

ii) $C = \{v(x) \odot g(x) : \ v$ **es un polinomio cualquiera**$\}$

iii) $gr(g(x)) = n - k$

iv) $g(x)$ **divide a** $1 + x^n$

Prueba:

Sea $C_1 = \{p(x) \in \mathbb{Z}_2(x) : gr(p) < n \wedge g(x)|p(x)\}$ y $C_2 = \{v(x) \odot g(x) : v \in \mathbb{Z}_2(x)\}$. Hay que demostrar que $C \subseteq C_1, C_2 \subseteq C$ y $C_1 \subseteq C_2$.

Tenemos que ver $C_1 \subseteq C_2$, entonces $p(x) \in C_1$ y veremos si esta en C_2.

Entonces $gr(p) < n \wedge g(x)|p(x)$, entonces existe $q(x) : p(x) = g(x)q(x)$ tomando modulo nos queda:

$$\begin{aligned} p(x) \mod (1 + x^n) &= g(x)q(x) \mod (1 + x^n) \\ &= g(x) \odot q(x) \in C_2 \end{aligned}$$

Como $gr(p) < n \Rightarrow p(x) \mod (1 + x^n) = p(x) \Rightarrow p(x) \in C_2$.

Ahora veamos que $C_2 \subset C$, puesto que $g(x) \in C_2$ y debido a esto dada cualquier palabra $v(x)$ en C, $v(x) \odot g(x) \in C$

$C \subseteq C_1$, Sea $p(x) \in C$ como las palabras de C tienen longitud n, entonces $gr(p) < n$ (A).

Dividamos $p(x)$ por $g(x)$ entonces existe $q(x)$ y $r(x)$ tal que:

$$p(x) = g(x)q(x) + r(x),\ gr(r) < gr(g)$$

Tomando modulo y usando $gr(p) < n$

$$\begin{aligned} p(x) \mod (1+x^n) &= g(x)q(x) + r(x) \mod (1+x^n) \\ &= g(x)q(x) \mod (1+x^n) + \underbrace{r(x) \mod (1+x^n)}_{gr(r)<gr(g)<n} \end{aligned}$$

$$p(x) = g(x) \odot q(x) + r(x)$$

Por lo tanto:

$$r(x) = \underbrace{\underbrace{p(x)}_{\in C} + \underbrace{g(x) \odot q(x)}_{\in C_2}}_{\in C \text{ porque } C \text{ es lineal}}$$

Pero $gr(r) < gr(g) =$ menor grado de un polinomio no nulo en C. Entonces $r(x) = 0 \Rightarrow p(x) = g(x)q(x) \Rightarrow g(x)|p(x)$ (B).

Entonces por (A) y (B) $\Rightarrow C \subseteq C_1$

Prueba de 3): $C = C_1$ dice que:

$$\begin{aligned} C &= \{q(x)g(x) : gr(qg) < n\} \\ &= \{q(x)g(x) : gr(q) + gr(g) < n\} \\ &= \{q(x)g(x) : gr(q) < n - gr(g)\} \end{aligned}$$

Limito el grado de q. Entonces:

$$\begin{aligned} |C| &= |\,\{q(x)g(x) : gr(q) < n - gr(g)\}\,| \\ &= \{\text{Como conjunto no son iguales, pero tienen la misma \# de elementos}\} \\ &= |\,\{q(x) : gr(q) < n - gr(g)\}\,| \\ &= 2^{n-gr(g)} \end{aligned}$$

Pero $|C| = 2^k \Rightarrow k = n - gr(g)$. Entonces $gr(g) = n - k$.

4): Dividamos $1 + x^n$ por $g(x)$, entonces:

$$\begin{aligned} 1 + x^n &= g(x)q(x) + r(x),\ gr(r) < gr(g) \\ &= \{\text{Despejando } r(x)\} \\ r(x) &= (1 + x^n) + g(x)q(x) \\ &= \{\text{Como } gr(r) < gr(g) < n \text{ entonces } r(x) = r(x) \mod (1+x^n)\} \\ r(x) &= ((1 + x^n) + g(x)q(x)) \mod (1 + x^n) \\ &= (1 + x^n) \mod (1 + x^n) + (g(x)q(x)) \mod (1 + x^n) \\ &= 0 + q(x) \odot g(x) \\ &= q(x) \odot g(x) \in C \end{aligned}$$

Como $r(x) \in C$ y $gr(r) < gr(g)$, entonces $r(x) = 0 \Rightarrow g(x)|(1 + x^n)$.

15) **Probar que 3SAT es NP-completo**

Teorema 9 *(Karp, 1978) 3−SAT es NP−COMPLETO.*

Nosotros vamos a demostrar que 3−Color es $NP-$COMPLETO, para ello usaremos $SAT \to 3-SAT \to 3-Color$, para el salto de 3−SAT a 3−Color usaremos un grafo bastante complejo que se puede colorear con 3 colores. Cabe resaltar que el salto de 3 a 2 es $NP-$COMPLETO por esto.

Prueba del teorema:

Veremos que $SAT \leqslant_p 3-SAT$.

Entonces, sea B en forma conjuntiva normal (CNF) tal que $B = D_1 \wedge \ldots \wedge D_r$ con $D_j = l_{1,j} \vee l_{2,j} \vee \ldots \vee l_{i,j}$ con $l_{r_j,j}$ literales.

Queremos construir $\widetilde{B}$ polinomialmente tal que $\widetilde{B}$ este en CNF con 3 literales por disjunción tal que B sea satisfacible si y solo si $\widetilde{B}$ es satisfacible.

Voy a definir unos E_j y $\widetilde{B} = E_1 \wedge \ldots \wedge E_n$.

Cada E_j lo definimos a partir de D_j, y procedemos de diferente manera según la cantidad de literales r_j presentes en D_j, entonces:

Si $r_j = 3$ tenemos que $E_j = D_j$, por lo que no hago nada.

Si $r_j < 3$ agregamos variables mudas que no afectan el resultado, vemos los casos:

- $r_j = 2 \to D_j = l_{1,j} \vee l_{2,j}$
  $E_j = (l_{1,j} \vee l_{2,j} \vee y_j) \wedge (l_{1,j} \vee l_{2,j} \vee \bar{y}_j)$
- $r = 1 \to D_j = l_{1,j}$
  $E_j = (l_{1,j} \vee y_{1,j} \vee y_{2.j}) \wedge (l_{1,j} \vee \bar{y}_{1,j} \vee y_{2,j}) \wedge (l_{1,j} \vee y_{1,j} \vee \bar{y}_{2,j}) \wedge (l_{1,j} \vee \bar{y}_{1,j} \vee \bar{y}_{2,j})$

Si $r_j > 3$ es el problema más grande:

$$D_j = l_{1,j} \vee l_{2,j} \vee \ldots \vee l_{r_j,j}$$

entonces:

$$\begin{aligned} E_j =& (l_{1,j} \vee l_{2,j} \vee y_{1,j}) \wedge (l_{3,j} \vee y_{2,j} \vee \bar{y}_{1,j}) \\ & \wedge (l_{4,j} \vee y_{3,j} \vee \bar{y}_{2,j}) \wedge \ldots \wedge (l_{r_{j-2},j} \vee y_{r_{j-3},j} \vee \bar{y}_{r_{j-4},j}) \\ & \wedge (l_{r_{j-1},j} \vee l_{r_j,j} \vee \bar{y}_{r_{j-3},j}) \end{aligned}$$

Son variables booleanas para evaluarlas tengo que elegir un vector de $0's$ y $1's$.

Ejemplo: Supongamos $\widetilde{B}$ satisfacible, $\widetilde{B}$ es suma de función de variables $x_1, \ldots x_{algo}$ y variables extra $y_{i,j}$. Podriamos denotarlo con $\widetilde{B}(\overrightarrow{x}, \overrightarrow{y})$ mientras que B depende solo de las $\overrightarrow{x}$, es decir, $B(\overrightarrow{x})$.

Es decir:

$$\underbrace{B = D_1 \wedge \ldots \wedge D_n}_{\overrightarrow{x}} \to \underbrace{\widetilde{B} = E_1 \wedge \ldots \wedge E_n}_{\overrightarrow{x}, \overrightarrow{y}}$$

Vamos a demostrar que si $\exists \overrightarrow{a} : B(\overrightarrow{a}) = 1 \Leftrightarrow \exists \overrightarrow{c}, \overrightarrow{b} : \widetilde{B}(\overrightarrow{c}, \overrightarrow{b}) = 1$. (la idea es ver que $\overrightarrow{a} = \overrightarrow{c}$)

($\Leftarrow$) Supongamos que $\widetilde{B}(\overrightarrow{c}, \overrightarrow{b}) = 1$ queremos probar $B(\overrightarrow{a}) = 1$. Supongamos que no se cumple y llegaremos a un absurdo, entonces se da $B(\overrightarrow{a}) = 0$.

Como $B = D_1 \wedge \ldots \wedge D_n$ entonces $\exists j : D_j(\overrightarrow{a}) = 0$, pero $D_j = l_{1,j} \vee l_{2,j} \vee \ldots \vee l_{r_j,j}$ donde todos los terminos tienen que ser cero.

Entonces, $l_{i,j}(\overrightarrow{a}) = 0 \ \forall i$ (y ese j). Por otro lado $\widetilde{B}(\overrightarrow{a}, \overrightarrow{b}) = 1 \Rightarrow \exists E_j(\overrightarrow{a}, \overrightarrow{b}) = 1 \ \forall j$ en particular para ese j que mencionamos antes.

Ahora tenemos que ver los casos:

- Si $r_j = 3$ tenemos que $E_j = D_j$ asi que esto es imposible. $D_j(\overrightarrow{a}) = 0 \Rightarrow E_j = 0 \Rightarrow \widetilde{B}(\overrightarrow{a}, \overrightarrow{b}) = 0$ Absurdo.
- Si $r_j = 2$ tenemos que $E_j = (l_{1,j} \vee l_{2,j} \vee y_j) \wedge (l_{1,j} \vee l_{2,j} \vee \bar{y}_j)$ pero $l_{i,j}(\overrightarrow{a}) = 0$ entonces:

$$\begin{aligned} 1 &= E_j(\overrightarrow{a}, \overrightarrow{b}) \\ &= (0 \vee 0 \vee y_j(\overrightarrow{b})) \wedge (0 \vee 0 \vee \bar{y}_j(\overrightarrow{b})) \\ &= y_j(\overrightarrow{b}) \wedge \bar{y}_j(\overrightarrow{b}) \\ 1 &= 0 \end{aligned}$$

  Absurdo.

- $r_j = 1$

$$\begin{aligned} 1 &= E_j(\overrightarrow{a}, \overrightarrow{b}) \\ &= (l_{1,j} \vee y_{1,j} \vee y_{2,j}) \wedge (l_{1,j} \vee y_{1,j} \vee \bar{y}_{2,j}) \wedge (l_{1,j} \vee \bar{y}_{1,j} \vee y_{2,j}) \wedge \\ &(l_{1,j} \vee \bar{y}_{1,j} \vee \bar{y}_{2,j})[\overrightarrow{a}, \overrightarrow{b}] \\ &= (p \vee q) \wedge \underbrace{(\bar{p} \vee q) \wedge (p \vee \bar{q})}_{sii} \wedge (\bar{p} \vee \bar{q}) \\ &= 0 \end{aligned}$$

  Absurdo.

- $r_j > 4$

$$\begin{aligned} 1 &= E_j(\overrightarrow{a}, \overrightarrow{b}) \\ &= (l_{1,j} \vee l_{2,j} \vee y_{1,j}) \wedge (l_{3,j} \vee \bar{y}_{1,j} \vee y_{2,j}) \wedge (l_{4,j} \vee \bar{y}_{2,j} \vee y_{3,j}) \wedge \ldots \\ &\wedge (l_{r_j-2,j} \vee \bar{y}_{r_j-4,j} \vee y_{r_j-3,j}) \wedge (\bar{y}_{r_j-3,j} \vee l_{r_j-1,j} \vee l_{r_j,j})[\overrightarrow{a}, \overrightarrow{b}] \\ &\text{sabemos que: } l_{i,j}(\overrightarrow{a}) = 0 \\ &\text{si } p_i = y_{i,j}(\overrightarrow{b}) = p_1 \wedge (\bar{p}_1 \vee p_2) \wedge (\bar{p}_2 \vee p_3) \wedge \ldots \wedge (\bar{p}_{r_j-4} \vee p_{r_j-3}) \wedge \bar{p}_{r_j-3} \\ &p_1 \wedge (p_1 \Rightarrow p_2) \wedge (p_2 \Rightarrow p_3) \wedge \ldots \wedge (p_{r_j-4} \Rightarrow p_{r_j-3}) \wedge \bar{p}_{r_j-3} = 0 \end{aligned}$$

  Todos tienen que ser 1 y el último 0, es un absurdo.

($\Rightarrow$) Si $\exists \overrightarrow{a} : B(\overrightarrow{a}) = 1 \Rightarrow \exists \overrightarrow{b} : \widetilde{B}(\overrightarrow{a}, \overrightarrow{b}) = 1$

Para $r_j \leqslant 3$ se le puede dar cualquier valor a los $y_{i,j}$ por ejemplo 0 y es trivial ver los casos $r_j = 1$ y $r_j = 2$

El único problema grande es $r_j > 3$

Como $B(\overrightarrow{a}) = 1$ entonces $D_j(\overrightarrow{a}) = 1 \, \forall j$ (y al menos un término de D_j es 1). Entonces, $\exists i_j : l_{i_j,j}(\overrightarrow{a}) = 1$. Si hay más de uno tomo cualquiera, por ejemplo el primero.

Evaluamos los $y_{i,j}$ de forma tal que:

$$y_{i,j}(\overrightarrow{b}) = 1 \; si, \; i = 1, \ldots i_j - 2$$
$$y_{i,j}(\overrightarrow{b}) = 0 \; si, \; i \geqslant i_j - 1$$

Entonces, si evaluamos tenemos:

$$\begin{aligned} E_j(\overrightarrow{a}, \overrightarrow{b}) = (l_{1,j} \vee l_{2,j} \vee \underbrace{y_{1,j}}_{=1}) \wedge \\ (l_{3,j} \vee \bar{y}_{1,j} \vee \underbrace{y_{2,j}}_{=1}) \wedge \ldots \wedge \\ (l_{i_j-1,j} \vee \bar{y}_{i_j-3,j} \vee \underbrace{y_{i_j-2,j}}_{=1}) \wedge \\ (\underbrace{l_{i_j,j}}_{=1(t)} \vee \bar{y}_{i_j-2,j} \vee y_{i_j-1,j}) \wedge \\ (l_{i_j+1,j} \vee \underbrace{\bar{y}_{i_j-1}, j}_{=1} \vee y_{i_j,j}) \wedge \ldots \\ = 1 \end{aligned}$$

□

16) **Probar que** $3-$**COLOR es NP-completo**

Teorema 10 *(Karp)* $3-Color \; es \; NP-COMPLETO.$

Prueba:

Veremos que $3-\text{SAT} \leq_p 3-\text{Color}$. Es decir, dada una expresión booleana B en CNF con 3 literales por disjunción, debemos construir polinomialmente un grafo G tal que se cumpla que:

$$B \text{ es satisfacible} \Leftrightarrow \chi(G) \leq 3$$

Suponemos $B = D_1 \wedge \ldots \wedge D_m$ con variables $x_1, \ldots x_n$ y $D_j = l_{1,j} \vee l_{2,j} \vee l_{3,j}$. (por ser $3-$SAT, esto queda fijo).

Construcción polinomialmente del grafo G:

Vertices:

$$\{s, t\} \cup \{v_l : l \text{ es literal}\} \cup \{a_{i,j}, e_{i,j}\}_{i=1,2,3; j=1,2,\ldots m}$$

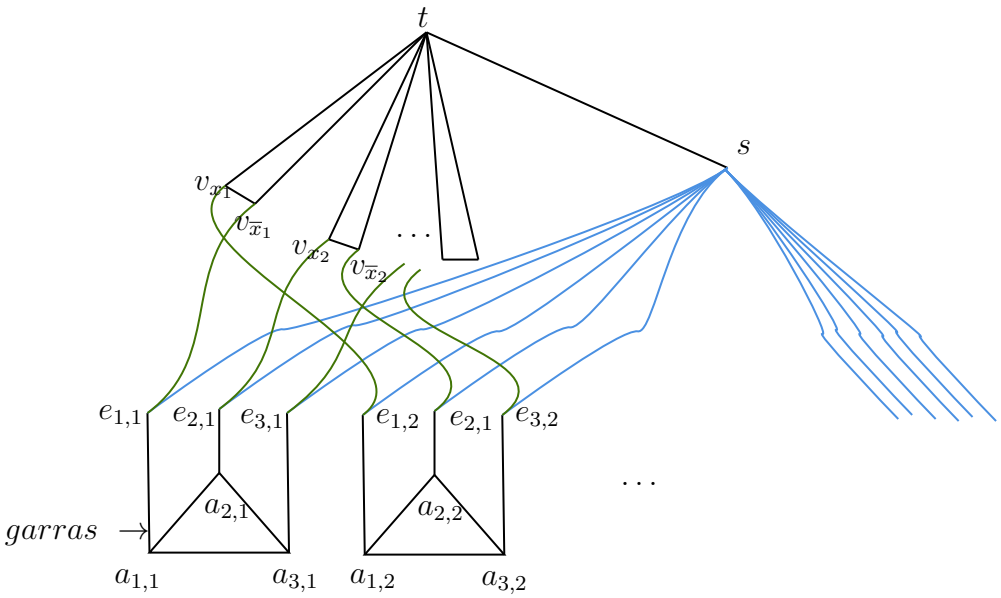
donde $\{v_l : l \text{ es literal}\}$ es equivalente a $\{v_{x_1}, v_{x_2}, \ldots, v_{x_n}, v_{\bar{x}_1}, v_{\bar{x}_2}, \ldots, v_{\bar{x}_n}\}$

Lados:

- st
- tv_l para todo literal l

- $v_{x_i}v_{\bar{x}_i}$ $\forall i = 1 \ldots, n$
- $a_{1,j}a_{2,j}$
- $a_{2,j}a_{3,j}$
- $a_{1,j}a_{3,j}$ estos últimos para $j = 1, 2, \ldots, m$ y forman un triangulo los tres.
- $a_{i,j}e_{i,j}$ para $i = 1, 2, 3$ y $j = 1, 2, \ldots, m$
- $se_{i,j}$ para $i = 1, 2, 3$ y $j = 1, 2, \ldots, m$
- Usando que es 3−SAT, es decir, $D_j = l_{1,j} \vee l_{2,j} \vee l_{3,j}$ entonces tenemos $e_{i,j}v_{l_i,j}$ para $i = 1, 2, 3$ y $j = 1, 2, \ldots, m$. Ejemplo: si $D_j = (x \vee \bar{x}_2 \vee x_4)$ entonces tengo $\overrightarrow{v_x e_{1,j}}, \overrightarrow{v_{\bar{x}_2} e_{2,j}}, \overrightarrow{v_{x_4} e_{3,j}}$

El grafo es como en la siguiente imagen:



Observemos que G tiene un triangulo, entonces $\chi(G) \geqslant 3$, por lo que

$$\chi(G) \leqslant 3 \Leftrightarrow \chi(G) = 3$$

Probemos primero:

$$\chi(G) = 3 \Rightarrow B \text{ es satisfacible}$$

Como $\chi(G) = 3$ entonces existe un coloreo propio de G con 3 colores, llamemosle C.

Necesitamos definir un vector $(b_1, b_2, \ldots, b_n) \in \{0, 1\}^n$ tal que $B(\overrightarrow{b}) = 1$.

Entonces, en base al coloreo C definimos:

$$b_i = \begin{cases} 1 \rightarrow C(v_{x_i}) = C(s) \\ \\ 0 \rightarrow \quad c.c \end{cases}$$

Queremos ver $B(\overrightarrow{b}) = 1$ basta ver que $D_j(\overrightarrow{b}) = 1$ $\forall j$. Entonces tomemos un j cualquiera, como para todo j los $a_{1,j}a_{2,j}a_{3,j}$ forman un triangulo entonces los 3 colores deben aparece ahí.

En particular $\exists i : C(a_{i,j}) = C(t)$ (voy desde la garra hacia arriba).

Como $e_{i,j}a_{i,j} \in E \Rightarrow C(e_{i,j}) \neq C(t)$.

Como $e_{i,j}s \in E \Rightarrow C(e_{i,j}) \neq C(s)$.

Como $ts \in E \Rightarrow C(t) \neq C(s)$.

Entonces $C(e_{i,j})$ = tercer color (el color que es distinto del color de s y t).

Ahora ya sabemos que color tienen los extremos de las garras. Entonces, vemos que $v_{l_{i,j}}e_{i,j} \in E \Rightarrow C(v_{l_{i,j}}) \neq C(e_{i,j})$ = tercer color. $\Rightarrow C(v_{l_{i,j}}) = C(t)$ o $C(s)$

Pero $tv_{l_{i,j}} \in E \Rightarrow C(v_{l_{i,j}}) \neq C(t)$ entonces no queda otra que $C(v_{l_{i,j}}) = C(s)$ que es el candidato para cuando evalue en $b_i = 1$.

El $l_{i,j}$ es un literal por lo tanto es una variable o una negación. Veamos primero el caso que es una variable:

Entonces $\exists k : l_{i,j} = x_k$ entonces $v_{l_{i,j}} = v_{x_k}$ a su vez se da que $C(v_{l_{i,j}}) = C(s) \Rightarrow C(v_{x_k}) = C(s) \Rightarrow b_k = 1$.

Entonces $l_{i,j} = x_k \Rightarrow l_{i,j}(\overrightarrow{b}) = x_k(\overrightarrow{b}) = b_k = 1 \Rightarrow D_j(\overrightarrow{b}) = 1$.

Supongamos ahora que es negación de una variable: $\exists k : l_{i,j} = \bar{x}_k \Rightarrow l_{i,j}(\overrightarrow{b}) = \bar{x}_k(\overrightarrow{b}) = 1 - b_k$

Por $k : l_{i,j} = \bar{x}_k$ podemos decir que $C(s) = C(v_{l_{i,j}}) = C(v_{\bar{x}_k})$

Pero $v_{x_k}v_{\bar{x}_k} \in E \Rightarrow C(v_{x_k}) \neq C(v_{\bar{x}_k})$

Como son distintos y $C(v_{\bar{x}_k}) = C(s)$ entonces $C(v_{x_k}) \neq C(s) \Rightarrow b_k = 0$

Entonces $\bar{x}_b(\overrightarrow{b}) = 1 - 0 = 1$
($\Leftarrow$)

$$B \text{ es satisfacible} \Rightarrow \chi(G) \leqslant 3$$

Supongamos B satisfacible $\Rightarrow \exists \overrightarrow{b} \in \{0,1\}^n : B(\overrightarrow{b}) = 1$ y hay que definir un coloreo a partir de esto. Entonces, debemos definir un coloreo C.

Definimos $C(s) = 1$ y $C(t) = 2$. Hay que analizar si el coloreo es propio en cada caso.

Entonces el lado st no crea problemas (NCP).

Para los v_l definimos, $C(v_l) = l(\overrightarrow{b})$, en otras palabras:

$$C(v_{x_i}) = b_i \in (0,1)$$
$$C(v_{\bar{x}_i}) = 1 + b_i \in (0,1)$$

Entonces hay que ver los casos:

$$\underbrace{v_{x_i}}_{0 \text{ ó } 1}\underbrace{v_{\bar{x}_i}}_{1 \text{ ó } 0} \Rightarrow NCP$$

$$\underbrace{v_l}_{0 \text{ ó } 1}\underbrace{t}_{2} \Rightarrow NCP$$

Ahora como $B(\overrightarrow{b}) = 1$ (esto hay que usarlo si o si) entonces $D_j(\overrightarrow{b}) = 1\ \forall j$.

$$\Rightarrow \forall j \exists i = i_j : l_{i_j,j}(\overrightarrow{b}) = 1$$

Si hay más de uno elijo uno por ejemplo el primero.

Coloreamos los $a_{i,j}$ de la siguiente forma (base de la garra)

$$C(a_{i_j,j}) = 2$$
$$C(a_{i,j}) = \text{le doy color 1 a uno y 0 al otro}$$

Cómo los colores de las $a_{i,j}$ son $0, 1, 2$ entonces el triangulo NCP y coloreamos los $e's$ de la siguiente forma:

$$C(e_{i,j}) = \begin{cases} 2 & \rightarrow \quad i \neq i_j \\ \\ 0 & \rightarrow \quad i = i_j \end{cases}$$

Chequeamos los lados:

$$i \neq i_j \rightarrow \underbrace{a_{i,j}}_{0 \text{ ó } 1}\ \underbrace{e_{i,j}}_{2} \Rightarrow NCP$$
$$i = i_j \rightarrow \underbrace{a_{i_j,j}}_{2}\ \underbrace{e_{i_j,j}}_{0} \Rightarrow NCP$$
$$\underbrace{s}_{1}\ \underbrace{e_{i,j}}_{0 \text{ ó } 2} \Rightarrow NCP$$

solo queda ver los $e_{i,j}v_{l_{i,j}}$

$$i \neq i_j \rightarrow \underbrace{e_{i,j}}_{2}\ \underbrace{v_{l_i,j}}_{0 \text{ ó } 1} \Rightarrow NCP$$

$i = i_j \rightarrow$

$C(e_{i_j,j}) = 0$

$C(v_{l_{i_j,j}}) = l_{i_j,j}(\overrightarrow{b}) = 1$

es igual a 1 por la elección del i_j

$$\Rightarrow \underbrace{e_{i,j}}_{0}\ \underbrace{v_{l_i,j}}_{1} \Rightarrow NCP$$

□