Discreta II: Demostraciones

Mansilla, Kevin Gaston*

16 de junio de 2023

Índice

*kevingston47@gmail.com

I. Coloreo

Algo de notación:

$d(x)$ es el grado de x, $\delta = \min\{d(x) : x \in V\}$, $\Delta = \max\{d(x) : x \in V\}$. Si $\delta = \Delta$ es un grafo regular.

I.1. Greedy

Teorema 1 *(VIT) Sea G un grafo que tiene un coloreo propio con r colores. Y sea $c_1, c_2, \ldots, c_r$ un ordenamiento de r colores. Sea $V_i = \{x \in V : color(x) = i\}$ ordenamos los vertices de G tal que V_i este antes de $V_2 \ldots$ y corremos greedy con ese orden. Entonces greedy colorea a G con $\leqslant r$ colores.*

Demostración: Se hace por inducción, cabe resaltar que greedy tiene complejidad $O(n)$.

Sea el orden de coloreo: $w_i = V_1 \cup V_2 \cup \ldots \cup V_i$ con $i = 1, 2, \ldots, r$.

Hipotesis inductiva: Greedy en ese orden, colorea w_i con $\leqslant i$ colores, si probamos HI como $w_r = V$ ya está.

Caso Base: $i = 1$

Tenemos que probar que Greedy colorea V_1 con 1 color. Pero $V_1 = \{x \in G : color(x) = c_1\}$ (con el que viene coloreado). Como $color(x)$ es un coloreo propio y todos los vertices de V_1 tienen el mismo color, no puede haber lados entre vertices de V_1. Entonces, Greedy los colorea a todos con un solo color.

Paso Inductivo: $i \to i + 1$.

Supongamos que **no vale** para $i + 1$, es decir, que es falso que Greedy colorea w_{i+1} con $\leqslant i + 1$ colores.

Entonces **debe existir vertice** $x \in w_{i+1}$ **coloreado con color** $i + 2$ (si coloreamos con colores $1, 2, \ldots, r + i$). Para que Greedy se vea forzado a colorear x con $i + 2$ debe existir un vecino de x anterior a x en el orden que estoy coloreando que tenga color $i + 1$, llamemoslo y.

Por HI, w_i puede ser coloreado por Greedy con a lo sumo i colores y como $i+1, i+2 > i$ entonces $x, y \notin w_i$. Pero $x \in w_{i+1}$ e y es anterior a x entonces $y \in w_{i+1}$. Entonces Tenemos:

$$x, y \in w_{i+1} = V_1 \cup V_2 \cup \ldots V_i \cup V_{i+1}$$

$$x, y \notin wi = V_1 \cup V_2 \cup \ldots V_i$$

Entonces si $x, y \in V_{i+1}$, por lo que $color(x) = color(y) = c_{i+1}$. Pero es un absurdo porque el coloreo original era propio y x, y son vecinos. □

Corolario 1 *Existe un orden de los vértices tal que Greedy, en ese orden, colorea G con $\leqslant \chi(G)$ colores.*

Demostración:

Sea $\chi(G) = min\{k : \exists \text{coloreo propio de } G \text{ en } k \text{ colores}\}$

Entonces existe un coloreo propio de G con $\chi(G)$ colores. Usamos el VIT ordenando los colores de alguna forma. Además por el VIT Greedy en ese orden colorea G con $\chi(G)$ colores o menos.

Es importante dar una una cota para $\chi(G)$. Prueba:

Cuando vamos a colorear un vertice x con Greedy debemos mirar los colores de los vecinos anteriores a x y no le podemos dar ninguno de esos colores. El peor caso posible es que todos los vecinos de x sean vecinos anteriores y que todos tengan colores distintos.

En el peor caso posible debo **eliminar** $d(x)$ colores con $d(x) = \Delta$ a lo sumo elimino Δ colores. Asi que si tengo $\Delta + 1$ colores disponibles siempre habrá uno que sobre. Entonces $\chi(G) \leqslant \Delta + 1$.

Los grafos completos alcanzan la cota $\Delta(K_n) = n - 1 \to \chi(K_n) = n = \Delta + 1$. Y los grados impares $\Delta(C_{2r+1}) = 2 \to \chi(C_{2r+1}) = 3 = \Delta + 1$. $\square$

Definición 1 *Un camino en G es una sucesión de vértices $x_1, x_2, \ldots, x_r$ tales que $x_i x_{i+1} \in E$. $x \sim y \iff \exists$ camino entre x e y (es una relación de equivalencia).*

Teorema 2 *(Broks) Sea G un grado conexo que no sea un K_n o C_{2r+1}. Entonces $\chi(G) \leqslant \Delta(G)$.*

Demostración: Para el caso no regular.

Sea $x \in V$, tal que el grado $d(x) = \delta$. Corramos DFS o BFS a partir de x, como G es conexo, encuentro todos los vertices, en el orden DFS o BFS.

Vamos a correr Greedy en orden inverso. Por lo que si x es el último vértice, en el orden DFS o BFS todo vertice es puesto por algún vecino que ya estaba. Entonces implica que todo vértice ($\neq x$) tiene un vecino anterior en el orden BFS (o DFS) implica que en el orden inverso todo vertice $\neq x$ tiene un vecino posterior.

Entonces cuando Greedy colorea un vértice z, debe mirar los vecinos anteriores ($z \neq x$) como z tiene un vecino posterior, los anteriores son $\leqslant d(x) - 1$.

Entonces Greedy va a eliminar a los sumo $d(z) - 1 \leqslant \Delta - 1$ colores. Entonces si tengo Δ colores, siempre va a sobrar uno para colorear a z si $z \neq x$ y para colorear x eliminamos a lo sumo $d(x) = \delta < \Delta$ colores. Por lo tanto lo puedo colorear. $\square$.

Problema: "k-color": Dado G es $\chi(G) \leqslant k$? 1−color es trivial (no es polinomial). 2−color es polinoimal (hay algún algoritmo que corre en tiempo polinomial en un rango de la entrada). Idea (la demostración por ahora no) Basta correrlo para grados conexos. Tomamos un $x \in V$, corremos BFD empezando en x. Si:

$$\begin{aligned} N(z) &= \text{nivel } z \text{ en el arbol BFS} \\ &= \text{distancia entre } z \text{ y } x \text{ en el arbol BFS} \\ &= \text{distancia entre } z \text{ y } x \text{ en } G \end{aligned}$$

Sea:

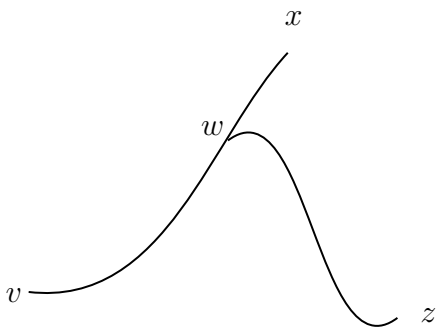
$$\begin{aligned} C(z) &= (N(z) \, mod \, 2) \\ &IF(C \text{ es propio, return si, es 2−colorable}) \\ &ELSE(\text{return no es 2−colorable}) \end{aligned}$$

El algoritmo es polinomial porque BFS es $O(m)$ y chequear que es propio es $O(m)$, lo que hay que probar es si es correcto. Entonces supongamos que la respuesta es "no es 2−colorable".

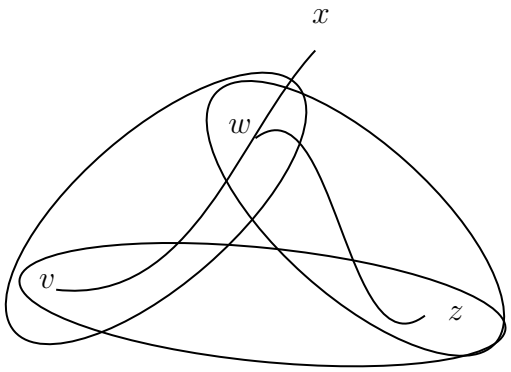
$$\Rightarrow \exists v, x : c(v) = c(z) \wedge vz \in E$$

Entonces $d(x, v) = d(x, z)\ mod\ 2$

tomamos un camino entre x y v en BFS y un camino entre x y z en BFS y sea w el único vertice en común (como lo muestra la siguiente figura).



Miramos el ciclo en G: $w \ldots \underbrace{vz}_{\text{cruzo a } z} \ldots \underbrace{w}_{\text{vuelvo a } x}$ (como lo muestra la siguiente imagen).



Calculamos la longitud de este ciclo:

$$\begin{aligned}
longitud &= 1 + d(v, w) + d(z, w) \\
longitud\, mod\, 2 &= (1 + d(v, w) + \ldots + d(z, w))\, mod\, 2 \\
&= (1 + d(v, w) + d(z, w) + 2d(x, w))\, mod\, 2 \\
&= (1 + d(x, v) + d(x, z))\, mod\, 2 = (1 + \overbrace{c(x) + c(z)}^{=0})\, mod\, 2 \\
&= 1
\end{aligned}$$

Es un ciclo impar, entonces no se puede colorear con 2 colores $\Rightarrow \chi_(G) \geqslant 3$. $\square$

II. Flujos y networks:

Un grafo dirigido es un par (V, E) con $E \subseteq V \times V$. Un network (estructura) es un triple (V, E, C) donde (V, E) es un grafo dirigido y $C : E \to \mathbb{R}_{\geqslant 0}$ es la capacidad.

Notación: Si g está definida en E y $A, B \subseteq V$ definimos:

$$g(A,V) = \sum_{x\in A, y\in B, xy\in E} g(x,y)$$

Dado $x \in V$, g definida en E. Definimos:

$$out_g(x) = g(\{x\}, V) = \sum_{y\in V, xy\in E} g(x,y)$$
$$in_g(x) = g(V, \{x\}) = \sum_{y\in V, yx\in E} g(y,x)$$

Si $\overrightarrow{xy} = (x,y)$, entonces:

$$\Gamma^+(x) = \{y : (x,y) \in E\}$$

En cambio si $\overrightarrow{yx} = (y,x)$, tenemos:

$$\Gamma^-(x) = \{y : (y,x) \in E\}$$

Entonces, podemos definir:

$$out_g(x) = \sum_{y\in\Gamma^+(x)} g(\overrightarrow{xy})$$
$$in_g(x) = \sum_{y\in\Gamma^-(x)} g(\overrightarrow{yx})$$

Dado un network (V, E, C) y vertices s, t un flujo de s a t es una función f en los lados con (s es la fuente y t el sumidero):

a) $0 \leqslant f(\overrightarrow{xy}) \leqslant c(\overrightarrow{xy})$ para todo $\overrightarrow{xy} \in E$.

b) $In_f = Out_f$, $\forall x \neq s, t$ (la red no tiene perdidas en los vertices de transito).

c) $In_f(s) = Out_f(t) = 0$

El valor del flujo es $v(f) = out_f(s) - in_f(s) = out_f(s)$. Un flujo es máximal si $v(g) \leqslant v(f)$, $\forall g$ flujo de s a t. Entonces, el problema de maxflow, dado un network hallar flujo maximal.

Propiedad: $V(f) = In_f(t)$

$$f(V,V) = \sum_{x,y\in V, x,y\in E} f(\overrightarrow{xy}) = \sum_{x\in V}\left(\sum_{y\in V, \overrightarrow{xy}\in E} f(\overrightarrow{xy})\right)$$
$$= \sum_{x\in V} out_f(x)$$

Por otro lado,

$$f(V,V) = \sum_{z\in V, \overrightarrow{zx}\in E} f(\overrightarrow{zx}) = \sum_{x\in V}\left(\sum_{z\in V,\overrightarrow{zx}\in E} f(\overrightarrow{zx})\right)$$
$$= \sum_{x\in V} In_f(x)$$

Entonces

$$\sum In_f(x) = \sum out_f(x)$$
$$\sum_{x\in V}(In_f(x) - out_f(x)) = 0, \text{ si } x = s,t$$
$$= \{\text{Como en } s \text{ no entra flujo y en } t \text{ no sale}\}$$
$$\Rightarrow \underbrace{In_f(s)}_{=0} - out_f(s) + In_f(t) - \underbrace{out_f(t)}_{=0} = 0$$
$$\Rightarrow -out_f(s) + in_f(t) = 0$$
$$out_f(s) - In_f(t) = v(f) \geq 0$$
$$\square$$

Definición 2 *Un camino dirigido desde x a y es una sucesión de vértices $x_0, x_1, \ldots, x_r$ con $x_0 = x, x_r = y$. Pues $\overrightarrow{x_i x_{i+1}} \in E$.*

II.1. Algoritmo Greedy aplicado a networks:

Empezar con flujo $f = 0$
Mientras se pueda, hacer lo siguiente
Hallar camino dirigido desde s a t tal que $f(\overrightarrow{x_i x_{i+1}}) < c(\overrightarrow{x_i x_{i+1}})$
Tomar $e = min\{c(\overrightarrow{x_i x_{i+1}}) - f(\overrightarrow{x_i x_{i+1}}) : \overrightarrow{x_i x_{i+1}} \in E\}$
Tomar f^* tal que:

$$f^*(\overrightarrow{xy}) = \begin{cases} f(\overrightarrow{xy}) & si \quad xy \neq \overrightarrow{x_i x_{i+1}} \, \forall i \\ f(\overrightarrow{x_i x_{i+1}}) + e & si \quad \overrightarrow{xy} = \overrightarrow{x_i x_{i+1}} \end{cases}$$

Hacer $f = f^*$

II.2. FF (Greedy alterado)

Es alterado porque ahora se puede devolver flujo además de mandar.

Definición 3 *Dado un flujo f en un network un camino aumentante (o $f-$camino aumentante) es una sucesión de vertices $x_0, x_1, \ldots x_r$ tal que $x_0 = s$ y $x_r = t$ y tal que $\forall t < r$ pasa una de las siguientes dos cosas:*

1. $\overrightarrow{x_i x_{i+1}} \in E \wedge f(\overrightarrow{x_i x_{i+1}}) < c(\overrightarrow{x_i x_{i+1}})$ *(puedo mandar más flujo por ese lado - lado fordward).*

2. $\overrightarrow{x_{i+1} x_i} \in E \wedge f(\overrightarrow{x_{i+1} x_i}) > 0$ *(importa que haya mandado flujo - lado backward).*

Algoritmo de Ford-Fulkerson (Greedy con camino aumentante):

1. $f = 0$
2. Buscar un camino aumentante $s = x_0, x_1, \ldots, x_r = t$.
3. Defino:

$$\epsilon_i = \begin{cases} c(\overrightarrow{x_i x_{i+1}}) - f(\overrightarrow{x_i x_{i+1}}) & \text{en los lados fordward} \\ f(\overrightarrow{x_{i+1} x_i}) & \text{no puedo devolver más de lo que me mandan} \end{cases}$$

   $\epsilon = min\{\epsilon_i\}$

4. Cambiar f a lo largo del camino.

$$f(\overrightarrow{x_i x_{i+1}}) = f(\overrightarrow{x_i x_{i+1}}) + \epsilon \text{ en fordward}$$
$$f(\overrightarrow{x_{i+1} x_i}) = f(\overrightarrow{x_{i+1} x_i}) - \epsilon \text{ en backward}$$

   y repetir 2 hasta que se pueda.

5. return f.

Correctitud de F-F:

Definición 4 *Un corte es un $S \subseteq V$ tal que $s \in S, t \notin S$.*

Ejemplo:

$$S_1 = \{s\}$$
$$S_2 = V - \{t\}$$

Son los cortes triviales, cabe resaltar que puede haber a lo sumo hay 2^{n-2} cortes.

La capacidad de un corte es:

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

Un corte S es minimal si $cap(S) = cap(T), \forall T$ corte.

Lemma 1 *Al cambiar el flujo f como en el algoritmo de F-F a un f^*, lo que queda es flujo y $v(f^*) = v(f) + \epsilon$*

Demostración:

Lados fordward:

$$f^*(\overrightarrow{x_i x_{i+1}}) = f(\overrightarrow{x_i x_{i+1}}) + \epsilon \leqslant c(\overrightarrow{x_i x_{i+1}})$$
$$\text{con } \epsilon \in \epsilon_i = c(\overrightarrow{x_i x_{i+1}}) - f(\overrightarrow{x_i x_{i+1}})$$

Lados backward:

$$f^*(\overrightarrow{x_{i+1} x_i}) = f(\overrightarrow{x_{i+1} x_i}) - \epsilon \geqslant 0$$
$$\epsilon \in \epsilon_i = f(\overrightarrow{x_{i+1} x_i})$$

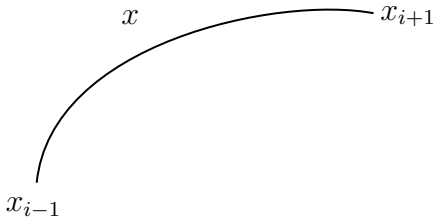
Tenemos que ver que $in_f(x) = out_f(x), \forall x \neq s, t$.

Si $x \neq x_i$ entonces:

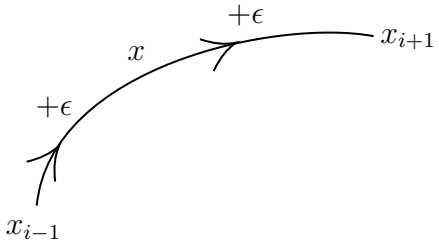
$$in_{f^*}(x) = in_f(x)$$
$$out_{f^*}(x) = out_f(x)$$
$$\Rightarrow in_{f^*}(x) = out_{f^*}(x)$$

Supongamos ahora que $x = x_i, x = s, t \Rightarrow \exists x_{i-1}, x_{i+1}$. Tenemos lo siguientes casos:

Caso 1: $x_{i-1}x_i, x_i x_{i+1}$ fordwards. Como lo muestra la siguiente imagen:



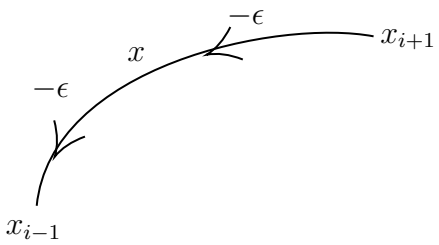
Si mandamos ϵ de x_{i-1} a x_i y de x_i a x_{i+1} entonces:

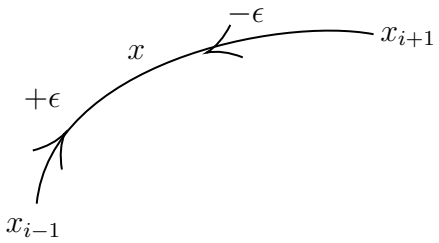


Entonces tenemos:

$$in_{f^*}(x_i) = in_f(x) + \epsilon$$
$$out_{f^*}(x_i) = out_f(x) + \epsilon$$
$$\Rightarrow in_{f^*}(x_i) = out_{f^*}(x_i)$$

Caso II: Ambos lados backwards





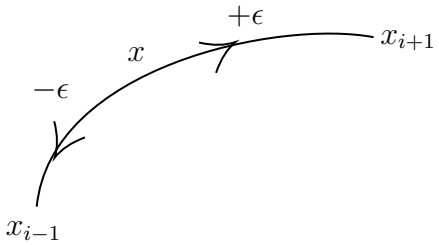
Entonces:

$$\begin{aligned} out_{f^*}(x_i) &= out_f(x_i) - \epsilon \\ in_{f^*}(x_i) &= in_f(x_i) - \epsilon \\ \Rightarrow in_{f^*}(x_i) &= out_{f^*}(x_i) \end{aligned}$$

Caso III:
Entonces:

$$\begin{aligned} out_{f^*}(x_i) &= out_f(x_i) \\ in_{f^*}(x_i) &= in_f(x_i) - \epsilon + \epsilon \\ \Rightarrow in_{f^*}(x_i) &= out_{f^*}(x_i) \end{aligned}$$

Caso IV:



Entonces:

$$\begin{aligned} in_{f^*}(x_i) &= in_f(x_i) \\ out_{f^*}(x_i) &= out_f(x_i) - \epsilon + \epsilon \\ \Rightarrow in_{f^*}(x_i) &= out_{f^*}(x_i) \end{aligned}$$

Por lo que:

$$\begin{aligned} v(f^*) &= out_{f^*}(S) \\ &= out_f(S) + \epsilon \\ &= v_f(S) + \epsilon \end{aligned}$$

II.3. Max-flow-min-cut theorem

Teorema 3 *a. f flujo, S corte $\Rightarrow v(f) = f(S,\overline{S}) - f(\overline{S},S)$*

b. f flujo, S corte $\Rightarrow v(f) \leqslant cap(S)$

c. Si f es flujo las siguientes son equivalentes:

1. *$\exists S$ corte: $v(f) = cap(S)$*
2. *f es maximal.*
   $(1 = 2)$ dice:
   "f maximal $\Longleftrightarrow \exists S$ corte $v(f) = cap(S)$"
   y se suele llamar 'max-flow-min-cut theorem'.
3. *$\nexists f-$caminos aumentanes entre s y t*

Demostración:

a. f flujo, S corte $\Rightarrow v(f) = f(S,\overline{S}) - f(\overline{S},S)$

Prueba:

Sea $A = \sum_{x\in S}[out_f(x) - in_f(x)]$, como $out_f(x) = in_f(x), \forall x \in s,t$. Si tenemos que $s \in S$ (todos ceros menos el correspondiente al S) y $t \notin S$, entonces:

$$\begin{aligned} A &= out_f(S) - \overbrace{in_f(S)}^{=0} + 0 + 0 \\ A &= v(f) \end{aligned}$$

Por otro lado:

$$\begin{aligned} A = \sum_{x\in S}[out_f(x) - in_f(x)] &= \sum_{x\in S} out_f(x) - \sum_{x\in S} in_f(x) \\ &= \sum_{x\in S} \overbrace{f(\{x\},V)}^{\text{todo lo que sale}} - \sum_{x\in S} \overbrace{f(V,\{x\})}^{\text{todo lo que entra}} \\ &\text{por definición:} \\ &= f(S,V) - f(V,S) \\ &= f(S, S\cup\overline{S}) - f(S\cup\overline{S}, S) \\ &= \cancel{f(S,S)} + f(S,\overline{S}) \cancel{- f(S,S)} - f(\overline{S},S) \\ &= f(S,\overline{S}) - f(\overline{S},S) \end{aligned}$$

b. f flujo, S corte $\Rightarrow v(f) \leqslant cap(S)$

Prueba:

Sale directo de a):

$$\begin{aligned} v(f) &= \underbrace{f(S,\overline{S}) - \underbrace{f(\overline{S},S)}_{\geqslant 0}}_{\leqslant 0} \\ &\leqslant f(S,\overline{S}) \leqslant \underbrace{c(S,\overline{S})}_{0\leqslant f\leqslant c} = cap(S) \end{aligned}$$

c. Si f es flujo las siguientes son equivalentes:

1. $\exists S$ corte: $v(f) = cap(S)$
2. f es maximal.
   $(1 = 2)$ dice:
   "f maximal $\iff \exists S$ corte $v(f) = cap(S)$"
   y se suele llamar 'max-flow-min-cut theorem'.
3. $\nexists f-$caminos aumentanes entre s y t
   y si se cumplen, el s es minimal.

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

1) $\Rightarrow$ 2) Sea g un flujo por la parte b, del teorema:

$$v(g) \leqslant cap(S) = v(f) \Rightarrow v(g) \leqslant v(f) \Rightarrow f \text{ maximal}$$

Además S es minimal pues T corte:

$$\Rightarrow cap(T) \geqslant v(f) = cap(S) \Rightarrow S \text{ minimal}$$

2) $\Rightarrow$ 3) Supongamos f maximal y que es falso, $\neg 3) \Rightarrow \neg 2)$ (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.

3) $\Rightarrow$ 1), 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, por la parte a:

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

Entonces:

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

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

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

Conclusión: $f(\overrightarrow{xy}) \not< c(\overrightarrow{xy}) \Rightarrow f(\overrightarrow{xy}) = c(\overrightarrow{xy})$ 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:

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

□

Corolario 2 *Si F-F termina, termina con flujo minimal*

Complejidad de F-F: Primero veamos.

Complejidad de Greedy con camino aumentante solo con lados fwd:

1. Buscamos camino dirigido $\rightarrow O(m)$.
2. Cada camino que se usa, satura al menos un lado.
3. Por 2, hay $O(m)$ caminos. $\Rightarrow O(m^2)$.

F-F:

1. Buscamos camino dirigido $\rightarrow O(m)$.
2. Cada camino que se usa, satura o vacia al menos un lado.
3. **No vale en greedy porque no devuelvo flujo.**
   Como en F-F un lado puede desaturarse y luego volver a saturarse o vaciarse o luego llenarse, entonces 2) $\Rightarrow$ 3) en F-F. Entonces, la complejidad es ∞ porque puede no terminar. Hay que buscar caminos de longitud minima, en ese caso si término.

II.4. Teorema de integralidad

Teorema 4 *(Integralidad) Si las capacidades son enteras F-F termina y el flujo maximal que se obtiene es entero.*

Demostración:

Si f es entero, entonces:

$$\epsilon_i = \begin{cases} c(\overrightarrow{x_i x_{i+1}}) - f(\overrightarrow{x_i x_{i+1}}) & \rightarrow \text{ entero} \\ f(\overrightarrow{x_{i+1} x_i}) & \rightarrow \text{ entero} \end{cases}$$

los ϵ_i serán enteros entonces ϵ sera entero, entonces:

$$f = \begin{cases} f+\epsilon & \rightarrow \text{ entero} \\ \\ f-\epsilon & \rightarrow \text{ entero} \end{cases}$$

seguira siendo entero. Además como empezamos con $f = 0$ (entero), entonces f entero es un invariante, que al terminar nos dice que f será entero y termina porque:

$$v(f_{nuevo}) = v(f_{viejo}) + \epsilon \geqslant v(f_{viejo}) + 1$$

Alguna vez termina pues $v(f) \leqslant cap(\{s\})$

II.5. Complejidad EK

Teorema 5 *La complejidad de EK es $O(m^2n)$*

Corolario 3 *Siempre existen flujos maximales (siempre termina y termina con un flujo maximal).*

Prueba:

Dados x, z vértices y f flujo, definimos:

$$\begin{aligned} d_f(x,z) &= \text{menor longitud de un } f-\text{ca entre } x \text{ y } z \\ &= 0\,, \text{si } x = z \\ &= \infty\,, \text{si no existe } f-\text{ca entre } x \text{ y } z \end{aligned}$$

Sean $f_0, f_1, f_2, f_3, \ldots$ los flujos producidos por EK. Denotamos para un flujo k.

$$\begin{aligned} d_k(x) &= d_{f_k}(s,x) \\ b_k(x) &= d_{f_k}(x,t) \end{aligned}$$

que representan lo siguiente:

$$\underbrace{s \ldots\ldots x}_{d_k} \underbrace{\ldots\ldots t}_{b_k}$$

Vamos a demostrar que $d_k \leqslant d_{k+1}$, $b_k \leqslant b_{k+1}$ (las distancias no disminuyen, la demostración para $b_k \leqslant b_{k+1}$ es análoga)

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

Sea z el vertice inmediatamente anterior a x en ese camino $(s \dots zx)$. 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.

Pero primero supongamos que $\overrightarrow{zx} \in E$. Es $f_k(\overrightarrow{zx}) = c(\overrightarrow{zx})$ o no?. 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:

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

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 $f - k$ ca del tipo $s \dots zx$.

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

Si $f_k(\overrightarrow{xz}) = 0$. Como $\underbrace{s \dots z}_{f_{k+1}} x$ es ca. Entonces, zx es backward, $s \dots \overleftarrow{zx}$, entonces $f_{k+1}(\overrightarrow{xz}) > 0$.

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

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).

Parte final de la demo, esto sería la complejidad de EK (teorema 5): lo anterior era que las distancias no disminuyen que es muy importante para esta prueba.

Entonces, diremos que un lado se vuelve critico al pasar de f_k a f_{k+1} si se **satuta** 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, 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 $\overrightarrow{zx} \in E$, el analisis es similar, se volvio critico porque se vacio, entonces para volver a ser critico debe saturarse o volverse a vaciar, para lo cual debo 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:

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

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

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

$$d_k(z) = d_k(x) + 1$$
$$d_k(x) = d_l(z) + 1$$

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 $O(m) * O(nm) = O(nm^2)$.

II.6. Dinitz

Teorema 6 *(Dinitz) La cantidad de NA usados es $O(n)$*

Prueba:

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

Sabemos por EK que $d \leqslant d'$, acá queremos ver solo el $<$. Si lo probamos, entonces como las distancias entre s y t aumentan entre NA, solo podemos tener $O(n)$ networks auxiliares y estaría demostrando el teorema.

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. primero. Entonces si falta un vertice, tomamos un x cualquiera por lo que $x_i \notin NA \Rightarrow d(t) < d(x_i)$.

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

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

Si falta al menos un lado (parte b). 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. $d(x_{i+1}) < i + 1$ Entonces hacemos algo similar al caso a. pero con x_{i+1}.

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

Caso 2. $d(x_{i+1}) \not< i + 1$
Pero $d(x_{i+1}) \leqslant d'(x_{i+1}) = i + 1$.
Como i es el primer indice con $\overrightarrow{x_i x_{i+1}} \notin NA$
$\overrightarrow{x_j x_{j+1}} \in NA$, $\forall j < i$, como NA es por niveles.
$\Rightarrow d(x_j) = j, \forall j < i$. Es decir, que vale $\forall j \leqslant i + 1$.

En particular $d(x_i) = i, d(x_{i+1}) = i + 1$. Entonces, x_i este en el nivel i y x_{i+1} esta en el nivel $i + 1$. Entonces $\overrightarrow{x_{i+1}x_i} \notin NA$.

Como sabemos que $\overrightarrow{x_{i+1}x_i} \notin NA$. Entonces tenemos que no esta ninguno de los dos, es decir, que $\overrightarrow{x_ix_{i+1}} \notin NA$ 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 $\overrightarrow{x_ix_{i+1}}$ podría estar, pero no puede mandar ni devolver flujo.

Como no esta, caso fw saturado o caso bw vacio. Pero $\overrightarrow{x_ix_{i+1}} \in NA'$, entonces que se desaruto o lleno un poco según el caso al ir de NA a NA'.

Pero entonces, para pasar de NA a NA' tengo que hacer usado el lado al revés, pero no lo puedo haver usado porque x_ix_{i+1} no existe. □

Corolario 4 *La complejidad de cualquier algoritmo que use NA, es*

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

Dinitz original: Además de NA como lo construimos se depura con la poda al principio y luego de cada camino usado borrando todos los vértices que no tengan lado de salida.

Esto garantiza que cuando buscamos caminos todo vertice recisado tiene al menos un lado par adelante.

II.6.1. Complejidad Diniz

Teorema 7 *Complejidad de Diniz es* $O(n^2m)$

Prueba:

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

En el primer NA, tengo que depurar DFS esto funciona en $O(r) = O(n)$, pues tenemos que retroceder.

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 2 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 - 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.

Pseudocodigo: para flujo bloquenate del NA.

```
g=0
flag=1
while (flag) {
    k = s (pivote)
    Inicializar el camino en s
    while (x != t and flag) {
        if ($\Gamma^{+}(x)$ != vacio) AVANZAR(x,p,g,NA)
        else if (x != s) RETROCEDER(...)
            else flag=0
    if (x == t) INCREMENTAR(g)
    }
return g
}
AVANZAR
    tomar y \in $\Gamma^{+}(x)$
    agregar y al camino p
    x = y
RETROCEDER
    tomar y = vertice anterior a x en el camino
    borrar el lado $\overrightarrow{y,x}$ del NA
    borrar x del camino
    x = y
INCREMENTAR
    recorrer el camino p para calcular $\epsilon$
    aumentar g en $\epsilon$ a lo largo de p
    borrar lados saturados
```

Entonces, AVANZAR es $O(1)$, RETROCEDER es $O(1)$ y INCREMENTAR es $O(r) = O(n)$.

II.6.2. Complejidad Dinitz-Even

Teorema 8 *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:

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

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 $= O(r) \Rightarrow O(m)$.

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 $O(m)$ con complejidad $O(n)$ cada una, entonce $O(nm)$ total.□

II.7. Complejidad Wave

Teorema 9 *La complejidad de Wave es $O(n^3)$. Como es del tipo Dinic, la complejidad es $O(n^2)$ por la complejidad de hallar el flujo bloqueante.*

Lema: Hay a lo sumo $O(n)$ olas hacia adelante (hacia atrás).

Prueba teorema 9, usando el lema:

Pues en cada ola hacia adelante (salvo en la última) al menos un vertice se bloquea, si los Forwardbalence (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 ejecura una vez, entonces es $O(1)$.

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

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 $O(\delta(x))$ en cada x, entonces:

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

.

Con backwardbalance (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 $M(x)$, por lo que esta acotado por $O(\#\,\text{número de elementos máximos de } M(x)) = O(\delta(x))$, entonces $V = O(\delta(x)) = O(m)$.

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

Si lo saque, con bwb un fwb los puede volver a agregar, entonces $M(x)$ puede crecer y puede pasar que borremos un vertice de $M(x)$ pero después lo volvamos a agregar.

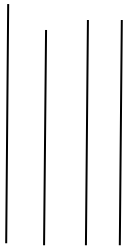
Pero esto no pasa, pues para que un $bwb(x)$ remueva a y de $M(x)$, lo primero que tiene que pasar para poder hacer esto es ejecurar $bwb(x)$, pero para poder ejecutarlo x debe estar bloqueado. Y si esta bloqueado nadie nunca más le va a poder mandar flujo (en particular y), entonces una vez que removó a y no lo voy a poder agregar nuevamente a $M(x)$. Entonces:

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

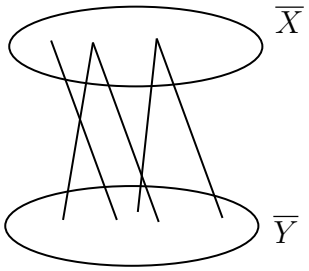
III. Matchings

Definición 5 *Un matching es un grafo G, y M es un subgrafo con $d_M = 1, \forall x \in V(M)$*

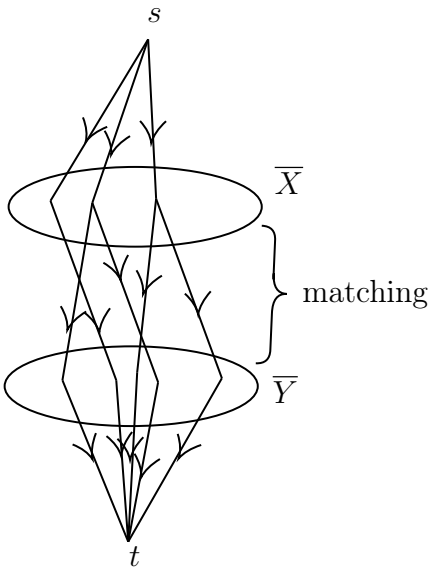
Ejemplo:



Problema: Dado G, hallar un matching en G con la mayor cantidad de lados posibles. Solo veremos matching en grafos bipartitos, como se muestra en figura de abajo.



Reduciremos el problema a un problema de flujo maximal. Tenemos que transformar un grafo bipartito G en partes de $\bar{X}$ y $\bar{Y}$, en un network. Los vertices del network serán $\{s,t\} \cup \bar{X} \cup \bar{Y}$, y los lados $\left\{\overrightarrow{xy} : x \in \bar{X}, y \in \bar{Y}, xy \in E\right\} \cup \left\{\overrightarrow{sx} : x \in \bar{X}\right\} \cup \left\{\overrightarrow{yt} : y \in \bar{Y}\right\}$. Y las capacidades de los vertices son todas 1. Ejemplo:



Propiedad: Flujos enteros maximales en ese network se corresponden con matching maximales en G.

Prueba: Dado un flujo f (maximal entero o no) en el network definimos:

$$M_f = (V_f, E_f) \text{ con } V_f = \{v \in V : n_f(v) = 1\}$$
$$E_f = \left\{xy, x \in \bar{X}, y \in \bar{Y} : f(xy) = 1\right\}$$

M_f es un matching Sea $v \in V_q$. Si $v \in \bar{X}$, supongamos que $\delta_M(v) \neq 1$, pero $v \in V_f \Rightarrow In_f(v) = 1 \Rightarrow out_f(v) = 1$. $\Rightarrow \exists y \in \bar{Y} : f(\overrightarrow{xy}) = 1 \Rightarrow In_f(y) = 1$. (observacion: como f es entero y $c = 1$ entonces $In_f(v) = 0$ ó 1 $\forall v \neq s, t$)

Entonces $y \in V_f$ y $vy \in E_f \Rightarrow \delta_{M_f}(v) \geqslant 1$. Sino es 1, entonces $\exists z \neq y : vz \in E_f \Rightarrow f(\overrightarrow{vz}) = 1 \Rightarrow out_f(v) \geqslant f(\overrightarrow{vy}) + f(\overrightarrow{vz}) = 2$. Lo que es un absurdo, pues $in_f(v) = 1$.

Entonces $\delta_M(v) = 1,\ \forall v \in V_f \cap \bar{X}$.

De la misma forma lo demostramos si $v \in \bar{Y}$. $In_f(v) = 1 \Rightarrow \exists x \in \bar{X} : f(\overrightarrow{xv}) = 1 \Rightarrow \delta_M(v) \geqslant 1$ y si fuera 2 o más $\exists x \neq x : f(\overrightarrow{zv}) = 1 \Rightarrow In_f(v) \geqslant 2$ Absurdo.

Por lo tanto M_f es un matching. Entonces:

$$\begin{aligned} |E_{M_f}| &= \#\{xy \in E_f\} \text{ como es un matching} \\ &= \#\left\{x \in \bar{X} : In_f(x) = 1\right\} = out_f(s) = v(f) \end{aligned}$$

Y viceversa, sea M un matching en G y definimos:

$$f(\overrightarrow{xy}) = \begin{cases} 1 \ \rightarrow \ xy \in E(M) \\ \\ 0 \ \rightarrow \quad \text{c.c.} \end{cases}$$

$$f(\overrightarrow{yt}) = \begin{cases} 1 \ \rightarrow \ y \in V(M) \cap \bar{Y} \\ \\ 0 \ \rightarrow \quad \text{c.c.} \end{cases}$$

$$f(\overrightarrow{sx}) = \begin{cases} 1 \ \rightarrow \ xy \in V(M) \cup \bar{X} \\ \\ 0 \ \rightarrow \quad \text{c.c.} \end{cases}$$

Definición 6 *Si $W \subseteq V$, entonces*

$$\begin{aligned} \Gamma(\bar{W}) &= \left\{z : \exists w \in \bar{W} : zw \in E\right\} \\ &= \cup_{w \in \bar{W}} \Gamma(w) \end{aligned}$$

Definición 7 *Un matching $M = (V_M, E_M)$ en G es perfecto si $V_M = V(G)$ (si sus vertices coinciden con los de G).*

Definición 8 *(matching completo) Si $G = (\bar{X} \cup \bar{Y}, E)$ es bipartito. Un matching $M = (V_M, E_M)$ es completo de $\bar{X} \cap \bar{Y}$ si $V_M \cap \bar{X} = \bar{X}$.*

III.1. Teorema de Hall

Teorema 10 *(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 $\underbrace{|S| \leqslant |\Gamma(S)|}_{P} \Rightarrow \underbrace{\exists \text{ un matching completo de } \bar{X} \cap \bar{Y}}_{Q}$.

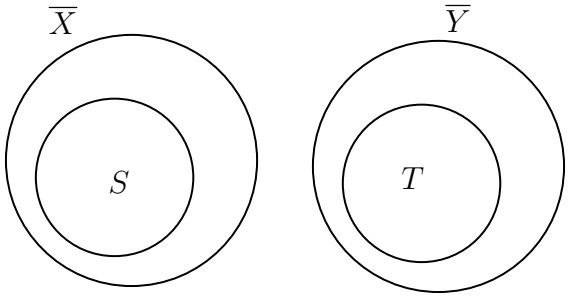
Lo veremos de la siguiente forma $[P \Rightarrow Q] = [\neg Q \Rightarrow \neg P]$ por contrareciproca.

Probaremos que sino existe matching entonces no se cumple Hall.

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, la correr E-K).

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



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 hacer 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:

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 $z \in Y$. 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 $y \in c \cap Y = T$, 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 $v(f) \neq |x|$, 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

$$TyS - S_0 \tag{10}$$

Finalmente:

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

III.2. Teorema del matrimonio de konig

Teorema 11 *(matrimonio de konig) Todo grafo bipartito regular tiene un matching perfecto (todos los vertices dorman parte del matching).*

Prueba: Dado un conjunto de vertices definimos:

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

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

$|E_w| = |\{xw : x \in W\}|$

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

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

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

Como G es regular

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

Si $W \subseteq Y$ 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 $y \in W$. 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 $S \subseteq X \Rightarrow |E_S| = \Delta|S|$. A su vez, $\Gamma(S) \subseteq Y \Rightarrow |E_{\Gamma(S)}| = \Delta|\Gamma(S)|$. Entonces podemos decir que $\cancel{\Delta}|S| \leqslant \cancel{\Delta}|\Gamma(S)| \Rightarrow |S| \leqslant |\Gamma(S)|$.

Entonces se satisface la condición de Hall, entonces podemos afirmar que existe matching compreto 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
$\Delta|\bar{X}| = \Delta|\bar{Y}|$
$|\bar{X}| = |\bar{Y}|$

Corolario 5 *(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 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 $\widetilde{G}$ siguie siendo regular. Por lo que tiene un matching perfecto. Coloreo esos lados con el color 2, los remuevo, etc.

Siempre va a ser regular hasta el final y terminamos coloreando con Δ colores. Qué pasa si G no es regular?

Lemma 2 *G bipartito, entonces existe H bipartito regular tal que $G \subseteq H$. Entonces $\Delta(G) = \Delta(H)$.*

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

IV. Matchings con pesos: Minimizar la suma

Observaciones triviales: si A tiene entrada no negativa tal que exista un matching de ceros, es decir, un matching donde todas las entradas correspondientes al matching son ceros.

Entonces ese matching minimiza la suma, pues la suma da 0 y cualquier otro matching tiene suma mayor o igual a 0.

Lemma 3 *Sea A una matriz de pesos ($n \times n$). Sea $\widetilde{A}$ la matriz que se obtienen de A restando una constante a cada entrada de una sola fila o columna de A. Entonces un matching minimiza la suma relativa a A sii la minimiza (la suma) relativa a $\widetilde{A}$.*

Prueba:

Observemos que un matching da origen a una permutación de $\{1, 2, \ldots, n\}$, y viceversa. Sea σ la permutación correspondiente a un matching M (que puede no minimizar la suma). Entonces la suma correspondiente al matching M, relativa a A es:

$$S = \sum_{i=1}^{n} A_{i,\sigma(i)}$$

y relativa a $\widetilde{A}$ es:

$$\widetilde{S} = \sum_{i=1}^{n} \widetilde{A}_{i,\sigma(i)}$$

$\widetilde{A}$ se obtiene de A restando una constante c de alguna fila o columna de A para concretar supongamos que es la fila k. Entonces:

$$\widetilde{A}_{i,j} = \begin{cases} A_{i,j} & si \quad i \neq k \\ \\ A_{k,j} & si \quad i = k \end{cases}$$

Entonces:

$$\begin{aligned} \widetilde{S} = \sum_{i=1}^{n} A_{i,\sigma(i)} &= \left(\sum_{i \neq k} \widetilde{A}_{i,\sigma(i)} \right) + \widetilde{A}_{k,\sigma(k)} \\ &= \left(\sum_{i \neq k} A_{i,\sigma(i)} \right) + A_{k,\sigma(k)} - c \\ &= \left(\sum_{i=1}^{n} A_{i,\sigma(i)} \right) - c \\ &= S - c \end{aligned}$$

Entonces si M_1 y M_2 son matchings y denotamos sus sumas por $S_1, \widetilde{S}_1, S_2, \widetilde{S}_2$ respectivamente tenesmos que:

$$\widetilde{S}_1 \subseteq \widetilde{S}_2 \Leftrightarrow S_1 - x \leqslant S_2 - c \Leftrightarrow S_1 \leqslant S_2$$

Complejidad de los algoritmos de matchings:

1. Algoritmo de hallar matching maximal:

   - Matching inicial, revisar cada fila es $O(n)$, como hay n filas entonces es $O(n^2)$.

   - Extender el matching en un lado, es le mismo análisis entonces es $O(n^2)$.

   - Hay $O(n)$ extensiones de matching.

   Entonces la complejidad total es $O(n^2) * O(n) = O(n^3)$.

   Si se hace con Dinitz se puede ver que es $O(n^{2,5})$ (si o si leer el network).

2. Minimizar el maximo:

   - Ordenar las n^2 entradas, es $O(n^2 log(n^2)) = O(n^2 log(n))$.

   - Hay que hacer (usando busqueda binaria). Entonces es $O(log(n^2))$ el algoritmo de hallar matching maximal.

   Entonces la complejidad total es $O(n^3 log(n))$ o $O(n^{2,5} log(n))$ si lo hicieramos con Dinitz.

3. Minimizar la suma (algoritmo de Kuhn-Munkres o hungaro):

   Lemma 4 *Luego de que el algoritmo realiza un cambio de matriz asociado a un S ($|S| > \Gamma(S)|$), si se sigue corriendo el algoritmo a partir del matching parcial que se tenia, entonces crece el matching o crece S.*

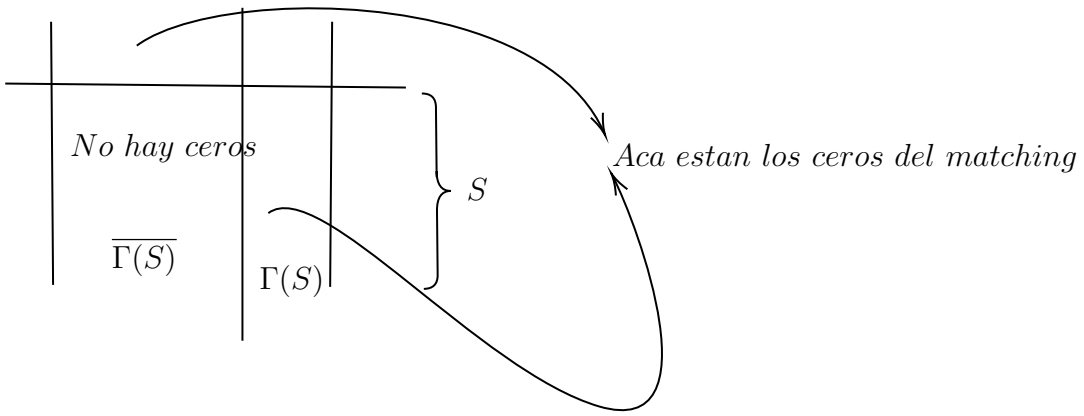
   Prueba:

   Al cambiar la matriz, sumamos $m = \text{mín}\{S \times \Gamma(S)\}$ y restamos m de S, sumamos m a $\Gamma(S)$.

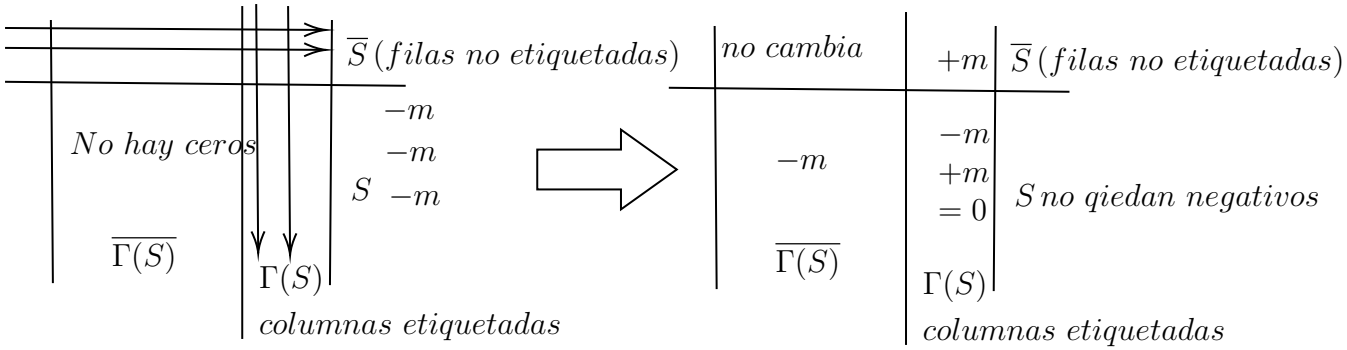
   Primero repasemos un poco:

   1) Qué constante restar/sumar?
   2) Queda una matriz de entradas no negativas?

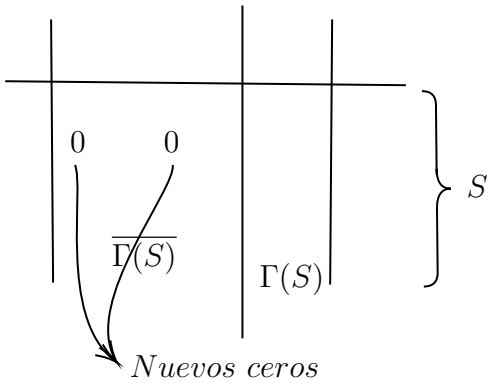
   Queremos al menos un nuevo cero, pero que no queden entradas negativas, inicialmente tenemos:



   Como $\Gamma(S)$ = vecinos de S, en el conjunto $\overline{\Gamma(S)}$ no hay ceros, en la fila de S, es decir, en $S \times \Gamma(S)$ no hay ceros. Entonces, al sumar m y restar los m en S (filas) y sumamos m en $\Gamma(S)$ (columnas), como se muestra acontinuación:



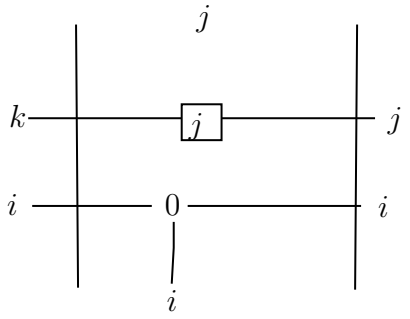
Entonces, en el algoritmo al buscar nuevos ceros los tendremos en la siguiente región de la matriz:



Cualquier nuevo cero estará en $S \times \overline{\Gamma(S)}$. Supongamos que está en la fila i, columna j, entonces $i \in S$ y $j \in \overline{\Gamma(S)}$ (la columna j no tiene etiqueta). Ese nuevo cero hará que etiquetemos la columna j con la etiquete i. Al chequear si la columna j esta libre o forma parte del matching, tenemos:

- Si esta libre, entonces extendemos el matching por lo que crece.
- Si forma parte del matching, existe una fila k tal que (k, j) es parte del matching.

  En particular, j es vecino de k. Entonces $k \notin S$, pues $j \notin \Gamma(S)$ pero el algoritmo al chequear j va a agregar a k al nuevo S por lo que S crece. Como lo muesta la siguiente imagen.



Corolario 6 *Puede haber a lo sumo $O(n)$ cambios de matriz. Antes de extender el matching en un lado (pues S puede crecer a lo sumo $O(n)$ veces).*

IV.1. Complejidad del algoritmo hungaro

Teorema 12 *La complejidad del algoritmo Hungaro como lo vimos en clase es* $O(n^4)$.

Prueba: Resta el *min* de una fila es $O(n)$, entonces restar *min* de cada columna es $O(n^2)$. Mismo análisis para restar *min* a cada columna es $O(n^2)$.

Hallar matching inicial es $O(n^2)$. Este matching se puede extender a los umo $O(n)$ veces. Por lo tanto:

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

Para extender el matching, hay que revisar m filas buscando ceros, entonces cada busqueda es $O(n)$.

Luego revisar de revisar a lo sumo n filas, si o si extendemos el matching pero en el medio quezas 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 le 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 $m = \text{mín}\{S \times \Gamma(S)\}$ (esto es $O(|S \times \Gamma(S)|) = O(n^2)$).
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 13 *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. Debemo 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:

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

donde $M_y = \text{mín}_{x \in S}\left\{C_{x,y}\right\}$, siempre hay que tenerlo precalculado. Precalcular los M_y demanda $O(n)$ pero los podemos hacer cuanod escanemos una fila al buscar ceto, 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 $SC(x)$ 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 $O(n)$.

V. Códigos de corrección de errores

Un código (binario de bloque) es un subconjunto de $\{0,1\}^n$ para algún n fijo. Que sea en bloque quiere decir que todas las palabras tienen la misma longitud.

Suponemos que el medio de transmisión puede cambiar bits pero no agregar ni sustraer. También suponemos que la probabilidad de cambiar $0 \to 1$ es igual a la de $1 \to 0$ es la misma para todos los bits, y e independiente entre los bits.

Si a esa probabilidad la llamamos p, entonce $0 < p < \frac{1}{2}$. Y la probabilidad de t errores es p^t.

Definición 9 *La distancia de hamming entre dos palabras $x, y \in \{0,1\}^n$ es:*

$$d_H(x,y) = \#\textit{de bits de diferencia entre } x \textit{ e } y$$

Propiedad: Como d_H es una distancia, valen las siguientes propiedades:

A) $d_H(x,y) = d_H(y,x)$

B) $d_H(x,y) \geqslant 0$

C) $d_H(x,y) = 0 \iff x = y$

D) Desigualdad triangular: $d_H(x,y) \leqslant d_H(x,z) + d_H(z,y)$

V.1. Código de corrección de errores:

Definición 10 *Sea* $\delta = \delta(c) = \min\{d_H(x,y) : x, y \in C, x \neq y\}$

Definición 11 *Un código* C *detecta* r *errores.*
Si $D_r(x) \cap C = \{x\}, \forall x \in C$ *(detecta si hubo por lo menos un error, pero no sabe cuantos.). Donde* $D_r(x) = \{y \in \{0,1\}^n : d_H(x,y) \leqslant r\}$. C *corrige* t *errores si* $D_t(x) \cap D_y(y) = \emptyset$, $\forall x, y \in C, x \neq y$

Teorema 14 *Sea* C *un código tal que* R *es la mayor capacidad de detección de errores de* C*, es decir,* C *detecta* R *errores pero no* $R+1$ *y* Q *la mayor capacidad de corrección de errores de* C*, es decir ,* C *corrige* Q *errores y no corrge* $Q+1$ *errores. Entonces:* $R = \delta - 1$ *y* $Q = \lfloor \frac{\delta-1}{2} \rfloor$

Prueba:

Sea $z \in D_{\delta-1}(x) \cap C$ para algún $x \in C$. Entonces como $z \in D_{\delta-1}(x)$ podemos decir que $dH(x,z) \leqslant \delta - 1$ por definición de disco.

Como $x, z \in C$ entonces $x = z$ o $\delta \leqslant d_H(x,z)$. Pero es un absurdo. Conclusión $x = z$ por lo tanto $D_{\delta-1}(x) \cap C = \{x\}$ y C detecta $\delta - 1$ errores.

C no detecta δ errores. Pues sean $x, y \in C$ donde $x \neq y$, $d_H(x,y) = \delta$, entonces $y \in D_\delta(x)$ además $y \in C$ entonces $D_\delta(x) \cap C \neq \{x\}$.

Veamos ahora que C corrige $t = \lfloor \frac{t-1}{2} \rfloor$ errores.

Sean $x, y \in C$, con $x \neq y$ supongamos que $D_t(x) \cap D_t(y) \neq \emptyset$. Entonces $\exists x \in D_t(x) \cap D_t(y)$, por lo que $d_H(x,z) \leqslant t$ y $d_H(z,y) \leqslant t$.

Por desigualdad triangular $d_H(x,y) \leqslant d_H(x,z) + d_H(z,y) \leqslant 2x \leqslant \delta - 1$. Pero $x \neq y$ entonces $\delta \leqslant \delta_H(x,y) \Rightarrow \delta \leqslant \delta - 1$, que es un absurdo.

C no corrige $t+1$ errores: Sean $x, y \in C, x \neq y : d_H(x,y) = \delta$. Entonces y difiere de x en δ lugares. Sea z talque z sea distinto de x en $t+1$ de esos δ lugares.

Entonces por construcción:

$$d_H(x,z) = t + 1 \Rightarrow z \in D_{t+1}(x)$$

Si probamos que $z \in D_{t+1}(y)$ tendremos $D_{t+1}(x) \cap D_{t+1}(y) \neq \emptyset$. Y habremos terminado.

Caso δ impar: $t = \frac{\delta-1}{2}$.

$$d_H(z,y) = \delta - (t+1) = \delta - t - 1 = \delta - \frac{\delta-1}{2} - 1 =$$
$$= \frac{\delta}{2} + \frac{1}{2} - 1 = \frac{\delta}{2} - \frac{1}{2} = \frac{\delta-1}{2} = t$$

Entonces $z \in D_t(y) \subsetneq D_{t+1}(y)$.

Caso Par: $\lfloor \frac{\delta-1}{2} \rfloor = \frac{\delta-2}{2} = \frac{\delta}{2} - 1 \Rightarrow t + 1 = \frac{\delta}{2}$

$$d_H(x,y) = \delta - (t+1)$$
$$= \delta - \frac{\delta}{2} = \frac{\delta}{2} = t + 1$$

Entonces $z \in D_{t+1}(y)$. $\square$

V.1.1. Cota de Hamming

Teorema 15 *(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 $|A| = \sum_{x \in C} |D_t(x)|$.

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

Definición 12 *Un código de longitud n con $\delta = \delta(C)$ es perfento si*

$$|C| = \frac{2^n}{1 + n + \binom{n}{2} + \cdots + \binom{n}{t}} \text{ con } t = \lfloor \frac{\delta - 1}{2} \rfloor$$

V.2. Códigos lineales:

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

Definición 13 *El peso de Hamming es* $||x|| = d_H(x,0) = \#$ *de 1's de* x.

Propiedad C lineal $\Rightarrow \delta(C) = \text{mín}\,\{||x|| : x \neq 0, x \in C\}$. es lineal en el orden de palabras, sino es lineal $n \times n$ comparaciones, donde n es el número de palabras.

Prueba:

Sea $m = \text{mín}\,\{||x|| : x \in C, x \neq 0\}$, $\delta = \delta(C)$. Sea $x, y \in C, x \neq y$ con $d_H(x,y) = \delta$. Entonces $x + y \in C$ por ser código lineal. Como $x \neq y$ entonces $x + y \neq 0$, esto implica que $||x + y|| \geqslant m$.

Pero $||x + y|| = d(x + y, 0) = d(x, y) = \delta$. entonces $\delta \geqslant m$.

Viceversa: Sea $x \in C, x \neq 0$ con $||x|| = m$, entonces $m = ||x|| = d(x,0) \geqslant \delta$ (la desigualdad sale por definición).

Cosas a tener en cuenta:

C lineal es un subespacio vectorial de $\{0,1\}^n$. Todo espacio vectorial tiene dimención k y al menos una base. La base es vista como un conjunto generador y además es LI es decir que $c_1\alpha_1 + \ldots + c_r\alpha_r = 0 \Rightarrow c_1 = \ldots = c_r = 0$.

Una matriz generadora de un código lineal C es una matriz cuyas filas son base de C. Como las filas deben ser base, cualquier matriz generadora debe ser $k \times n$.

Observación: Si $k = Dim(C)$ entonces C es isomorfismo a $\{0,1\}^k$, entonces $|C| = 2^k$.

Definición 14 *Una Matriz de chequeo: es una matriz de chequeo de un código* C *si* $C = Nu(H)$.

$$C = Nu(H) = \left\{y \in \{0,1\}^n : Hy^t = 0\right\}$$

(es para saber si lo que recibimos esta en el código, entonces se multiplica por la matriz y si está el resultado es 0).

Teorema 16 *Si* H *es matriz de chequeo de* C*, entonces:*

$$\begin{aligned}\delta(C) &= \text{mín}\ \{\textit{número de columnas de } H \textit{ que son LD}\} \\ &= \text{mín}\,\{r : \exists r \textit{ columnas LD de } H\}\end{aligned}$$

Prueba:

Sea $m = \text{mín}\,\{r : \exists r \,\text{columnas LD de}\, H\}$. Denotaremos la $j-$ésima columna de H por $H^{(j)}$. Por definición de m existe $j_1, \ldots j_r$ tal que $H^{(j_1)}, \ldots, H^{(j_r)}$ son LD.

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

$$c_1H^{(j_1)} + \cdots + c_mH^{(j_r)} = 0$$

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 $x = c_1 e_{j_1} + c_2 e_{j_2} + \ldots + c_r e_{j_r}$. Entonces:

$$\begin{aligned} Hx^t &= H(c_1 e_{j_1}^t + c_2 e_{j_2}^t + \ldots + c_r e_{j_r}^t) \\ &= c_1 H e_{j_1}^t + c_2 H e_{j_2}^t + \ldots + c_r H e_{j_r}^t = 0 \\ &\Rightarrow Hx^t = 0 \Rightarrow x \in C \end{aligned}$$

Como $x \neq 0$ los c_i no son todos ceros. Entonces $\delta(C) = ||x||$. Pero en realidad los c_i son todos unos, porque si alguno fuese cero, tendria una cantidad menor de columnas LD de H.

Entonces, $||x|| = r : \delta \leqslant m$

Para el otro lado. Sea $x \neq 0 : \delta(C) = ||x||$, entonces:

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

□

Corolario 7 *Si H no tiene la columna 0 ni columnas repetidas y $c = Nu(H)$, entonces $\delta(C) \geqslant 3$, i.e., C corrige al menos un error.*

Prueba: Como $H^{(j)} \neq 0$ para todo j, entonces mín $\{r : \exists r \text{ columnas LD de } H\} \geqslant 2$. Supongamos que fuese 2, entonces existen j_1, j_2 tales que $H^{(j_1)} + H^{(j_2)} = 0$, $c_1 = c_2 = 1$. Entonces como estamos en $\mathbb{Z}^2$ y sumar es igual a restar tenemos que $H^{(j_1)} = H^{(j_2)}$, lo cual es absurdo. □.

Observación: Como calcular $Dim(C)$ a partir de n. Supongamos que las filas de H son LI. Usaremos el teorema de que si $T : V \to W$ es una transformación lineal entonces $Dim(V) = Dim(Nu(T)) + Dim(Im(T)) = k + Dim(Im(T))$.

Pero

$$\begin{aligned} Dim(Im(T)) &= Dim(\text{espacio columna de } H) \\ &= \text{rango columna} \\ &= \text{rango fila, por teorema del rango} \\ &= \#\text{ de filas LI de } H \qquad\qquad \Rightarrow k = n - \#\text{ de filas LI de } H \end{aligned}$$

Propiedad: Si H es de la forma $[A|I]$ y $C = Nu(H)$, entonces $G = [I|A^t]$ es generadora de C y viceversa si G es de la forma $[I|A^t]$ y $C = Im(G)$, entonces $H = [A|I]$ es matriz de chequeo de C.

Cabe resaltar que si estuvieramos en $\mathbb{Z}^{3\,o\,4}$, entonces $G = [I| - A^t]$

Prueba:

Sea $w = \{uG : w \in \{0,1\}^n\}$ con $G = [I|A^t]$. Entonces queremos ver $w = C$ es decir que $w \subset C$ y $C \subset w$.

Sea $x \in w$ entonces $\exists u : x = uG = u[I|A^t] = u + uA^t$. Entonces

$$\begin{aligned} Hx^t &= H(uG)^t \\ &= HG^t u^t \\ &= [A|I] \left[\frac{I}{A}\right] u^t \\ &= (A + A)u^t = 0 \\ &\Rightarrow x \in C \Rightarrow w \subset C \\ &\text{Pero } Dim(w) = k \text{ y } Dim(C) = k \Rightarrow w = C \end{aligned}$$

Cabe resaltar que $A + A = 0$ solo por estar en $\mathbb{Z}^2$, si estamos en otra dimensión es importante el $-$ que resaltamos más atrás.

Propiedad: Los códigos de Hamming son perfectos.

Prueba: Como no tienen calumnas repetidas ni columnas todas de ceros, sabemos que $\delta(C) \geqslant 3$. Pero siempre existe e_1^t, e_2^t y $(e_1 + e_2)^t$ por lo tanto $\delta(C) = 3$ (LD).

C es pefecto en ese tado sii $|C| = \frac{2^n}{1+n}$, pero H tiene todos las columnas no nulas si son r filas, con $2^r - 1$ columnas nuevas (resto 1 porque no puedo poner la de todos ceros). Entonces $n = 2^r - 1$.

Entonces:

$$\frac{2^n}{1+n} = \frac{2^{2^r-1}}{1+2^r-1} = \frac{2^{2^r-1}}{2^r} = 2^{2^r-r-1}$$

Pero $|C| = 2^k$ entonces basta ver que $k = 2^r - r - 1$. Pero:

$$\begin{aligned} k &= \#\,\text{columas} - \#\,\text{filas de H} \\ &= 2^r - 1 - r \end{aligned}$$

V.3. Códigos Ciclicos:

Cambio de notación: a la palabra $w_1, \ldots, w_n$ la denotamos como $w_0, w_1, \ldots, w_{n-1}$ pues identificaremo la palabra $w = w_0, \ldots, w_{n-1}$ con el polinomio $w(x) = w_0 + w_1 x + \ldots + w_{n-1}x^{n-1}$ en $\mathbb{Z}_2(x)$

Importante: no confundir función polinoimal con polinomio. Ejemplo $1 + x \neq 1 + x^2$ como polinomios, pero como funciones sobre $\mathbb{Z}_2$ si son iguales (funciones que mandan a 0 y 1).

Entonces queremos multiplicar palabras de longitud n podriamos multiplicar los polinomios asociados pero queremos que el resultado tenga grado $< n$ lo que se hace es tomar la multiplicación modulo algún polinomio de grado n, por lo que tomaremos $1 + x^n$.

Definición 15 $v(x) \bigodot w(x) = v(x)w(x) \mod (1 + x^n)$

Ejemplo:

$$\begin{aligned}(1011) \odot (0110) =? \\ &= 1 + x^2 + x^3 \odot x + x^2 = \\ &= (1 + x^2 + x^3)(x + x^2) \mod (1 + x^4) \\ &= x + x^2 + x^3 + \cancel{x^4} + \cancel{x^4} + x^5 \mod (1 + x^4) \\ &= x + x^2 + x^3 + x^5 \mod (1 + x^4)\end{aligned}$$

Observación:

$$\begin{aligned}&(1 + x^n) \mod (1 + x^n) = 0 \\ &1 \mod (1 + x^n) + x^n \mod (1 + x^n) = 0 \\ &x^n \mod 1 + x^n = 1 \\ &\Rightarrow x^n \mod (1 + x^n) = 1 \\ &\Rightarrow x^{n+1} \mod (1 + x^n) = x \\ &\Rightarrow x^{n+2} \mod (1 + x^n) = x^2\end{aligned}$$

Entonces:

$$\begin{aligned}&x + x^2 + x^3 + x^5 \mod (1 + x^4) \\ &x + x^2 + x^3 + x = x^2 + x^3 = (0011)\end{aligned}$$

Definición 16 *$Rot(w) = Rot(w_0 \dots w_{n-1}) = w_{n-1}w_0w_1 \dots w_{n-2}$*

Definición 17 *(Propiedad Obvia) $Rot(w) = x \odot w(x)$*

Prueba:

$$\begin{aligned}x \odot w(x) &= xw(x) \mod (1 + x^n) \\ &= x(w_0 + w_1x + \dots + x_{n-1}x^n) \mod (1 + x^n) \\ &= w_0x + w_1x^2 + \dots + w_{n-1}x^{n+1} \mod (1 + x^n) \\ &= w_0x + w_1x^2 + \dots + w_{n-1} \\ &= w_{n-1} + w_0x + w_1x^2 + \dots + w_{n-2}x^{n-1} \\ &= Rot(w)4\end{aligned}$$

□

Definición 18 *Un código C es ciclico si es lineal e invatiante por rotaciones, es decir, $rot(w) \in C, \forall w \in C$*

Definición 19 *(Propiedad) Si C es ciclico, existe un único polinomio no nulo en C de grado minimo.*

Prueba:

Supongamos que hay dos polinomios no nulos, $p(x)$ y $q(x)$ de grado minimo. Sea $t = gr(p) = gr(q)$. Entonces:

$$\begin{aligned} p(x) &= x^t + \text{ Cosas de grado } \leqslant t-1 \\ q(x) &= x^t + \text{ Cosas de grado } \leqslant t-1 \\ &\Rightarrow \\ p(x)+q(x) &= x^t + x^t + \text{ Cosas de grado } \leqslant t-1 \\ p(x)+q(x) &= \text{ Cosas de grado } \leqslant t-1 \end{aligned}$$

pero t era el menor grado de un polinomio no nulo en C y $p(x)+q(x) \in C$, entonces como son lineales $p(x)+q(x) \in C$ pero $gr(p+q) \leqslant t-1$ esto solo pasa si $p(x)+q(x) = 0 \Rightarrow p = q$ lo cual es un absurdo. $\square$

Definición 20 *Al único polinomio no nulo de grado minimo de un código cilico C se le llama, el polinomio generador y se lo denota como $g(x)$.*

Corolario 8 *(de la 17) Sea C ciclico y $w(x)$ cualquier palabra en C y $v(x)$ cualquier polinomio de cualquier grado entonces:*

$$w(x) \odot v(x) \in C$$

(absorvente)

Prueba:

Sea $v(x) = \sum_{i=0}^{f} v_i x^i$ entonces:

$$v(x) \odot w(x) = \sum_{i=0}^{f} v_i x^i \odot w(x)$$

Como C es lineal, si probamos que $x^i \odot w(x) \in C \, \forall i$. Lo de arriba será una suma de cosas de C por lo tanto estará en C. Pero $x \odot w(x) = rot(w)$. Por lo tanto, $x^i \odot w(x) = rot^i(w) \in C$. $\square$

Teorema 17 *(Fundamental de código ciclico) Sea C un código ciclico de longitud n con generador $g(x)$ entonces:*

1) *$C = \{p(x) \in \mathbb{Z}_2(x) : gr(p) < n \wedge g(x) | p(x)\}$ por esto se dice que C es generador (son los multiplicos de $g(x)$ de menor grado).*
2) *$C = \{v(x) \odot g(x) : v \in \mathbb{Z}_2(x)\}$ son los multiplos de g modulares.*
3) *Si $k = Dim(C)$ entonces $gr(g) = n - k$.*
4) *$g(x) | (1 + x^n)$.*
5) *Si $g(x) = g_0 + g_1 x + \ldots +$ entonces $g_0 = 1$.*

Prueba:

Sea $C_1 = \{p(x) \in \mathbb{Z}_2(x) : gr(p) < n \wedge g(x)|p(x)\}$ y $C_2 = \{v(x) \odot g(x) : v \in \mathbb{Z}_2(x)\}$. Tenemos que $C_1 \subseteq C_2$, pues $p(x) \in C_1$ 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 $C_2 \subset C$, es obvio por 17

$C \subseteq C_1$, 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:

$$\begin{aligned} p(x) &= g(x)q(x) + r(x) \; gr(r) < gr(g) \\ &\quad \text{Tomando modulo y usando } gr(p) < n \\ &= p(x) \mod (1+x^n) \\ &= g(x)q(x) + r(x) \mod (1+x^n) \\ &= g(x)q(x) \mod (1+x^n) + \underbrace{r(x) \mod (1+x^n)}_{gr(r)<gr(g)<n} \\ &= g(x) \odot q(x) + r(x) \end{aligned}$$

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:

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

Por lo tanto $r = 0$.

5)

Si $1 + x^n = g(x)q(x)$ entonces $1 = g_0 q_0 \Rightarrow g_0 = 1$.

Método 2 de códificación:

$$\forall p, \ (p \mod g) + p \text{ es multiplo de } g$$

pues $((p \mod g) + p) \mod g = p \mod g + p \mod g = 0$ (lo divide).

Usaremos este truco pero con cuidado,

$$\begin{aligned} \{0,1\}^k &\to \{0,1\}^n \\ q &\to q + (q \mod g) \end{aligned}$$

No funciona porque no es inyectiva. Primero hay que asegurarse que sea 1 a 1. Lo que hacemos es:

$$q \to (x^{n-k} q \mod g) + x^{n-k} q$$

Ejemplo: $g(x) = 1 + x^2 + x^3$, $n = 7$
$q(x) = 1100 = 1 + x$ entonces $n - k = 3$ y $k = 4$.

$$\begin{aligned} x^{n-k} q(x) &= x^3(1 + x) = x^3 + x^4 \\ &\Rightarrow x^3 + x^4 \mod g \end{aligned}$$

Idea de $g \mod g = 0$

$$\begin{aligned} &1 + x^2 + x^3 \mod g = 0 \\ &1 + x^2 \mod g + x^3 \mod g = 0 \\ &\Rightarrow x^3 \mod g = 1 + x^3 \\ &\text{Otras potencias} \\ &x^4 \mod g = x(1 + x^2) \mod g \\ &= x + x^3 \mod g \\ &= x + 1 + x^2 = 1 + x + x^2 \\ &\text{es como} \\ &x^4 \mod g \\ &x(x^3 \mod g) \mod g \\ &x(1 + x^2) \mod g \end{aligned}$$

Entonces:

$$\begin{aligned} q = 1100 \to (x^3 + x^4) \mod x^3 + x^4 \\ = \cancel{1 + x^2 + 1} + x + \cancel{x^2} + x^3 + x^4 \\ = x + x^3 + x^4 = 010\underbrace{1100}_{q} \end{aligned}$$

Matriz asociada: Corresponde a cada fila la base canonica de $\{0,1\}^k$.

$$\begin{bmatrix} 100 & \to & 1 \\ 101 & \to & x \\ 001 & \to & x^2 \\ \vdots & \vdots & \vdots \\ & \to & x^{k-1} \end{bmatrix} \Rightarrow \begin{bmatrix} (x^{n-k} & \mod g) + x^{n-k} \\ (x^{n-k-1} & \mod g) + x^{n-k-1} \\ & \vdots \\ (x^{n-1} & \mod g) + x^{n-1} \end{bmatrix}$$

En nuestro caso:

$$\begin{bmatrix} (x^3 & \mod g) + x^3 \\ (x^4 & \mod g) + x^4 \\ (x^5 & \mod g) + x^5 \\ (x^6 & \mod g) + x^6 \end{bmatrix} = \begin{bmatrix} x^3 & \mod g = 1 + x^2 \\ x^4 & \mod g = 1 + x + x^2 \\ x^5 & \mod g = 1 + x(*) \\ x^6 & \mod g = x + x^2 \end{bmatrix}$$

(*) $x(x^4 \mod g) \mod g = x(1 + x + x^2) \mod g = x + x^2 + x^3 \mod g = x + x^2 + 1 + x^2$
Entonces:

$$G = \begin{bmatrix} 1 + x^2 + x^3 \\ 1 + x + x^2 + x^4 \\ 1 + x + x^5 \\ x + x^2 + x^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Se puede pedir que verifiquen, verifique que $g(x)|1 + x^n$ debe ser

$$\begin{aligned} (1 + x^n) \mod g = 0 \\ x^n \mod g = 1 \end{aligned}$$

Multiplicar el $(x^{n-1} \mod g)$ que sale de la matriz g. Entonces nuestro ejemplo:

$$\begin{aligned} x^6 \mod g = x + x^2 \\ x^7 \mod g = (x^2 + x^3) \mod g \\ = x^2 + 1 + x^2 = 1 \end{aligned}$$

Polinomio chequeador:

$$g|1 + x^n \Rightarrow \exists h(x) : 1 + x^n = g(x)h(x)$$

donde $h(x)$ es el polinomio chequeador. ie. $h(x) = \frac{1+x^n}{g(x)}$. Por qué es chequeador?

Sea $p \in C \Rightarrow p = qg$ para algún q del grado apropiado. Entonces:

$$ph = qgh = q(1 + x^n) \Rightarrow ph \mod (1 + x^n) = 0$$

es decir, $p(x) \odot h(x) = 0$ (como en la matriz de chequeo).

Matriz de chequeo:

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow H = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right]$$

F_1 es 1, $F_2 = x$, $F_3 = x^2$, $F_4 = x^3$, $F_5 = x^4$, $F_6 = x^5$, $F_7 = x^6$. todos $\mod g$.

Error Traping: Eficiente pero no siempre funciona es eficiente mientras los errores esten en una ventana de $n - k$ si estan dispersos no es eficiente.

Supongamos que C es ciclico de longitud n con polinomio generadot y t errores. Supongamos que mandamos $v \in C$ y llega $w = v + e$ con e de peso t, es decir, $||e|| \leqslant t$.

Si tomamos $\mod g$

$$(w \mod g) = v + e \mod g = \underbrace{v \mod g}_{=0, \text{ pues } v \in C} + e \mod g$$

$$\Rightarrow w \mod g = e \mod g$$

Pero esto mismo vale para las rotaciones, en general:

$$rot^i(w) = rot^i(v) + rot^i(e)$$

$$rot^i(w) \mod g = \underbrace{rot^i(v) \mod g}_{=0} + rot^i(e) \mod g$$

$$\Rightarrow rot^i(w) \mod g = rot^i(e) \mod g$$

Y obviamente $||rot^i(e)|| = ||e|| = t$.

Si para algún i, $gr(rot^i(e)) \leqslant gr(g)$ entonces $rot^i(e) \mod g = rot^i(e)$. Entonces:

$$gr(rot^i(e)) < gr(g)$$

Equivale a decir que los i de e estan en alguna ventana de longitud $n - k = gr(g)$

Si eso pasa, como vimos:

$$rot^i(w) \mod g = rot^i(e) \mod g$$
$$\Rightarrow r = rot^{-i}(rot^i(w) \mod g)$$

En termino de polinomios el algoritmo queda:

$S_0 = w \mod g$ $(elsindrome)$
$si\ i \geqslant 1$
$S_1 = (xS_{i-1} \mod g)$
hasta que$|S_t| \leqslant t \rightarrow error = x^{n-1}S_i \mod (1 + x^n)$

Ejemplo: $g(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}, n = 23$, corrige $t = 3$ errores.

Se recibe $w = 1 + x + x^8 + x^9 + x^{12} + x^{13}$. Hallar v que es la palabra enviada. Entonces $k = 23 - 11 = 12$ dimensión.

$$S = w \mod g$$
$$S_0 = 1 + x + x^8 + x^9 + (x^{12} \mod g) + (x^{13} \mod g)$$

Calculos auxiliares:

$$\begin{aligned}
x^{11} \mod g &= 1 + x^2 + x^4 + x^5 + x^6 + x^{10} \\
x^{12} \mod g &= x + x^3 + x^5 + x^6 + x^7 + 1 + x^2 + x^4 + x^5 + x^6 + x^{10} \\
&= 1 + x + x^2 + x^3 + x^7 + x^{10} \\
x^{13} \mod g &= x + x^2 + x^3 + x^8 + 1 + x^2 + x^4 + x^5 + x^6 + x^{10} \\
&= 1 + x + x^3 + x^6 + x^8 + x^{10}
\end{aligned}$$

Entonces:

$$\begin{aligned}
S &= w \mod g \\
S_0 &= 1 + x + x^8 + x^9 + (x^{12} \mod g) + (x^{13} \mod g) \\
&= 1 + x + x^8 + x^9 + (1 + x + x^2 + x^3 + x^7 + x^{10}) + (1 + x + x^3 + x^6 + x^8 + x^{10}) \\
&= 1 + x + x^2 + x^4 + x^6 + x^7 x^9 \text{ es el sindrome}
\end{aligned}$$

Como $|S_0| = 7 > 3$ tengo que calcular S_1, entonces:

$$\begin{aligned}
S_1 &= xS_0 \mod g \\
&= x(1 + x + x^2 + x^4 + x^6 + x^7 + x^9) \mod g \\
&= x + x^2 + x^3 + x^5 + x^7 + x^8 + x^{10}
\end{aligned}$$

$|S_1| = 7 > 3$

$$\begin{aligned}
S_2 &= xS_1 \mod g \\
&= x(x + x^2 + x^3 + x^5 + x^7 + x^8 + x^{10}) \mod g \\
&= x^2 + x^3 + x^4 + x^6 + x^8 + x^9 + (x^{11} \mod g) \\
&= x^2 + x^3 + x^4 + x^6 + x^8 + x^9 + (1 + x^2 + x^4 + x^5 + x^6 + x^{10}) \\
&= 1 + x^3 + x^5 + x^8 + x^9 + x^{10}
\end{aligned}$$

$|S_2| = 6 > 3$

$$\begin{aligned} S_3 &= xS_2 \mod g \\ &= x(1 + x^3 + x^5 + x^8 + x^9 + x^{10}) \mod g \\ &= x + x^4 + x^6 + x^9 + x^{10} + (x^{11} \mod g) \\ &= x + x^4 + x^6 + x^9 + x^{10} + (1 + x^2 + x^4 + x^5 + x^6 + x^{10}) \\ &= 1 + x + x^2 + x^5 + x^9 \end{aligned}$$

$|S_3| = 5 > 3$

$$\begin{aligned} S_4 &= xS_3 \mod g \\ &= x(1 + x + x^2 + x^5 + x^9) \mod g \\ &= x + x^2 + x^3 + x^6 + x^{10} \end{aligned}$$

$$\begin{aligned} S_5 &= xS_4 \mod g \\ &= x(x + x^2 + x^3 + x^6 + x^{10}) \mod g \\ &= x^2 + x^3 + x^4 + x^7 + (x^{11} \mod g) \\ &= x^2 + x^3 + x^4 + x^7 + (1 + x^2 + x^4 + x^5 + x^6 + x^{10}) \\ &= 1 + x^3 + x^5 + x^6 + x^7 + x^{10} \end{aligned}$$

$|S_5| = 6 > 3$

$$\begin{aligned} S_6 &= xS_5 \mod g \\ &= x(1 + x^3 + x^5 + x^6 + x^7 + x^{10}) \mod g \\ &= x + x^4 + x^6 + x^7 + x^8 + (x^{11} \mod g) \\ &= x + x^4 + x^6 + x^7 + x^8 + (1 + x^2 + x^4 + x^5 + x^6 + x^{10}) \\ &= 1 + x + x^2 + x^5 + x^7 + x^8 + x^9 \end{aligned}$$

$|S_6| = 7 > 3$

$$\begin{aligned} S_7 &= xS_6 \mod g \\ &= x(1 + x + x^2 + x^5 + x^7 + x^8 + x^9) \mod g \\ &= x + x^2 + x^3 + x^6 + x^8 + x^9 + (x^{11} \mod g) \\ &= x + x^2 + x^3 + x^6 + x^8 + x^9 + (1 + x^2 + x^4 + x^5 + x^6 + x^{10}) \\ &= 1 + x + x^3 + x^4 + x^5 + x^8 + x^9 + x^{10} \end{aligned}$$

$|S_7| = 9 > 3$

$$\begin{aligned} S_8 &= xS_7 \mod g \\ &= x(1 + x + x^3 + x^4 + x^5 + x^8 + x^9 + x^{10}) \mod g \\ &= x + x^2 + x^4 + x^5 + x^6 + x^9 + x^{10} + (x^{11} \mod g) \\ &= x + x^2 + x^4 + x^5 + x^6 + x^9 + x^{10} + (1 + x^2 + x^4 + x^5 + x^6 + x^{10}) \\ &= 1 + x + x^9 \end{aligned}$$

$|S_8| = 3$ ahora si.

Entonces, $e = x^{23-8}S_8 \mod (1 + x^{23})$, donde x^{23-8} es para rotarlo. Resolviendo:

$$\begin{aligned} &= x^{15}(1 + x + x^9) \mod (1 + x^{23}) \\ &= x^{15} + x^{16} + x^{24} \mod (1 + x^{23}) \\ &= x + x^{15}x + x^{16} \end{aligned}$$

$\therefore$

$$\begin{aligned} v &= w + (x + x^{15}x + x^{16}) \\ &= 1 + x^8 + x^9 + x^{12} + x^{13} + x^{15} + x^{16} \end{aligned}$$

Decodificar:

$$\underbrace{v}_{\text{palabra enviada}} = \underbrace{10000000110}_{\text{bits chequea}} | \underbrace{0110110000000}_{\text{Decodificar}}$$

$\therefore$ La palabra enviada es 0110110000000, en polinomio $x + x^2 + x^4 + x^5$

VI. P-NP

Son basicamente problemas de decisión (no hay calculo). Es decir, problemas con solo dos respuestas posibles, usualmente SI o NO.

Ejemplo:

- $\underbrace{\text{Dada } A_{n\times n}}_{\text{instancia}}$, es invertible? La solución seria calcular el determinante.
- Dado N network, existe un flujo f tal que $v(f) \leq 100$?. La solución es calcular el flujo máximal.
- Otro que vimos es $k-$color. Dado un grafo G, es $\chi(G) \leq k$? $2-$color es polinomial.

La clase P es la clase de problemas de decision que tienen un algoritmo polinomial que los resuelve. Ejemplo, $2-$color $\in P$, $3-$color no se sabe.

NP es **no es no polinomial**. NP viene de *non deterministic polynomial* (polinomial no deterministico). Es decir, que puede tomar decisiones al hazar al realizar un algoritmo. Ejemplo, $100-$color en algún orden de Greedy da $\chi(G)$, lo calculo y si da 100 tengo suerte y sino no. Las respuestas sobre el problema de decision acá son $NO\ SE$ y SI.

Entonces NP es la clase de problemas de decisión para los cuales existe un algoritmo no deterministico polinomial que lo resuelve para el SI. Un algoritmo no deterministico A **resuelve** para el SI un problema de decisión Π. Si las únicas respuestas de A son NO SE o SI y si para una instancia I de Π, $A(I) = SI$ entonces $\Pi(I) = SI$ y que si $\Pi(I) = SI$, entonces alguna elección NO.

Deterministica de A digo que SI. Si reemplazamos SI por NO la clase es $CO-$NP. Donde claramente $P \subseteq NP \cap CO-$NP.

La pregunta del millon es. $P = NP$?

Ejemplo de algunos problemas:

- Primo: Dado $n \in \mathbb{N}$. es n primo? Se mide en terminos de $log(n)$. Primo $\in CO - NP$ elegimos no deterministicamente, $a, b < n$ y chequeamos si $n = ab$ y si eso pasa, respondemos NO y sino NO SE.
- Primos $\in NP$ Algoritmos aleatorios que si dice SI el número es primo se probo que si. Fue en 1976 la criptografia y los algoritmos de las tarjetas de crédito se basan en esto, en caso de que $P = NP$ tendríamos algoritmos para resolver estos problemas en tiempo polinomial.
- Primos $\in P$ algoritmo con alta complejidad $O(log(n)^8)$
- Clique: Dado un grafo G y un número n. Tiene G un K_n adentro? Clique $\in NP$ y no se sabe si hay algún algoritmo polinomial que lo resuelve, es decir, no sa sabe si clique $\in P$.
- $n-$clique: Dado un grafo G, tiene G un K_n adentro? $n-$clique $\in NP$. Buscar todos los subconjuntos de cardinalidad n de G y chequear si alguno es completo es

$$\underbrace{O(n^2)}_{\text{por el chequeo}} + \underbrace{O(\binom{N}{n})}_{\#\text{ de subconjuntos}}$$

  $N = \#\text{vertices de}G$, para n fijo, es un polinomico en N.

Clase VP: son los problemas de decisión verificables polinomicamente con un certificado polinomial. $VP \to SI$.

Se pueden hacer pasajes, VP a NP elijo un certificado al azar. NP a VP las decisiones aleatorias las transformo en un certificado.

Reducción polinomial (Karp): es mucho más fina que la de Cook.

Dados problemas de decisión Π, τ diremos que Π se reduce polinomialmente a $\tau(\Pi \leq_p \tau)$ si existe un algoritmo polinomial (deterministico) A tal que cualquier instancia I de Π, $A(I)$ es una instancia de τ (tiene que haber una relación entre Π y $Pi(I)$) tal que

$$\Pi(I) = SI \Leftrightarrow \tau(A(I)) = SI$$

Esto crea una jerarquia de problemas los cuales son comparables.

SAT: *satisfactibility* Variable booleana $\to$ 1 o 0 como valores posibles.

Expresión booleana $\to$ función de variables booleanas.

Dadas expresiones boolenas $B_1, \ldots B_r$ la conjunción de ellas es $B_1 \wedge \ldots \wedge B_r$ y la disyunción es $B_1 \vee \ldots \vee B_r$.

Un literal es una variable o negación de la variable. Una expresión booleana esta en forma conjuntiva normal (CNP), si es conjunción de disjunciones de literales. Ejemplo:

$$(x \vee y) \wedge (\bar{x} \vee z \vee w) \wedge (y \vee \bar{z} \vee u) \ldots$$

Una expresión booleana es **satisfacible** si existe un asignacmiento de valores para las variablas que la valuen verdadera (no es cierto que sea siempre fácil).

Como en un $\vee$ solo alcanza que una variable booleana se satisfaga, pero como esta el *wedge* se tiene que satisfacer todo simultaneamente.

Teorema 18 *(Cook, 1972)*

$$\Pi \in NP \Rightarrow \Pi \leq_p SAT$$

No lo demostramos, es extramadamente dificil la demostración.

Dado un problema τ tal que $\Pi \in NP \Rightarrow \Pi \leq_p \tau$ se dice que es $NP-$HARD y si τ es $NP-$HARD y $\tau \in NP$ se dice que es $NP-$COMPLETO.

Por lo tanto Cook dice que SAT es $NP-$COMPLETO.

SAT polinomial la Reducción dice que todos los problemas son polinomial.

Corolario 9 $SAT \in P \Rightarrow P = NP$

Absolutamente si $SAT \notin P \Rightarrow P \neq NP$.

Karp también propuso 21 problemas equivalentes entre si. Entonces la idea es:

$$\Pi \leq_p SAT \leq_p \tau_1 \leq_p \tau_2$$

Definición 21 *(SAT) Dado una expresión booleana B en CNF, $SAT(B)$ es el problema de decidir si B es satisfacible.*

3−SAT es como SAT pero se requiere que cada disjunción tenga exactamente 3 literales.

VI.1. 3-SAT es NP-COMPLETO

Teorema 19 *(Karp, 1978) $3-SAT$ es $NP-COMPLETO$.*

Nosotros vamos a demostras 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 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 $\widetilde{B} = E_1 \wedge \ldots \wedge E_n$.

Con cada E_j definimos a partir de 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 $1's$.

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, $B(\overrightarrow{x})$.

Por lo que $\widetilde{B}$ satisfacible entonces existen vectores $\overrightarrow{a}$, $\overrightarrow{b}$ tales que $\widetilde{B}(\overrightarrow{a}, \overrightarrow{b}) = 1$. Vamos a demostrar que $B(\overrightarrow{a}) = 1$.

Es decir:

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

$\exists \overrightarrow{a} : B(\overrightarrow{a}) = 1 \Leftrightarrow \exists \overrightarrow{c}, \overrightarrow{b} : \widetilde{B}(\overrightarrow{c}, \overrightarrow{b}) = 1$. (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, entoces se da $B(\overrightarrow{a}) = 0$.

Como $B = D_1 \wedge \ldots \wedge D_n$ entonces $\exists j : D_j(\overrightarrow{a}) = 0$, pero $D_j = l_{1,j} \wedge l_{2,j} \wedge \ldots \wedge l_{r_j,j}$ 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.

- 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:

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

  Absurdo.

- $r_j = 1$

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

  Absurdo.

- $r_j > 4$

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

  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 $y_{i,j}$ y se cumple (ejercicio)

El único problema grande es $r_j > 4$

Como $B(\overrightarrow{a}) = 1$ entonces $D_j(\overrightarrow{a}) = 1$ (al menos un término 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:

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

Entonces, si valueamos tenemos:

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

□

VI.2. 3-Color es NP-COMPLETO

Teorema 20 *(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 $3-\text{SAT}$, esto queda fijo).

Construcción polinomialmente del grafo G:

Vertices:

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

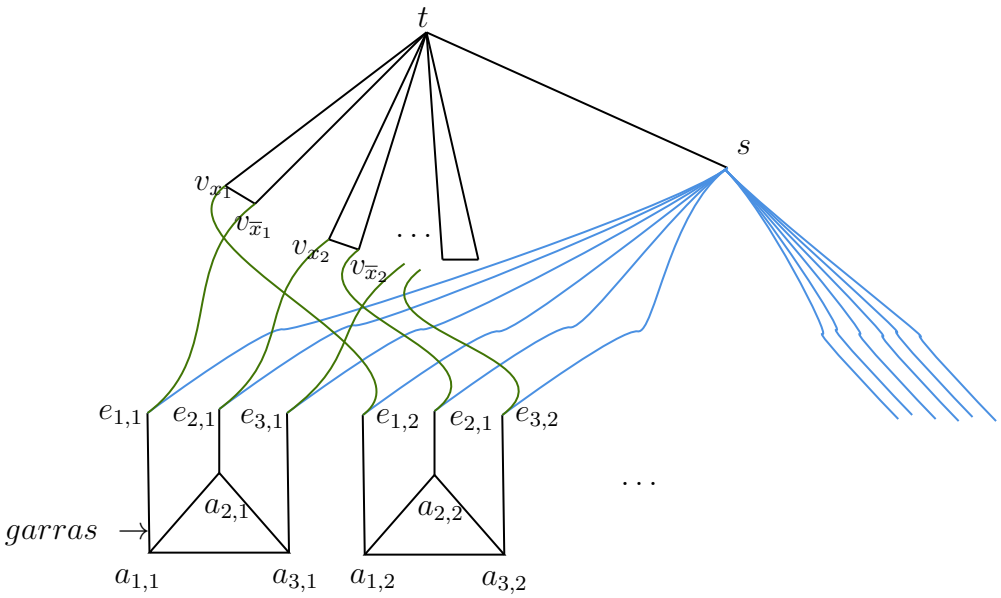
donde $\{v_l : l \text{ es literal}\}$ es equivalente a $\{v_{x_1}, v_{x_2}, \ldots, v_{x_n}, v_{\bar{x}_1}, v_{\bar{x}_2}, \ldots, v_{\bar{x}_n}\}$

Lados:

- st
- tv_l para todo literal l
- $v_{x_i} v_{\bar{x}_i}$ $\forall i = 1 \ldots, n$
- $a_{1,j} a_{2,j}$
- $a_{2,j} a_{3,j}$
- $a_{1,j} e_{3,j}$ 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 $j = 1, 2, \ldots, m$
- $sl_{i,j}$ para $i = 1, 2, 3$ y $j = 1, 2, \ldots, m$

- Usando que es 3−SAT, es decir, $D_j = l_{1,j} \vee l_{2,j} \vee l_{3,j} = 0$ 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 $\overrightarrow{v_x e_{1,j}}, \overrightarrow{v_{\bar{x}_2} e_{2,j}}, \overrightarrow{v_{x_4} e_{e,j}}$

El grafo es como en la siguiente imagen:



Observemos que G tiene un triangulo, 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 locoreo 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)

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

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:

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

$$i = j \rightarrow \underbrace{a_{i_j,j}}_{2} \underbrace{e_{i_j,j}}_{0} \Rightarrow NCP$$

$$\underbrace{s}_{1} \underbrace{e_{i,j}}_{0 \text{ ó } 2} \Rightarrow NCP$$

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

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

$i = i_j \rightarrow$

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

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

es igual a 1 por la elección del i_j

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

□