

Resumen de teoremas para el final de Matemática Discreta II

Agustin Curto, agucurto95@gmail.com

2016

Índice general

1. Parte A	2
1.1. La complejidad de EDMONS-KARP	2
1.2. Las distancias de Edmonds-Karp no disminuyen en pasos sucesivos	3
1.3. La complejidad de DINIC	5
1.4. La complejidad de WAVE	6
1.5. La distancia entre NA sucesivos aumenta	8
2. Parte B	9
2.1. 2-COLOR es polinomial	9
2.2. Teorema Max-Flow Min-Cut	10
2.3. Complejidad del Hungaro es $\mathcal{O}(n^4)$	12
2.4. Teorema de Hall	13
2.5. Teorema del matrimonio	14
2.6. Todo grafo bipartito es Δ coloreable	15
2.7. Cota de Hamming	17
2.8. Teorema de la matriz de chequeo de códigos lineales	18
2.9. Teorema del polinomio generador de códigos cíclicos	19
3. Parte C	21
3.1. 4-COLOR \leq_p SAT	21
3.2. 3-SAT es NP-Completo	23
3.3. 3-COLOR es NP-Completo	25

Capítulo 1

Parte A

1.1. La complejidad de EDMONS-KARP

Teorema: La complejidad de $\langle E - K \rangle$ con $n = |V|$ y $m = |E|$ es $\mathcal{O}(nm^2)$.

Prueba: Sean: f_0, f_1, f_2, \dots la sucesión de flujos creados por $\langle E - K \rangle$. Es decir, el paso k crea f_k .

Para cada k definimos funciones:

- $d_k(x)$ = “distancia” entre s y x en el paso k , en caso de existir, si no ∞ .
- $b_k(x)$ = “distancia” entre x y t en el paso k , en caso de existir, si no ∞ .

“Distancia”: longitud del menor camino aumentante entre dos vértices.

Sabemos que las distancias de $\langle E - K \rangle$ no disminuyen en pasos sucesivos, como esto será útil para esta demostración llamaremos \otimes a la demostración de:

$$d_k(x) \leq d_{k+1}(x)$$

$$b_k(x) \leq b_{k+1}(x)$$

Llamemos *crítico* a un lado disponible en el paso k pero no disponible en el paso $k + 1$. Es decir, si xy es un lado $\Rightarrow xy$ se satura ó yx se vacía en el paso k .

Supongamos que al construir f_k el lado xy se vuelve crítico, el camino: $s \dots x$, y $\dots t$ se usa para construir f_k .

$$\begin{aligned} d_k(t) &= d_k(x) + b_k(x) \\ &= d_k(x) + b_k(y) + 1 \end{aligned}$$

Para que xy pueda ser *crítico* nuevamente debe ser usado en la otra dirección. Sea l el paso posterior a k en el cual se usa el lado en la otra dirección, el camino $s \dots y$, $x \dots t$ se usa para construir f_l .

$$\begin{aligned} d_l(t) &= d_l(x) + b_l(x) \\ &= d_l(y) + 1 + b_l(x) \end{aligned}$$

Entonces:

$$\text{Como estoy en } \langle E - K \rangle \Rightarrow \begin{cases} d_k(y) = d_k(x) + 1 & \star \\ d_l(x) = d_l(y) + 1 & \dagger \end{cases}$$

Luego:

$$\begin{aligned}
d_l(t) &= d_l(x) + b_l(x) \\
&= d_l(y) + 1 + b_l(x) && \text{Por } \dagger \\
&\geq d_k(y) + 1 + b_k(x) && \text{Por } \circledast \\
&= d_k(x) + 1 + 1 + b_k(x) && \text{Por } \star \\
&= d_k(t) + 2 \\
\therefore d_l(t) &\geq d_k(t) + 2
\end{aligned}$$

Por lo tanto cuando un lado se vuelve crítico recién puede volver a usarse cuando la distancia de s a t haya aumentado en por lo menos 2. Puede existir $\mathcal{O}(n/t)$ tales aumentos, es decir:

$$\# \text{ Veces que un lado puede volverse crítico} = \mathcal{O}(n).$$

$$\begin{aligned}
\therefore \text{Complejidad}(\langle E - K \rangle) &= (\# \text{pasos}) * \text{Compl}(1 \text{ paso}) \\
&= (\# \text{veces que un lado se vuelve crítico}) * (\# \text{lados}) * \text{Compl}(BFS) \\
&= \mathcal{O}(n) * \mathcal{O}(m) * \mathcal{O}(m) \\
&= \mathcal{O}(nm^2)
\end{aligned}$$

1.2. Las distancias de Edmonds-Karp no disminuyen en pasos sucesivos

Teorema: Sean: f_0, f_1, f_2, \dots la sucesión de flujos creados por $\langle E - K \rangle$. Es decir, el paso k crea f_k .

Para cada k definimos funciones:

- $d_k(x)$ = “distancia” entre s y x en el paso k en caso de existir, si no ∞ .
- $b_k(x)$ = “distancia” entre x y t en el paso k en caso de existir, si no ∞ .

“Distancia”: longitud del menor camino aumentante entre dos vértices.

Queremos probar que:

1. $d_k(x) \leq d_{k+1}(x)$
2. $b_k(x) \leq b_{k+1}(x)$

Prueba: Lo probaremos por inducción y solo para d_k ya que para b_k la prueba es análoga.

$$\text{HI: } H(i) = \{\forall_z : d_{k+1}(z) \leq i, \text{ vale } d_k(z) \leq d_{k+1}(z)\}$$

1. Caso Base: $\boxed{i = 0}$ $H(0) = \{\forall_z : d_{k+1}(z) \leq 0, \text{ vale } d_k(z) \leq d_{k+1}(z)\}$

Pero $d_{k+1}(z) \leq 0 \Rightarrow z = s$, entonces:

$$\begin{aligned}
d_k(z) &= d_k(s) \\
&= 0 \\
&\leq d_{k+1}(s) \\
&\leq d_{k+1}(z) \\
\therefore d_k(z) &\leq d_{k+1}(z)
\end{aligned}$$

2. Caso Inductivo: Supongamos ahora que vale $H(i)$, veamos que vale $H(i + 1)$.

Sea z con $d_{k+1}(z) \leq i+1$, si $d_{k+1}(z) \leq i$ vale $H(i)$ para z .

$$\therefore d_k(z) \leq d_{k+1}(z)$$

Supongamos que $\boxed{d_{k+1}(z) = i + 1}$

Entonces existe un camino aumentante, relativo a f_k , de la forma: $s = z_0, z_1, \dots, z_i, z_{i+1} = z$.

Sea $\boxed{x = z_i}$

- Caso 1: Existe algun camino aumentante, relativo a f_{k-1} de la forma s, \dots, x, z .

$$\therefore d_k(z) \leq d_k(x) + 1$$

Pues al haber un camino $\underbrace{s, \dots, x}_{d_k(x)}, z$, llamemosle A, de longitud $d_k(x) + 1$ entre s y z , sabemos que el mínimo de todos los caminos de s a z será $\leq A$.

- Caso 2: No existe un camino aumentante, relativo a f_{k-1} , pero si existe un camino aumentante relativo a f_k . Por lo tanto el lado xz no esta “disponible” en el paso k , ya que xz está saturado, o bien zx está vacío relativo a f_{k-1} . Para construir f_k usamos un camino de la forma s, \dots, z, x . Es decir:

- 1) $f_{k-1}(\vec{xz}) = \text{Cap}(\vec{xz})$ pero $f_k(\vec{xz}) < \text{Cap}(\vec{xz})$, f_k devuelve flujo por \vec{xz} ó
- 2) $f_{k-1}(\vec{zx}) = 0$ pero $f_k(\vec{zx}) > 0$, f_k manda flujo por \vec{zx} .

Como $\langle E - K \rangle$ funciona con BFS, ese camino usado para construir f_k debe ser de longitud mínima. Es decir:

$$\begin{aligned} d_k(x) &= d_k(z) + 1 \\ \Rightarrow d_k(z) &= d_k(x) - 1 \\ &\leq d_k(x) + 1 \end{aligned}$$

Conclusión: En cualquiera de los dos casos tenemos:

$$\boxed{d_k(z) \leq d_k(x) + 1}$$

Ahora bien:

$$\begin{aligned} d_{k+1}(x) &= d_{k+1}(z_i) \\ &= i \\ &\Rightarrow H(i) \text{ vale para } x. \\ \therefore d_k(z) &\leq d_{k+1}(x) \end{aligned}$$

Por lo tanto:

$$\begin{aligned} d_k(z) &\leq d_k(x) + 1 \\ &\leq d_{k+1}(x) + 1 \\ &= i + 1 \\ &= d_{k+1}(z) \\ &\Rightarrow H(i + 1) \text{ vale.} \end{aligned}$$

1.3. La complejidad de DINIC

Teorema: La complejidad del algoritmo de Dinic es $\mathcal{O}(n^2m)$.

Prueba: Como Dinic es un algoritmo que trabaja con networks auxiliares y vimos que la distancia entre s y t en networks auxiliares consecutivos aumenta y puede ir a lo sumo entre 1 y $n - 1$ entonces hay a lo sumo $\mathcal{O}(n)$ networks auxiliares.

$$\text{Complejidad(Dinic)} = \mathcal{O}(n) * \text{Complejidad(Paso Bloqueante)}$$

Para probar que la complejidad de Dinic es $\mathcal{O}(n^2m)$ debemos probar que la complejidad del paso bloqueante es $\mathcal{O}(nm)$.

Sea:

- A = Avanzar()
- R = Retroceder()
- I = IncrementarFlujo() + Inicialización()

Una corrida de Dinic luce como:

$$AA \dots AIAAARA \dots AIAARAAARR \dots IA \dots$$

Dividamos la corrida en subpalabras del tipo:

$$\left. \begin{array}{l} \underbrace{AA \dots AI}_{\text{Todas } A's} \\ \underbrace{AA \dots AR}_{\text{Todas } A's} \end{array} \right\} \text{ Sea } X = I \text{ o } R$$

Nota: el número de A's puede ser 0.

Debemos determinar:

1. Cantidad de subpalabras

$$\left. \begin{array}{l} X = R : \text{borramos un lado} \\ X = I : \text{borramos al menos un lado} \end{array} \right\} \Rightarrow \text{Cada } X \text{ borra al menos un lado}$$

$$\therefore \text{hay } \leq m \text{ palabras de la forma } A \dots AX$$

2. Complejidad de cada subpalabra

Recordemos que:

$$\begin{aligned} A: [P[i+1] = \text{algún elemento de } \Gamma^+(P[i]) \\ \Rightarrow A \text{ es } \mathcal{O}(1) \end{aligned}$$

$$\begin{aligned} R: [\text{Borrar } P[i-1]P[i] \text{ del NA} \\ \Rightarrow R \text{ es } \mathcal{O}(1) \end{aligned}$$

$$\begin{aligned} I: [\text{Recorre un camino de longitud } \leq n \\ \Rightarrow I \text{ es } \mathcal{O}(n) \end{aligned}$$

Luego:

$$\begin{aligned} \text{Complejidad}(X) &= \mathcal{O}(n) \\ \Rightarrow \text{Complejidad}(\underbrace{A \dots A}_j X) &= \underbrace{\mathcal{O}(j) + \mathcal{O}(n)} \end{aligned}$$

Como cada Avanzar() mueve el pivote un nivel más cerca de t entonces hay a lo sumo n Avanzar() antes de un R o un I $\Rightarrow r \leq n$.

$$\therefore \text{Complejidad}(A \dots AX) = \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$$

De (1) y (2):

$$\begin{aligned} \text{Complejidad}(\text{Paso Bloqueante}) &= \# \text{ Cantidad de palabras} * \text{Complejidad de la palabra} \\ &= m * \mathcal{O}(n) \\ &= \mathcal{O}(m * n) \end{aligned}$$

1.4. La complejidad de WAVE

Teorema: La complejidad del algoritmo de Wave es $\mathcal{O}(n^3)$.

Prueba: Como Wave es un algoritmo que trabaja con networks auxiliares y vimos que la distancia entre s y t en networks auxiliares consecutivos aumenta y puede ir a lo sumo entre 1 y $n - 1$ entonces hay a lo sumo $\mathcal{O}(n)$ networks auxiliares.

$$\text{Complejidad}(\text{Wave}) = \mathcal{O}(n) * \text{Compl}(\text{Paso bloqueante de Wave})$$

Para probar que la complejidad de Wave es $\mathcal{O}(n^3)$ debemos probar que complejidad del paso bloqueante es $\mathcal{O}(n^2)$. El paso bloqueante de Wave consiste en una serie de:

- Olas hacia adelante: Sucesión de **fordwardbalance** (FB)
- Olas hacia atrás: Sucesión de **backwardbalance** (BB)

Cada FB y BB es una sucesión de “**buscar vecinos**” y “**procesar**” el lado resultante. Estos “procesamientos” son complicados pero $\mathcal{O}(1)$.

$$\therefore \text{Compl}(\text{Paso Bloqueante}) = \# \text{ “procesamientos de lado”}$$

Los “procesamientos” de lados los podemos dividir en dos categorías:

1. Aquellos procesamientos que saturan o vacían el lado. Denotaremos “T” al número de estos procesamientos.
2. Aquellos procesamientos que no saturan ni vacían el lado. Denotaremos “Q” al número de estos procesamientos.

Por lo tanto queremos acotar $T + Q$.

Complejidad de T:

- ¿Puede un lado \overrightarrow{xy} saturado volver a saturarse?

Para poder volver a saturarse primero tiene que vaciarse aunque sea un poco, es decir, antes de poder volver a saturarlo “y” debe devolver flujo a “x”, pero para que en Wave “y” le devuelva flujo a “x” debe ocurrir que “y” este bloqueado (porque BB(y) solo se ejecuta si “y” está bloqueado), pero si “y” está bloqueado “x” no puede mandarle flujo nunca más.

$\therefore \overrightarrow{xy}$ no puede resaturarse

Conclusión 1: Los lados se saturan solo una vez.

- ¿Puede un lado \overrightarrow{xy} vaciado completamente volver a vaciarse?

Para poder volver a vaciarse como está vacío completamente, primero hay que mandar flujo, pero si lo vacié “y” está bloqueado por lo que “x” no puede mandar flujo.

$\therefore \overrightarrow{xy}$ no puede volver a vaciarse

Conclusión 2: Los lados se vacían completamente a lo sumo una vez.

Las conclusiones (1) y (2) implican que $\boxed{T \leq 2m}$

Complejidad de Q:

En cada FB a lo sumo un lado no se satura y en cada BB a lo sumo un lado no se vacía completamente.

$\therefore Q \leq \# \text{ Total de FB's y BB's}$

- $\# \text{ FB's en cada ola hacia adelante es } \leq n$ (un FB por vértice)
- $\# \text{ BB's en cada ola hacia atrás es } \leq n$

$\therefore \text{ Total de FB's y BB's } \leq 2n \# \text{ Total de ciclos de “ola adelante – ola hacia atrás”}$

Ahora bien, en cada ola hacia adelante, pueden o no, bloquearse algunos vértices. Si no se bloquea ningún vértice entonces todos los vértices ($\neq s, t$) quedan balanceados por lo que estamos en la última ola. Luego en toda ola que no sea la última se bloquea al menos un vértice ($\neq s, t$).

$\therefore \# \text{ Total de ciclos es } \leq (n - 2) + 1 = n - 1$

$$\Rightarrow \boxed{Q \leq 2n(n - 1) = \mathcal{O}(n^2)}$$

$$\begin{aligned} \therefore T + Q &\leq 2m + \mathcal{O}(n^2) \\ &= \mathcal{O}(m) + \mathcal{O}(n^2) \\ &= \mathcal{O}(n^2) \end{aligned}$$

1.5. La distancia entre NA sucesivos aumenta

Teorema: Sea A un NA (network auxiliar) y sea A^* el siguiente NA. Sean $d(x)$ y $d^*(x)$ las distancias de s a t en A y A^* respectivamente, entonces: $d(t) < d^*(t)$.

Prueba: Como A y A^* se construyen con BFS sabemos que $d(t) \leq d^*(t)$ pero queremos ver el $<$.

Sea:

$$s = x_0, x_1, \dots, t = x_r$$

un camino dirigido en A^* .

Ese camino No existe en A ya que para pasar de A a A^* debemos bloquear todos los caminos dirigidos de A . Por lo tanto si ese camino estuviese en A , Dinic lo habría bloqueado y no estaría en A^* .

¿Cuáles son las razones posibles para que no esté en A ?

1. Puede faltar un vértice, es decir $\exists i : x_i \notin V(A)$ entonces:

$$\begin{aligned} d(t) &\leq d(x_i) \\ &\leq d^*(x_i) \quad \text{Por } \langle E - K \rangle \\ &= d^*(t) \quad \text{Porque } x_i \text{ esta antes que } t \end{aligned}$$

$$\boxed{\therefore d(t) < d^*(t)}$$

2. Están todos los vértices pero falta una arista, es decir $\exists i : \overrightarrow{x_i x_{i+1}} \notin E(A)$.

a) $\overrightarrow{x_i x_{i+1}}$ no está porque corresponde a un lado vacío o saturado en NA, es decir $\overrightarrow{x_i x_{i+1}}$ no está en el residual que dá origen a A pero si está en el residual que dá origen a A^* . Para que esto pase se tiene que haber usado el lado $x_{i+1}x_i$ en A . Luego podemos concluir, por la prueba de $\langle E - K \rangle$ que:

$$d^*(t) \geq d(t) + 2 > d(t)$$

$$\boxed{\therefore d(t) < d^*(t)}$$

b) $\overrightarrow{x_i x_{i+1}}$ si está en el residual pero: $d(x_{i+1}) \neq d(x_i) + 1$ (1)

Pero como $x_i x_{i+1}$ está en el residual entonces: $d(x_{i+1}) \leq d(x_i) + 1$ (2)

De (1) y (2) tenemos que: $d(x_{i+1}) < d(x_i) + 1$ \circledast

Entonces:

$$\begin{aligned} d(t) &= d(x_{i+1}) + b(x_{i+1}) && \text{Por } \langle E - K \rangle \\ &\leq d(x_{i+1}) + b^*(x_{i+1}) && \text{Por } \langle E - K \rangle \\ &< d(x_i) + 1 + b^*(x_{i+1}) && \text{Por } \circledast \\ &\leq d^*(x_i) + 1 + b^*(x_{i+1}) && \text{Por } \langle E - K \rangle \\ &= d^*(x_{i+1}) + b^*(x_{i+1}) && \text{Por } (\dagger) \\ &= d^*(t) \end{aligned}$$

$$\boxed{\therefore d(t) < d^*(t)}$$

(\dagger): Ya que s, x_1, \dots, x_r es un camino en A^* .

Capítulo 2

Parte B

2.1. 2-COLOR es polinomial

Teorema: 2-Color es polinomial, es decir, existe un algoritmo polinomial que lo resuelve.

Prueba: Consideremos el siguiente algoritmo con entrada $G = (V, E)$ con $n = |V|$ y $m = |E|$.

Algoritmo:

Por cada COMPONENTE CONEXA:

1. Elegir x
2. Correr $BFS(x)$
3. Colorear vertices con: $C(v) = Nivel_{BFS(x)}(v) \bmod 2$
4. Chequear que el coloreo de (3) sea propio. Si lo es, RETURN “SI”, si no lo es, RETURN “NO”.

Veamos ahora que el algoritmo es correcto y que su complejidad es polinomial.

■ Complejidad Polinomial:

- (1) es $\mathcal{O}(1)$.
- (2) + (3) es $\mathcal{O}(m)$ (recorrer todos los vértices).
- (4) es $\mathcal{O}(m)$ ya que chequear que un coloreo es propio, es recorrer todos los lados comprobando que los vértices tengan distintos colores.

■ Correctitud:

- Cuando dice “SI”: Obviamente dice “SI”, solo si el coloreo es propio y si el coloreo es propio, $\chi(G) \leq 2$.
- Cuando dice “NO”: Tenemos que ver que cuando dice “NO” ningún coloreo propio con 2 colores existe.

Por otro teorema visto en clase, si existe un ciclo impar $C_{2k+1} \subseteq G \Rightarrow \chi(G) \geq 3$. Por lo que bastaría ver que si el algoritmo dice “NO”, entonces exista un ciclo impar en G .

Si el algoritmo dice “NO” $\Rightarrow \exists z, y : zy \in E(G)$ y $C(z) = C(y) \Rightarrow Nivel_{BFS(x)}(z) = Nivel_{BFS(x)}(y)$

Sea:

- $x \dots z$ un camino en $\text{BFS}(x)$, entre x y z .
 - $t = \text{Nivel}_{\text{BFS}(x)}(z)$.
 - $x \dots y$ un camino en $\text{BFS}(x)$, entre x e y .
 - $r = \text{Nivel}_{\text{BFS}(x)}(y)$.
 - w el vértice de mayor nivel tal que: $x \dots w$ sea prefijo común en $x \dots z$ y $x \dots y$.
- Es decir:

$$\begin{array}{ccc} x \dots w & & \dots z \\ \underbrace{x \dots w}_{\text{Camino Igual}} & & \underbrace{\dots y}_{\text{Camino distinto}} \end{array}$$

Considramos el ciclo: $w \dots zy \dots w$

$$\begin{aligned} \text{Longitud del ciclo} &= (t - s) + 1 + (r - s) \\ &= (t + r) - 2s + 1 \\ &\equiv 0 + 0 + 1 = 1 \pmod{2} \end{aligned}$$

\therefore Es un ciclo impar

2.2. Teorema Max-Flow Min-Cut

Teorema:

- a) Si f es flujo y S es corte $\Rightarrow V(f) \leq \text{Cap}(S)$.
- b) Si $V(f) = \text{Cap}(S) \Rightarrow f$ es maximal y S es minimal.
- c) Si f es maximal $\Rightarrow \exists S$ con $V(f) = \text{Cap}(S)$.

Prueba: Denotemos (\star) , a la demostración de:

$$\boxed{V(f) = f(S, \bar{S}) - f(\bar{S}, S)}$$

a) f es flujo y S es corte $\Rightarrow V(f) \leq \text{Cap}(S)$.

$$\begin{aligned} V(f) &\stackrel{\text{Por } (\star)}{=} f(S, \bar{S}) - \underbrace{f(\bar{S}, S)}_{\substack{\geq 0 \\ \leq 0}} \\ &\leq f(S, \bar{S}) \\ &\leq \text{Cap}(S, \bar{S}) \\ &= \text{Cap}(S) \end{aligned}$$

b) $V(f) = \text{Cap}(S) \Rightarrow f$ es maximal y S es minimal.

Supongamos que $V(f) = \text{Cap}(S)$. Sea g un flujo cualquiera y T un corte cualquiera.

$$\blacksquare V(g) \stackrel{\text{Por a)}}{\leq} \text{Cap}(S) = V(f) \Rightarrow f \text{ es maximal.}$$

$$\blacksquare \text{Cap}(T) \stackrel{\text{Por a)}}{\geq} V(f) = \text{Cap}(S) \Rightarrow S \text{ es minimal.}$$

c) f es maximal $\Rightarrow \exists S$ con $V(f) = \text{Cap}(S)$.

Sea $S = \{s\} \cup \{x : \exists \text{ camino aumentante realtivo a } f \text{ entre } s \text{ y } x\}$

$\boxed{t \in S?}$ $t \in S?$

Si t estuviera en S : existiría un camino aumentante entre s y t .

Por el teorema del camino aumentante podemos construir un flujo g tal que:

$$\begin{aligned} V(g) &= V(f) + \epsilon \text{ para algun } \epsilon > 0 \\ \therefore V(g) &> V(f) \text{ Absurdo! pues } f \text{ es maximal} \\ \therefore t &\notin S \Rightarrow S \text{ es corte.} \end{aligned}$$

Solo resta ver que: $V(f) = \text{Cap}(S)$

$$\text{Por } (\star) : V(f) = \underbrace{f(S, \bar{S})}_{(1)} - \underbrace{f(\bar{S}, S)}_{(2)}$$

Analicemos (1) y (2)

$$(1) f(S, \bar{S}) = \sum_{\substack{x \in S \\ y \in \bar{S} \\ xy \in E}} f(\vec{xy})$$

$x \in S \Rightarrow \exists$ camino aumentante $s, \dots x$.

$y \in \bar{S} \Rightarrow \nexists$ camino aumentante entre s y y . En particular $s, \dots x, \dots y$ no es camino aumentante, por lo que no puede darse que: $f(\vec{xy}) < \text{Cap}(\vec{xy})$.

$$\Rightarrow f(\vec{xy}) = \text{Cap}(\vec{xy}) \quad \forall x \in S, \forall y \in \bar{S} : \vec{xy} \in E.$$

$$\Rightarrow \boxed{f(S, \bar{S})} = \sum_{\substack{x \in S \\ y \in \bar{S} \\ xy \in E}} f(\vec{xy}) = \sum_{\substack{x \in S \\ y \in \bar{S} \\ xy \in E}} \text{Cap}(\vec{xy}) = \text{Cap}(S, \bar{S}) = \boxed{\text{Cap}(S)}$$

$$(2) f(\bar{S}, S) = \sum_{\substack{x \in \bar{S} \\ y \in S \\ xy \in E}} f(\vec{xy})$$

$x \in \bar{S} \Rightarrow \nexists$ camino aumentante entre s y x .

$y \in S \Rightarrow \exists$ camino aumentante $s, \dots y$.

En particular $s \dots y, y \dots x$ no es camino aumentante $\Rightarrow f(\vec{xy}) = 0 \quad \forall x \in \bar{S}, \forall y \in S : \vec{xy} \in E$.

$$\therefore \boxed{f(\bar{S}, S) = 0}$$

Luego de (1) y (2):

$$\begin{aligned} V(f) &= f(S, \bar{S}) - f(\bar{S}, S) \\ &= \text{Cap}(S) - 0 \\ &= \text{Cap}(S) \end{aligned}$$

2.3. Complejidad del Hungaro es $\mathcal{O}(n^4)$

Teorema: La complejidad del algoritmo Húngaro es $\mathcal{O}(n^4)$.

Prueba:

1. La complejidad del matching inicial es $\mathcal{O}(n^2)$, ya que:

Restar mínimo de cada fila:

$$\left(\underbrace{\mathcal{O}(n)}_{\text{calcular min}} + \underbrace{\mathcal{O}(n)}_{\text{restar min}} \right) * \underbrace{n}_{\# \text{filas}} = \mathcal{O}(n^2)$$

Idem para las columnas.

2. Llamemos **extender** el matching, a incrementar su número de filas en 1, es decir, agregar una fila más al matching.

$$\# \text{ extensiones de matching} = \mathcal{O}(n)$$

Resta ver la complejidad de cada **extender**.

3. En cada extensión vamos a ir revisando filas y columnas, donde escanear una fila es $\mathcal{O}(n)$ y se realizan n escaneos, por lo que sería $\mathcal{O}(n^2)$ sin considerar que se debe realizar un cambio de matriz.

Hacer un **cambio de matriz** es $\mathcal{O}(n^2)$, ya que:

- Buscar $m = \min S \times \overline{\Gamma(S)} \rightarrow \mathcal{O}(n^2)$
- Restar m de $S \rightarrow \mathcal{O}(n^2)$
- Sumar m a $\Gamma(S) \rightarrow \mathcal{O}(n^2)$

Luego la implementación NAIVE lanzaría nuevamente el algoritmo desde cero. La forma correcta es continuar con el matching que teníamos, ya que el mismo no se pierde.

Si lo hacemos así, ¿Cuántos **cambios de matriz** habrá antes de extender un matching nuevamente?

Lema Interno: Luego de un **cambio de matriz**, se extiende el matching o bien se aumenta S .

Prueba:

$$\left(\begin{array}{c|c} A & A \\ \hline B & C \end{array} \right) \begin{array}{c} \overline{S} \\ S \end{array}$$

$$\Gamma(S) \quad \overline{\Gamma(S)}$$

Referencias:

- A: puede haber ceros.
- B: ceros del matching.
- C: no hay ceros, no hay matching.

Al restar $m = \min S \times \overline{\Gamma(S)}$ de las filas de S , habrá un nuevo cero en alguna fila $i \in S$ y columna $j \in \overline{\Gamma(S)}$ entonces la columna se etiquetará con i y se revisará. Tenemos dos resultados posibles:

- a) j está libre (i.e no forma parte del matching) \Rightarrow extendemos el matching.
b) j forma parte de matching $\Rightarrow \exists$ fila k matcheada con j . En este caso, la fila k se etiquetará con j , por lo que el "nuevo" $S \geq S \cup \{k\}$.

Entonces se termina con una extensión o se produce un nuevo S de cardinalidad, al menos $|S| + 1$.

Fin Lema Interno

Luego como $|S|$ solo puede crecer $\mathcal{O}(n)$ veces, tenemos que hay a lo sumo n **cambios de matriz** antes de extender el matching. Entonces:

$$\begin{aligned} \text{Complejidad}(1 \text{ Extensión}) &= \underbrace{\mathcal{O}(n)}_{\#CM} * \underbrace{\mathcal{O}(n^2)}_{\text{Compl}(CM)} + \underbrace{\mathcal{O}(n^2)}_{\text{Busqueda } n \text{ filas } \times n \text{ columnas}} \\ \therefore \text{Complejidad}(\text{Húngaro}) &= \underbrace{\mathcal{O}(n^2)}_{\text{Matching inicial}} + (\underbrace{\mathcal{O}(n)}_{\#extensiones} * \underbrace{\mathcal{O}(n^3)}_{\text{Compl}(extension)}) = \mathcal{O}(n^4) \end{aligned}$$

2.4. Teorema de Hall

Teorema: Sea $G = (X \cup Y, E)$ grafo bipartito $\Rightarrow \exists$ matching completo de X a $Y \Leftrightarrow |S| \leq |\Gamma(S)| \forall S \subseteq X$.

Prueba:

\Rightarrow) Si M es matching completo de X a Y entonces observemos que M induce una función inyectiva de X a Y .

$$f(x) = \text{unico } y : xy \in E(M)$$

1. Si $S \subseteq X \Rightarrow |S| = |f(S)|$
2. Además por definición de f , $f(x) \in \Gamma(x)$

$$\begin{aligned} \text{Si } x \in S &\Rightarrow f(x) \in \Gamma(S) \\ &\Rightarrow f(S) \subseteq \Gamma(S) \\ \therefore |f(S)| &\leq |\Gamma(S)| \end{aligned}$$

De (1) y (2) $\Rightarrow |S| \leq |\Gamma(S)|$.

\Leftarrow) Supongamos que no es cierto, entonces G es bipartito con $|S| \leq |\Gamma(S)| \forall S \subseteq X$ pero no tiene matching completo de X a Y . Esto equivalente a ver que: si \nexists un matching completo de X a $Y \Rightarrow \exists S \subseteq X : |S| > |\Gamma(S)|$.

Corramos el algoritmo para hallar matching. Al finalizar habrá filas sin matcher (las de s).

Sean:

- $S_0 = \{\text{filas sin matchear}\}$.
- $T_1 = \Gamma(S_0)$, es decir, las columnas etiquetadas por las filas de S_0 . Todas las columnas de T_1 están matcheadas, pues si no, se podría agregar alguna alguna fila de S_0 al matching.
- $S_1 = \{\text{filas etiquetadas por las columnas de } T_1\}$.

- $T_2 = \Gamma(S_1) - T_1$, es decir, columnas etiquetadas por las filas de S_1 .

En general:

- $S_i = \{\text{filas matcheadas con } T_i\}$.
- $T_{i+1} = \Gamma(S_i) - (T_1 \cup T_2 \cup \dots T_i)$.

Como el algoritmo para sin hayar matching, entonces $\forall i \ T_i \neq \emptyset$, produce un S_i (i.e $S_i \neq \emptyset$).
 \therefore La única forma de parar es en un k , tal que $T_{k+1} = \emptyset$.

Observaciones:

1. $|S_j| = |T_j|$, pues S_j son las filas matcheadas con T_j .
2. Por construcción, los S_i y T_i son todos distintos.
3. $\Gamma(S_0 \cup S_1 \cup \dots S_j) = T_1 \cup T_2 \cup \dots T_{j+1}$

Por inducción en j :

- Caso Base: $j = 0$ vale, ya que $T_1 = \Gamma(S_0)$
- Caso Inductivo: Supongamos que vale para j , veamos para $j+1$.

$$\begin{aligned}
 T_1 \cup T_2 \dots T_{j+1} &= T_1 \cup T_2 \cup \dots T_{j+1} \cup \underbrace{(\Gamma(S_{j+1}) - (T_1 \cup T_2 \dots T_{j+1}))}_{T_{j+2}} \\
 &= T_1 \cup T_2 \cup \dots T_{j+1} \cup \Gamma(S_{j+1}) \\
 &= \Gamma(S_0 \cup S_1 \cup \dots S_j) \cup \Gamma(S_{j+1}) \quad \text{Por H.I} \\
 &= \Gamma(S_0 \cup S_1 \cup \dots S_j \cup S_{j+1})
 \end{aligned}$$

Sea $S = S_0 \cup S_1 \cup \dots S_k$

$$\begin{aligned}
 |\Gamma(S)| &= |\Gamma(S_0 \cup S_1 \cup \dots S_k)| \\
 &= |T_1 \cup T_2 \cup \dots \underbrace{T_{k+1}}_{=\emptyset}| \quad \text{Por Obs 3} \\
 &= |T_1 \cup T_2 \cup \dots T_k| \\
 &= |T_1| + |T_2| + \dots |T_k| \quad \text{Por Obs 2} \\
 &= |S_1| + |S_2| + \dots |S_k| \quad \text{Por Obs 1} \\
 &= |S| - |S_0| \quad \text{Por Obs 2} \\
 &< |S| \quad \text{Pues } S_0 \neq \emptyset
 \end{aligned}$$

Absurdo!

El absurdo vino de suponer que G es bipartito con $|S| \leq |\Gamma(S)| \ \forall S \subseteq X$ pero que no tiene matching completo de X a Y .

2.5. Teorema del matrimonio

Teorema: Todo grafo regular bipartito tiene un matching perfecto.

Prueba: Sea $G = (X \cup Y, E)$ bipartito regular, tal que $\forall W \subseteq V(G)$, definimos:

$$\begin{aligned}
 E_W &= \{xy \in E(G) : x \in W \text{ o } y \in W\} \\
 &= \{\text{lados con un extremo en } W\}
 \end{aligned}$$

Supongamos que $W \subseteq X$ (de igual forma para $W \subseteq Y$). Además, como G es regular, $\exists \Delta = \delta > 0 : d(z) = \Delta \ \forall z$.

$$\begin{aligned} |E_w| &= |\{xy \in E : x \in W\}| \\ &= \sum_{x \in w} |\{y : xy \in E\}| \\ &= \sum_{x \in w} \underbrace{d(x)}_{\Delta} \\ &= \Delta * |w| \end{aligned}$$

Es decir, a cada w le corresponden Δ lados distintos. En particular:

$$\begin{aligned} |E_x| &= \Delta * |x| \\ |E_y| &= \Delta * |y| \end{aligned}$$

pero $E_x = E = E_y$ pues G es bipartito.

Por lo tanto:

$$\begin{aligned} |E_x| = |E_y| &\Rightarrow \Delta * |x| = \Delta * |y| \\ &\Rightarrow |x| = |y| \\ &\Rightarrow \text{Todo matching completo es perfecto} \end{aligned}$$

Basta ver que existe un matching completo de X a Y , es decir, que se cumple la condición de Hall.

Sea $S \subseteq X$:

$$\text{Sea } \underbrace{xy}_{\substack{x \in X \\ y \in Y}} \in E_s \Rightarrow \exists \left\{ \begin{array}{l} x \in S \\ y \in \Gamma(x) \end{array} \right\} \Rightarrow y \in \Gamma(S) \Rightarrow xy \in E_{\Gamma(S)}$$

Es decir, hemos probado que:

$$\begin{aligned} E_s &\subseteq E_{\Gamma(S)} \\ |E_s| &\leq |E_{\Gamma(S)}| \\ \Delta |S| &\leq \Delta |\Gamma(S)| \\ |S| &\leq |\Gamma(S)| \end{aligned}$$

2.6. Todo grafo bipartito es Δ coloreable

Teorema: Si G es bipartito $\Rightarrow \chi'(G) = \Delta$

Prueba:

Lema Interno: Todo grafo bipartito es subgrafo de un grafo bipartito regular con el mismo Δ , es decir:

$$G \text{ bipartito} \Rightarrow \exists H \text{ bipartito regular con } G \subseteq H \text{ y } \Delta(G) = \Delta(H)$$

Prueba: Sean:

- $G = (V = X \cup Y, E)$ grafo bipartito
- $G^* = (V^*, E^*)$ una copia de G
- $E^\dagger = \{vv^* : d_G(v) < \Delta(G)\}$
- $\overline{G} = (\overline{V}, \overline{E})$ con:
 - $\overline{V} = V \cup V^*$
 - $\overline{E} = E \cup E^* \cup E^\dagger$

Propiedades de \overline{G} :

1. \overline{G} es bipartito, sus partes son:

- $X \cup Y^*$
- $X^\dagger \cup Y$

No existen lados entre $x \leftrightarrow x$, $y^* \leftrightarrow y^*$, $x \leftrightarrow y^*$.

2. Sea $v \in V$ tal que $d_G(v) = \Delta = \Delta(G)$ entonces V no es parte de ningún lado de E^\dagger .
 $\Rightarrow d_{\overline{G}}(v) = d_{\overline{G}}(v^*) = \Delta$

Sea $v \in V$ tal que $d_G(v) < \Delta \Rightarrow$ forma parte de un nuevo lado.

$$\therefore d_{\overline{G}}(v) = d_{\overline{G}}(v^*) = d_G(v) + 1$$

Conclusión:

$$\begin{aligned} \Delta(\overline{G}) &= \Delta(G) \\ \delta(\overline{G}) &= \delta(G) + 1 \end{aligned}$$

Repitiendo este proceso, *i.e* $G \rightarrow \overline{G} \rightarrow \overline{\overline{G}} \rightarrow \overline{\overline{\overline{G}}} \dots$ eventualmente llegaremos a un G^\blacktriangle tal que $\delta(G^\blacktriangle) = \Delta \therefore$ regular.

Fin del Lema Interno

- Sea H bipartito regular con $G \subseteq H$ y $\Delta(G) = \Delta(H)$. Como H es bipartito regular $\Rightarrow \exists$ un matching perfecto en H , llamado M . Coloreemos todos los lados de M con **Color 1**.
- Sea $H_1 = H -$ los lados del matching M . Como M es matching perfecto, H_1 sigue siendo regular y $\delta_{H_1}(x) = \delta_H(x) - 1 = \Delta - 1$. Como H_1 es bipartito regular $\Rightarrow \exists$ un matching perfecto en H_1 , llamado M_1 . Coloreemos todos los lados de M_1 con **Color 2**.
- Sea $H_2 = H_1 -$ los lados del matching M_1 . Luego $\delta_{H_2}(x) = \delta_{H_1}(x) - 1 = \Delta - 2$.

Siguiendo así $H_{\Delta-1}$ seguirá siendo regular con $\delta_{H_{\Delta-1}}(x) = 1$. En total obtuvimos Δ matchings y Δ colores. $\therefore \chi'(H) = \Delta$

$$\Rightarrow \chi'(G) = \Delta \text{ pues } G \subseteq H.$$

2.7. Cota de Hamming

Teorema: Sea C un código de longitud n y sea $t = \lfloor \frac{\delta-1}{2} \rfloor$ la cantidad de errores que corrige, entonces:

$$|C| \leq \frac{2^n}{1 + n + \binom{n}{2} + \dots + \binom{n}{t}}$$

Prueba: Sea $A = \cup_{x \in C} B_t(x)$

Como ya dijimos C corrige t errores $\Rightarrow B_t(x) \cap B_t(y) = \emptyset \forall x, y \in C : x \neq y$.

\therefore La unión en A es disjunta y $\#A = \sum_{x \in C} \#B_t(x)$

Sea $S_r(x) = \{y \in Z_2^n : d(x, y) = r\}$

$\therefore B_t(x) = \cup_{r=0}^t S_r(x)$ y la unión es disjunta $\Rightarrow \#B_t(x) = \sum_{r=0}^t \#S_r(x)$

¿Cuánto vale $S_r(x)$?

$$\begin{aligned} y \in S_r(x) &\Leftrightarrow d(x, y) = r \\ &\Leftrightarrow |x \oplus y| = r \text{ donde defino } (x \oplus y) = \epsilon \\ &\Leftrightarrow \exists \epsilon : |\epsilon| = r : y = x \oplus \epsilon \\ &\quad \underbrace{\epsilon \in S_r(0)} \end{aligned}$$

$\therefore S_r(x) = x \oplus S_r(0) \Rightarrow \#S_r(x) = \#S_r(0)$

Luego

$$\begin{aligned} \#S_r(x) &= \#S_r(0) \\ &= \# \text{ vectores de longitud } n \text{ con } r \text{ unos} \\ &= \binom{n}{r} \end{aligned}$$

Por lo tanto

$$\begin{aligned} \#A &= \sum_{x \in C} \#B_t(x) \\ &= \sum_{x \in C} \sum_{r=0}^t \#S_r(x) \\ &= \sum_{x \in C} \sum_{r=0}^t \binom{n}{r} \\ &= \#C * \sum_{r=0}^t \binom{n}{r} \end{aligned}$$

Como $A \in Z_2^n \Rightarrow \#A \leq 2^n$, entonces:

$$\begin{aligned} \left| \sum_{r=0}^t \binom{n}{r} \right| * \#C &\leq 2^n \\ \Rightarrow \#C &\leq \frac{2^n}{\sum_{r=0}^t \binom{n}{r}} \end{aligned}$$

2.8. Teorema de la matriz de chequeo de códigos lineales

Teorema: Sea C un código lineal con matriz de chequeo H , entonces:

- a) $\delta(C) = \min \{|x| : x \in C, x \neq 0\}$ (# columnas LD de H).
- b) Si H no tiene columnas 0, ni columnas repetidas, entonces $\delta \geq 3$ y corrige al menos un error.

Prueba: Sean:

- w : mín # columnas LD de H .

- $e_i = 000 \dots \underbrace{1}_i 0 \dots 0$

- a) $\boxed{w \leq \delta}$ Sea $v \in C, v \neq 0$ y $|v| = \delta \Rightarrow v$ tiene δ unos.

Es decir; $\exists j_1, j_2 \dots j_\delta$ tal que:

$$v = e_{j_1} + e_{j_2} + \dots e_{j_\delta}$$

Como $v \in C \Rightarrow v = NU(H)$ entonces $Hv^t = 0$.

$$\begin{aligned} \Rightarrow 0 = Hv^t &= H(e_{j_1}^t + e_{j_2}^t + \dots e_{j_\delta}^t) \\ &= He_{j_1}^t + He_{j_2}^t + \dots He_{j_\delta}^t \\ &= H^{j_1} + H^{j_2} + \dots H^{j_\delta} \end{aligned}$$

$\therefore \{H^{j_1} + H^{j_2} + \dots H^{j_\delta}\}$ es LD $\Rightarrow w \leq \delta$.

$\boxed{w \geq \delta}$ Sea ahora $\{H^{i_1} + H^{i_2} + \dots H^{i_w}\}$ un conjunto LD, entonces \exists constantes $c_w \in Z_2^n$ no todas nulas, tales que:

$$c_1 H^{i_1} + c_2 H^{i_2} + \dots c_w H^{i_w} = 0$$

Sea $v = c_1 e_{i_1} + c_2 e_{i_2} + \dots c_w e_{i_w}$, $v \neq 0$ ya que dijimos que no todos los c_i eran nulos. Luego:

$$\begin{aligned} Hv^t &= H(c_1 e_{i_1}^t + c_2 e_{i_2}^t + \dots c_w e_{i_w}^t) \\ &= Hc_1 e_{i_1}^t + Hc_2 e_{i_2}^t + \dots Hc_w e_{i_w}^t \\ &= H^{e_{i_1}} + H^{e_{i_2}} + \dots H^{e_{i_w}} \\ &= 0 \end{aligned}$$

$\Rightarrow v \in C$ y $|v| = w$. Como δ es la menor distancia, tenemos que $\delta \leq |v| = w$.

Por lo tanto vale $\boxed{w \leq \delta}$ y $\boxed{w \geq \delta} \Rightarrow \boxed{w = \delta}$.

b)

- Si H no tiene columnas ceros $\Rightarrow w \leq 2$.
- Si H no tiene columnas repetidas $\Rightarrow \nexists i, j : i \neq j$ y $H^i = H^j$. Como no puede pasar que $H^i = H^j$ tampoco $H^i + H^j = 0$, es decir, $H^i + H^j \neq 0 \forall i, j : i \neq j$ por lo que \nexists conjuntos de columnas LD $\Rightarrow w \geq 3$.

Luego $\delta \geq 3 \Rightarrow t = \lfloor \frac{\delta-1}{2} \rfloor \geq 1$, es decir, C corrige al menos un error.

2.9. Teorema del polinomio generador de códigos cíclicos

Teorema: Sea C un código cíclico de dimensión k y longitud n y sea $g(x)$ su polinomio generador, entonces:

- a) C está formado por los múltiplos de $g(x)$ de grado menor a n .
- b) El grado de $g(x)$ es $n - k$.
- c) $g(x)$ divide a $1 + x^n$

Prueba:

a) Sea $v(x) \in C$, dividamos v por g . Entonces $\exists q$ y r con $gr(r) < gr(g)$ tal que:

$$\begin{aligned} v &= qg + r \\ \Rightarrow &\boxed{qg = v + r} \quad (1) \end{aligned}$$

Como $v \in C \Rightarrow gr(v) < n$ y $gr(r) < gr(g) \underbrace{\leq}_{g \in C} n$ con lo que concluimos que:

$$\boxed{gr(v + r) < n} \quad (2).$$

Luego de (1) y (2) deducimos: $\boxed{gr(qg) < n}$.

Recordemos que si $p \in C$ y $gr(p) < n$ entonces: $\boxed{p \bmod(1 + x^n) = p}$.

Por lo tanto:

$$\begin{aligned} qg \bmod(1 + x^n) &= qg \\ \text{i.e } &\boxed{q \odot g = qg} \quad (A). \end{aligned}$$

Además como $g(x) \in C \Rightarrow \boxed{q(x) \odot g(x) \in C} \quad (B).$

De (A) y (B) resulta que $qg \in C \Rightarrow v + r \in C$. Además dijimos que $v \in C$ y como C es lineal $\Rightarrow \boxed{r \in C} \quad (\dagger).$

Llamemos $\boxed{gr(r) < gr(g)} \quad (\star).$

De (\dagger) y (\star) deducimos $r = 0 \Rightarrow v = qg$.

b) Recordemos que si $v(x) \in C \Rightarrow \exists \mu : v = \mu g$. Entonces para que $gr(\mu g) < n$ debe darse que $gr(\mu) + gr(g) < n$, es decir, $gr(\mu) < n - gr(g)$.

Sea

$$C = \{v : \exists \mu, gr(\mu) < n - gr(g), v = \mu g\}$$

Entonces existe una biyección entre C y el conjunto: $\{\mu : gr(\mu) < n - gr(g)\}$.

$$\begin{aligned} \therefore |C| &= |\{\mu : gr(\mu) < n - gr(g)\}| \\ 2^k &= 2^{n-gr(g)} \\ k &= n - gr(g) \\ gr(g) &= n - k \end{aligned}$$

c) Recordemos que si g es polinomio generador, la $ROT^k(g) = x^k \odot g$.

Como $g(x) \in C \Rightarrow ROT^k(g) \in C$. Además $x^k \odot g = x^k g \text{ mod } (1 + x^n)$.

Sea $p(x)$ tal que:

$$\underbrace{x^k g}_{gr=k+(n-k)} = \underbrace{p(x)(1 + x^n)}_{gr=gr(p)+n} + \underbrace{x^k \odot g}_{gr < n}$$

Luego para que la igualdad valga, debe valer que $gr(p) = 0$, es decir, $p = 1$. Entonces:

$$\boxed{x^k g = (1 + x^n) + x^k \odot g} \quad (1)$$

pero $x^k \odot g \in C \Rightarrow \exists \mu(x) :$

$$\boxed{x^k \odot g = \mu(x)g} \quad (2)$$

De (1) y (2):

$$\begin{aligned} x^k g &= (1 + x^n) + \mu g \\ 1 + x^n &= x^k g + \mu g \\ &= (x^k + \mu)g \\ \therefore g &| 1 + x^n \end{aligned}$$

Capítulo 3

Parte C

3.1. 4-COLOR \leq_p SAT

Teorema: 4-COLOR \leq_p SAT.

Prueba: Sea G un grafo con vértices v_1, v_2, \dots, v_n , queremos construir (en tiempo polinomial) una expresión booleana B en CNF (*forma conjuntiva normal*), tal que:

$$\chi(G) \leq 4 \Leftrightarrow B \text{ es satisfacible}$$

Sea $n = \# \text{vértices de } G$, y sean las variables $x_{i,j}$ con $i = 1, 2, \dots, n$ y $j = 1, 2, 3, 4$

Sean

$$\begin{aligned} A_i &= x_{i,1} \vee x_{i,2} \vee x_{i,3} \vee x_{i,4} \\ A &= A_1 \wedge A_2 \wedge \dots \wedge A_n \end{aligned}$$

Defino

$$Q_{i,j,k,l} = \overline{x_{i,j}} \vee \overline{x_{k,l}} \quad i, k = 1 \dots n \quad j, l = 1, 2, 3, 4 \quad (1)$$

$$D_i = Q_{i,i,1,2} \wedge Q_{i,i,1,3} \wedge Q_{i,i,1,4} \wedge Q_{i,i,2,3} \wedge Q_{i,i,2,4} \wedge Q_{i,i,3,4} \quad (2)$$

$$D = D_1 \wedge D_2 \wedge \dots \wedge D_n \quad (3)$$

$$F_{i,k} = Q_{i,k,1,1} \wedge Q_{i,k,2,2} \wedge Q_{i,k,3,3} \wedge Q_{i,k,4,4} \quad (4)$$

$$F = \bigwedge_{\substack{i,k \\ v_i v_k \in E(G)}} F_{i,k} \quad (5)$$

\Rightarrow

Sea C un coloreo propio de G , con a lo sumo 4 colores. Debemos dar un asignamiento de valores a las variables $x_{i,j}$ que satisfagan B , es decir, un elemento $\vec{b} \in \mathbb{Z}_2^{4n}$ tal que $B(\vec{b}) = 1$.

Definimos

$$b_{i,j} = \begin{cases} 1 & \text{si } C(v_i) = j \\ 0 & \text{c.c} \end{cases}$$

Debemos probar $B(\vec{b}) = 1$, es decir, (1) $A(\vec{b}) = 1$, (2) $D(\vec{b}) = 1$ y (3) $F(\vec{b}) = 1$.

$$(1) \boxed{A(\vec{b}) = 1}$$

Debemos ver que $A_i(\vec{b}) = 1 \forall i$, es decir, hay que probar:

$$b_{i,1} \vee b_{i,2} \vee b_{i,3} \vee b_{i,4} = 1$$

pero

$$C(v_i) \in \{1, 2, 3, 4\} \Rightarrow \exists j \in \{1, 2, 3, 4\} \text{ tal que } C(v_i) = j \Rightarrow b_{i,j} = 1$$

$$(2) \boxed{D(\vec{b}) = 1}$$

Debemos ver que $D_i(\vec{b}) = 1 \forall i$, es decir,

$$Q_{i,j,l}(\vec{b}) = 1 \forall i, j, l, j < l$$

es decir:

$$\overline{b_{i,j}} \vee \overline{b_{i,l}} = 1 \forall i, j, l, j < l$$

por el absurdo, supongamos que no:

$$\begin{aligned} \Rightarrow \exists i, j, l, j < l : \overline{b_{i,j}} \vee \overline{b_{i,l}} &= 0 \\ \Rightarrow \overline{b_{i,j}} = \overline{b_{i,l}} &= 0 \\ \Rightarrow b_{i,j} = b_{i,l} &= 1 \\ \Rightarrow C(v_i) = j, C(v_i) = l, \text{ pero } j \neq l &\text{ Absurdo!} \end{aligned}$$

Luego, el absurdo vino de suponer que $\overline{b_{i,j}} \vee \overline{b_{i,l}} = 0$

$$\therefore \overline{b_{i,j}} \vee \overline{b_{i,l}} = 1$$

$$(3) \boxed{F(\vec{b}) = 1}$$

Por el absurdo, supongamos que no:

$$\begin{aligned} \Rightarrow \exists i, k : v_i v_k \in E(G) \text{ tal que } F_{i,k}(\vec{b}) &= 0 \\ \Rightarrow \exists j \in \{1, 2, 3, 4\} : Q_{i,k,j}(\vec{b}) &= 0 \\ \Rightarrow \overline{b_{i,j}} \vee \overline{b_{k,j}} &= 0 \\ \Rightarrow \overline{b_{i,j}} = \overline{b_{k,j}} &= 0 \\ \Rightarrow b_{i,j} = b_{k,j} &= 1 \\ \Rightarrow C(v_i) = 1 \text{ y } C(v_k) &= 1 \\ \Rightarrow C(v_i) = C(v_k) (= j) &\text{ Absurdo!} \end{aligned}$$

Pues, el coloreo C es propio y $v_i v_k \in E(G)$. El absurdo vino de suponer que $F(\vec{b}) = 0$

$$\therefore F(\vec{b}) = 1$$

$$\boxed{\Leftarrow}$$

Ahora sabemos que existe un \vec{b} con $B(\vec{b}) = 1$ y debemos construir un coloreo propio, con al menos 4 colores.

$$B(\vec{b}) = 1 \Rightarrow \begin{cases} A(\vec{b}) = 1 \Rightarrow A_i(\vec{b}) = 1 \forall i \Rightarrow (1) \forall i \exists j : b_{i,j} = 1 \\ D(\vec{b}) = 1 \Rightarrow D_i(\vec{b}) = 1 \forall i \Rightarrow (2) \forall i \nexists j \neq l : b_{i,j} = 1 = b_{i,l} \\ F(\vec{b}) = 1 \Rightarrow \forall i, j \in E(G), C(v_i) \neq C(v_j) \Rightarrow C \text{ es propio} \end{cases}$$

De (1) y (2) $\Rightarrow \boxed{\forall i \exists ! j : b_{i,j} = 1}$

Luego, definimos:

$$C(v_i) = \text{unico } j \text{ con } b_{i,j} = 1$$

3.2. 3-SAT es NP-Completo

Teorema: 3-SAT es NP-Completo.

Prueba: Probaremos que $\text{SAT} \leq_p \text{3-SAT}$, pues 3-SAT es NP Completo.

Sea

$$B = D_1 \wedge D_2 \dots \wedge D_n \text{ con variables } x_1, \dots, x_n \quad (6)$$

$$D_i = L_{i,1} \vee L_{i,2} \dots \vee L_{i,k_i} \quad (7)$$

$$l_{i,j} = \text{literales} \quad (8)$$

Para cada D_i construiremos E_i , que serán conjunciones de disyunciones de 3 literales, con variables extras, y luego tomaremos:

$$B' = E_1 \wedge E_2 \dots \wedge E_n$$

Construcción:

- Si D tiene 3 literales, es decir, $K_i = 3 \Rightarrow E_i = D_i$
- Si D tiene 2 literales, es decir, $K_i = 2 \Rightarrow$ tomo una variable extra $y_{i,1}$ y defino:

$$E_i = (l_{i,1} \vee l_{i,2} \vee y_{i,1}) \wedge (l_{i,1} \vee l_{i,2} \vee \overline{y_{i,1}})$$

- Si D tiene 1 literales, es decir, $K_i = 1 \Rightarrow$ tomo dos variables extras $y_{i,1}$ y $y_{i,2}$ y defino:

$$E_i = (l_{i,1} \vee y_{i,1} \vee y_{i,2}) \wedge (l_{i,1} \vee \overline{y_{i,1}} \vee y_{i,2}) \wedge (l_{i,1} \vee y_{i,1} \vee \overline{y_{i,2}}) \wedge (l_{i,1} \vee \overline{y_{i,1}} \vee \overline{y_{i,2}})$$

- Si D tiene más de 3 literales, es decir, $K_i > 4 \Rightarrow$ agrego k_{i-3} variables extras y defino:

$$E_i = (l_{i,1} \vee l_{i,2} \vee y_{i,1}) \wedge (l_{i,3} \vee \overline{y_{i,1}} \vee y_{i,2}) \wedge (l_{i,4} \vee \overline{y_{i,2}} \vee y_{i,3}) \wedge \dots (l_{i,k_{i-2}} \vee \overline{y_{i,k_{i-4}}} \vee y_{i,k_{i-3}}) \wedge (l_{i,k_{i-1}} \vee l_{i,k_i} \vee \overline{y_{i,k_{i-3}}})$$

Queremos ver:

$$B(\vec{b}) = 1 \Leftrightarrow \exists \vec{a} : B'(\vec{b}, \vec{a}) = 1$$

$\boxed{\Leftarrow}$

Por el absurdo. Supongamos que no, entonces $B(\vec{b}, \vec{a}) = 1$ pero $B(\vec{b}) = 0$.

Si $B(\vec{b}) = 0 \Rightarrow \exists i : D_i(\vec{b}) = 0$ pero $E_i(\vec{b}, \vec{a}) = 1$.

Casos:

- si $k_i = 3$ Absurdo! pues $D_i = E_i$

- si $k_i = 2$

$$\begin{aligned}
1 &= E_i(\vec{b}, \vec{a}) \\
&= \underbrace{(l_{i,1}(\vec{b}) \vee l_{i,2}(\vec{b}) \vee a_{i,1})}_{=0} \wedge \underbrace{(l_{i,1}(\vec{b}) \vee l_{i,2}(\vec{b}) \vee \overline{a_{i,1}})}_{=0} \\
&= a_{i,1} \wedge \overline{a_{i,1}} \\
&= 0 \text{ Absurdo!}
\end{aligned}$$

- si $k_i = 1$

$$\begin{aligned}
1 &= E_i(\vec{b}, \vec{a}) \\
&= (a_{i,1} \vee a_{i,2}) \wedge (\overline{a_{i,1}} \vee a_{i,2}) \wedge (a_{i,1} \vee \overline{a_{i,2}}) \wedge (\overline{a_{i,1}} \vee \overline{a_{i,2}}) \\
&= 0 \text{ Absurdo!}
\end{aligned}$$

- si $k_i > 4$

$$D_i(\vec{b}) = 0 \Rightarrow l_{i,j}(\vec{b}) = 0 \forall j.$$

$$\begin{aligned}
1 &= E_i(\vec{b}, \vec{a}) \\
&= a_{i,j} \wedge (\overline{a_{i,1}} \vee a_{i,2}) \wedge (\overline{a_{i,2}} \vee a_{i,3}) \wedge \dots (\overline{a_{i,k_i-4}} \vee a_{i,k_i-3}) \wedge \overline{a_{i,k_i-3}} \\
&= a_{i,1} \wedge (a_{i,1} \Rightarrow a_{i,2}) \wedge (a_{i,2} \Rightarrow a_{i,3}) \wedge \dots (a_{i,k_i-4} \Rightarrow a_{i,k_i-3}) \wedge \overline{a_{i,k_i-3}} \text{ Absurdo!}
\end{aligned}$$

Pues tendría $a_{i,1} = a_{i,2} = \dots a_{i,k_i-3} = 1$ pero $\overline{a_{i,k_i-3}} = 1$.

\Rightarrow

Casos:

- Si $k_i = 3 \Rightarrow E_i(\vec{b}, \vec{a}) = D_i(\vec{b}) = 1$
- Si $k_i = 1$ ó $k_i = 2$ defino $a_{i,j} = 0$, luego $E_i(\vec{b}, 0) = 1$
- Si $k_i > 4 \Rightarrow D_i(\vec{b}) = 1 \Rightarrow \exists j : l_{i,j}(\vec{b}) = 1$

Definimos

$$\begin{aligned}
a_{i,1} &= a_{i,2} = \dots a_{i,j-2} = 1 \\
a_{i,j-1} &= a_{i,j} = \dots a_{i,k_i-3} = 0
\end{aligned}$$

Luego

$$\begin{aligned}
E_i(\vec{b}, \vec{a}) &= (l_{i,1}(\vec{b}) \vee l_{i,2}(\vec{b}) \vee a_{i,1}) \\
&\wedge (l_{i,3}(\vec{b}) \vee \overline{a_{i,1}} \vee a_{i,2}) \\
&\vdots \\
&\wedge (l_{i,j-1}(\vec{b}) \vee \overline{a_{i,j-3}} \vee a_{i,j-2}) \\
&\wedge (l_{i,j}(\vec{b}) \vee \overline{a_{i,j-2}} \vee a_{i,j-1}) \\
&\wedge (l_{i,j+1}(\vec{b}) \vee \overline{a_{i,j-1}} \vee a_{i,j}) \\
&\vdots \\
&\wedge (l_{i,k_i-1}(\vec{b}) \vee l_{i,k_i}(\vec{b}) \vee \overline{a_{i,k_i-3}}) \\
&\therefore E_i = 1
\end{aligned}$$

3.3. 3-COLOR es NP-Completo

Teorema: 3-COLOR es NP-Completo

Prueba: Veremos que $3 - SAT \leq_p 3 - Color$, es decir, dado B en CNF con 3 literales por disyunción, debemos crear un grafo G, tal que:

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

Sea:

$$\begin{aligned} B &= D_1, D_2 \dots D_m \text{ con variables } \{x_1 \dots x_m\} \\ D_i &= l_{i,1} \vee l_{i,2} \vee l_{i,3} \end{aligned}$$

Construcción del grafo:

- Dado un literal l , definimos:

$$\psi(l) = \begin{cases} v_k & \text{si } l = x_k \\ w_k & \text{si } l = \overline{x_k} \end{cases}$$

- Vértices:

$$V(G) = \{s, t\} \cup \{v_1 \dots v_n, w_1 \dots w_n\} \cup \{\mu_{i,1}, \mu_{i,2}, \mu_{i,3}, b_{i,1}, b_{i,2}, b_{i,3}\}_{i=1}^m$$

- Aristas:

$$E(G) = \{st\} \cup \{tv_i, tw_i, v_i w_i\}_{i=1}^n \cup \{s\mu_{i,j}\}_{i=1, j=1,2,3}^m \cup F$$

$$\{b_{i,1}b_{i,2}, b_{i,1}b_{i,3}, b_{i,2}b_{i,3}, b_{i,1}\mu_{i,1}, b_{i,2}\mu_{i,2}, b_{i,3}\mu_{i,3}\}_{i=1}^m$$

Donde:

$$F = \{\mu_{i,j}\psi(l_{i,j})\}_{i=1, j=1,2,3}^m$$

$$G \text{ tiene: } \left\{ \begin{array}{l} 2 + 2n + 6m \text{ vertices} \\ 1 + 3n + 6m + 3m + 3m \text{ aristas} \end{array} \right\} \Rightarrow G \text{ es polinomial}$$



Como G tiene triángulos, si $\chi(G) \leq 3$, entonces debe ser $\chi(G) = 3$. Por lo tanto, existe un coloreo C de G con 3 colores.

Definición:

$$b_k = \begin{cases} 1 & \text{si } C(v_k) = C(s) \\ 0 & \text{si } C(v_k) \neq C(s) \end{cases}$$

Para probar $B(\vec{b}) = 1$ debemos probar que $D_i(\vec{b}) = 1 \forall i$.

Sea $i \in \{1, 2, \dots, m\}$, como $\{b_{i,1}, b_{i,2}, b_{i,3}\}$ es un triángulo, entonces deben aparecer los 3 colores.

Por lo tanto: $\exists j : C(b_{i,j}) = C(s)$

Luego:

$$\left. \begin{array}{l} (1) \mu_{i,j}b_{i,j} \in E(G) \Rightarrow C(\mu_{i,j}) \neq C(b_{i,j}) = C(s) \\ (2) \mu_{i,j}s \in E(G) \Rightarrow C(\mu_{i,j}) \neq C(s) \\ (3) st \in E(G) \Rightarrow C(s) \neq C(t) \end{array} \right\} \Rightarrow C(\mu_{i,j}) = \text{TERCER COLOR}$$

Por otro lado:

$$\left. \begin{array}{l} (1) \mu_{i,j}\psi(l_{i,j}) \in E(G) \Rightarrow C(\psi(l_{i,j}) \neq \text{TERCER COLOR}) \\ (2) \psi(l_{i,j})t \in E(G) \Rightarrow C(\psi(l_{i,j}) \neq C(t)) \end{array} \right\} \Rightarrow C(\psi(l_{i,j}) = C(s))$$

■ Caso (1):

$$\begin{aligned} l_{i,j} = x_k &\Rightarrow l_{i,j}(\vec{b}) = b_k \\ &\Rightarrow \psi(l_{i,j}) = v_k \end{aligned}$$

Entonces:

$$\begin{aligned} C(v_k) &= C(s) \Rightarrow b_k = 1 \\ \therefore l_{i,j}(\vec{b}) &= 1 \Rightarrow D_i(\vec{b}) = 1 \end{aligned}$$

■ Caso (2):

$$\begin{aligned} l_{i,j} = \overline{x_k} &\Rightarrow l_{i,j}(\vec{b}) = \overline{b_k} \\ &\Rightarrow \psi(l_{i,j}) = w_k \end{aligned}$$

Entonces:

$$\begin{aligned} \left. \begin{array}{l} C(w_k) = C(s) \\ v_k w_k \in E(G) \end{array} \right\} &\Rightarrow C(v_k) \neq C(s) \therefore b_k = 0 \\ \therefore b_k = 1 &\Rightarrow D_i(\vec{b}) = 1 \end{aligned}$$

\Rightarrow

Acá asumimos que $\exists \vec{b} : B(\vec{b}) = 1 \Rightarrow \forall i, D(\vec{b}) = 1 \Rightarrow \boxed{\forall i \exists j : l_{i,j}(\vec{b}) = 1} (\star)$. Debemos construir un coloreo propio con 3 colores.

Definimos:

$$\left. \begin{array}{l} C(s) = \text{BLANCO} \\ C(t) = \text{AZUL} \end{array} \right\} \Rightarrow st \text{ No Crea Problemas (NCP)}$$

$$C(v_k) = \left\{ \begin{array}{ll} \text{BLANCO} & \text{si } b_k = 1 \\ \text{NEGRO} & \text{si } b_k = 0 \end{array} \right\} \Rightarrow \underbrace{v_k}_{B \text{ o } N} \underbrace{w_k}_{B \text{ o } N} \text{ NCP}$$

$$C(w_k) = \left\{ \begin{array}{ll} \text{NEGRO} & \text{si } b_k = 1 \\ \text{BLANCO} & \text{si } b_k = 0 \end{array} \right\} \Rightarrow v_k t \text{ y } w_k t \text{ NCP}$$

Falta colorear las garras. Dado i , tomemos el j de (\star) y definimos:

$$C(\mu_{i,r}) = \left\{ \begin{array}{ll} \text{NEGRO} & \text{si } r = j \\ \text{AZUL} & \text{si } r \neq j \end{array} \right\} \Rightarrow \underbrace{\mu_{i,r}}_{N \text{ o } A} \underbrace{s}_B \text{ NCP } \forall r$$

■ Caso $r \neq j$: $\underbrace{\mu_{i,r}}_A \underbrace{\psi(l_{i,r})}_{B \text{ o } N} \Rightarrow \text{NCP}$

■ Caso $r = j$:

- Caso (1):

$$\begin{aligned}
l_{i,j} = x_k &\Rightarrow l_{i,j}(\vec{b}) = b_k = 1 \Rightarrow C(v_k) = \text{BLANCO} \\
&\Rightarrow \psi(l_{i,j}) = v_k \Rightarrow \mu_{i,j} \psi(l_{i,j}) = \underbrace{\mu_{i,j}}_N \underbrace{v_k}_B \text{ NCP}
\end{aligned}$$

- Caso (2):

$$\begin{aligned}
l_{i,j} = \overline{x_k} &\Rightarrow l_{i,j}(\vec{b}) = \overline{b_k} = 1 \Rightarrow b_k = 0 \Rightarrow C(w_k) = \text{BLANCO} \\
&\Rightarrow \psi(l_{i,j}) = w_k \Rightarrow \mu_{i,j} \psi(l_{i,j}) = \underbrace{\mu_{i,j}}_N \underbrace{w_k}_B \text{ NCP}
\end{aligned}$$

Quedan las bases:

$$C(b_{i,r}) = \begin{cases} \text{AZUL} & \text{si } r = j \\ \text{BLANCO} & \text{si } r \neq j \\ \text{NEGRO} & \text{si } r \neq j \end{cases}$$

Entonces:

- $\{b_{i,1}, b_{i,2}, b_{i,3}\}$ NCP
- $\underbrace{b_{i,j}}_A \underbrace{\mu_{i,j}}_N$ NCP
- $\therefore \underbrace{b_{i,r}}_{B \text{ o } N} \underbrace{\mu_{i,r}}_A$ NCP

Bibliografía

- [1] MAXIMILIANO ILLBELE, «Resumen de Discreta II, 16 de agosto de 2012», *FaMAF, UNC*.
- [2] LUCIA PAPPATERRA, «Resumen de Discreta II, 2014», *FaMAF, UNC*.
- [3] AGUSTÍN CURTO, «Carpeta de Clase, 2016», *FaMAF, UNC*.