Resumen de teoremas para el final de Matemática Discreta II

Agustin Curto, agucurto95@gmail.com

2016

Índice general

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: $f_0, f_1, f_2, \ldots$ 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.

Observaciónes:

1. - $d_k(s)=0$
   - $b_k(t)=0$
2. Sabemos que las distancias de $\langle E-K\rangle$ no disminuyen en pasos sucesivos, como esto será útil para esta demostración llamaremos $\circledast$ a la demostración de:

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

Llamemos <u>*crítico*</u> 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: $s \ldots$ x,y $\ldots$ t se usa para construir f_k.

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

Para que xy pueda ser *crítico* nuevamente debe ser usado en la otra dirección (*i.e* yx). Sea j el paso posterior a k en el cual se usa el lado en la otra dirección, el camino $s \cdots$ y,x $\cdots$ t se usa para construir f_j.

$$\begin{aligned} d_j(t) &= d_j(x) + b_j(x) \\ &= d_j(y) + 1 + b_j(x) \end{aligned} \tag{2}$$

Entonces:

$$\text{De (1) y (2)} \Rightarrow \begin{cases} d_j(x) = d_j(y) + 1 \ \star \\ d_k(y) = d_k(x) + 1 \ \dagger \end{cases}$$

Luego:

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

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

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

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

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

Teorema: Sean: $f_0, f_1, f_2, \ \ldots$ 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. $b_k(x) \leq b_{k+1}(x)$

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

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

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

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

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

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.

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

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

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

   Sea $\boxed{x = z_i}$

   - <u>Caso 1:</u> Existe algun camino aumentante, relativo a f_{k-1} de la forma $s,\ \ldots\ x,\ z$. $\Rightarrow \boxed{d_k(x) \leq d_k(x) + 1}$

     Pues al haber un camino $\underbrace{s,\ \ldots\ x,}_{d_k(x)}$ z, llamemosle A, de longitud $d_k(x) + 1$ entre s y z, sabemos que el minimo de todos los caminos de s a z seran $\leq$ 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 zx está vacío relativo a f_{k-1}. Para construir f_k usamos un camino de la forma $s,\ \ldots\ z,\ x$. Es decir:

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

     Como $\langle E - K\rangle$ funciona con BFS, ese camino usado pra construir f_k debe ser de longitud mínima. Es decir: $\quad d_k(x) = d_k(z) + 1$

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

   <u>Conclusión:</u> En cualquiera de los dos casos tenemos:

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

Ahora bien: $\quad d_{k+1}(x) = d_{k+1}(z_i) = i \;\Rightarrow\;$ vale H(i) para x.
$\therefore d_k(z) \leq d_{k+1}(x)$

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

Por lo tanto:

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

1.3. La complejidad de DINIC

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

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.

Complejidad(Dinic)= $\mathcal{O}(n)*$ Compl(Hallar un flujo bloqueante en un NA con Dinic)

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

Sea:

- A = Avanzar()
- R = Retroceder()
- I = Incrementar_Flujo + Inicialización ($\mathcal{O}(1)$)

Una corrida de Dinic luce como:

AA . . . AIAAARA . . . AIAARAAARR . . . IA . . .

Dividamos la corrida en subpalabras del tipo:

$* \underbrace{AA\ \ldots\ A}_{Todas A's} I$

$* \underbrace{AA\ \ldots\ A}_{Todas A's} R$

Nota: el número de A's puede ser 0.
Debemos determinar:

1. Cual es la complejidad de cada subpalabra.

2. Cuantas palabras hay de cada tipo.

Complejidad de cada subpalabra
Recordemos que:

$$\text{A:} \left[\begin{array}{l} P[i+1] = \text{algún elemento de } \Gamma^+(P[i]) \\ i = i+1 \end{array} \right.$$

$$\Rightarrow \text{A es } \mathcal{O}(1)$$

$$\text{R:} \left[\begin{array}{l} P[i+1] = \text{borrar } P[i-1]P[i] \text{ del NA} \\ i = i-1 \end{array} \right.$$

$$\Rightarrow \text{R es } \mathcal{O}(1)$$

$$\text{I:} \left[\begin{array}{l} P[i+1] = \text{recorre 2 veces, un camino de longitud } d = d(t) \end{array} \right.$$

$$\Rightarrow \text{I es } \mathcal{O}(d)$$

Por lo tanto:

$$\begin{aligned} Compl(\underbrace{A\ \dots\ A}_{j\ veces} R) &= \underbrace{\mathcal{O}(1) +\ \dots\ \mathcal{O}(1)}_{j\ veces} + \mathcal{O}(1) \\ &= \mathcal{O}(j) + \mathcal{O}(1) \\ &= \mathcal{O}(j) \end{aligned}$$

Pero como cada A hace $i = i+1$ y tenemos $0 \leq i \leq d \;\Rightarrow\; j \leq d$.

$$\therefore\ Compl(A\ \dots\ AR) = \mathcal{O}(d)$$

Similarmente:

$$\begin{aligned} Compl(A\ \dots\ AI) &= \underbrace{\mathcal{O}(1) +\ \dots\ \mathcal{O}(1)}_{\leq d\ veces} + \mathcal{O}(1) \\ &= \mathcal{O}(d) + \mathcal{O}(1) \\ &= \mathcal{O}(d) \end{aligned}$$

Cantidad de subpalabras

- R tiene la instrucción "**borrar lado**". Como los lados borrados quedan borrados hay a lo sumo m R's, es decir:

$$\therefore\ \#(A\ \dots\ AR's) \leq m$$

- I tiene también línes de la forma:
  if ... **then**
  borrar lado
  end if

  Lo que está dentro del **if** se cumple al menos una vez, es decir:

$$\therefore\ \#(A\ \dots\ AI's) \leq m$$

Este análisis muestra que:

$$\therefore \ \#(A \ \ldots \ AR's) + \#(A \ \ldots \ AI's) \leq m$$

Por lo tanto hay $\leq m$ palabras, cada una de complejidad $\mathcal{O}(d)$.

$$\begin{aligned} \therefore \ Compl(Paso\ Bloqueante) &= \mathcal{O}(m) + \mathcal{O}(md) \\ &= \mathcal{O}(mn) \end{aligned}$$

ya que $d \leq n$.

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.

Complejidad(Wave)= $\mathcal{O}(n)*$ Compl(Hallar un flujo bloqueante en un NA con 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 **forwrdbalance** (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 $\mathcal{O}(1)$.

$$\therefore \ Compl(Paso\ Bloqueante) = \#\text{"}procesamientos\text{"} \text{ de lados}$$

Los "procesamientos" de lados los podemos dividir en dos categorías:

1. Aquellos procesamientos que saturan o vacian el lado. Denotaremos "T" al número de estos procesamientos.
2. Aquellos procesamientos que no saturan ni vacian el lado. Denotaremos "Q" al número de estos procesamientos.

Por lo tantos queremos acotar $T + Q$.

Complejidad de T:

- ¿Puede un lado xy saturado volver a saturarse?

  Para poder volver a saturarse primero tiene que vaciarse auque 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.

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

Conclusión 1: Los lados se saturan solo una vez.

- ¿Puede un lado 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.

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

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

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

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 $\leq$ # Total de FB's y BB's

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

  $\therefore$ Total de FB's y BB's $\leq 2\,n$ #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 balaceados 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:

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

un camino dirigido en A^*.

Ese camino $\boxed{\text{No existe}}$ 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 $\exists\, i \;:\; x_i \not\ni V(A)$ entonces:

$$\begin{array}{rcl} d(t) & \leq & d(x_i) \\ & \leq & d^*(x_i) \\ & = & i < r \\ & = & d^*(t) \end{array}$$

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

2. Están todos los vértices pero falta una arista, es decir $\exists\, i \;:\; x_i x_{i+1} \not\ni E(A)$.

   a) $x_i x_{i+1}$ no está porque corresponde a un lado vacío o saturado en NA, es decir $x_i x_{i+1}$ no está en el recidual 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 cocluir, por la prueba de $\langle E - K \rangle$ que:

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

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

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

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

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

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

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

   ($\dagger$): Ya que $s,\ x_1,\ \ldots\ x_r$

Capítulo 2

Parte B

2.1. 2-COLOR es polinomial

2.2. Teorema Max-Flow Min-Cut

<u>Teorema:</u>

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

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

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

<u>Prueba:</u> Demostraremos primero que $V(f) = f(S,\overline{S}) - f(\overline{S},S)$ donde f es un flujo y S un corte. Esto nos ayudará en la demostración del ítem a).

<u>Observemos que:</u>

- $f(A \cup B, C) = f(A,C) + f(B,C)$: A y B disjuntos.
- $f(A, B \cup C) = f(A,B) + f(A,C)$: B y C disjuntos.
- $f(A,B) = \sum_{\substack{x\in A\\ y\in B}} f(x,y)$.

Sea x $\in$ S $\Rightarrow$ x $\neq$ t.

$$\text{f(x, V) - f(V, x)} = \begin{cases} V(f) & Si\ x = s \\ 0 & Si\ x \neq s\, pues\ t \notin S \end{cases}$$

Luego:

$$\begin{aligned} \sum_{x\in S}(f(x,V) - f(V,x)) &= 0 + 0 \cdots + V(f) \\ &= V(f) \end{aligned}$$

$$\begin{aligned} V(f) &= \sum_{x\in S} f(x,V) - \sum_{x\in S} f(V,x) & \\ &= f(S,V) - f(V,S) & \text{Por observación} \\ &= f(S, S\cup\overline{S}) - f(S\cup\overline{S}, S) & \text{Ya que } V = S\cup\overline{S} \\ &= f(S,S) + f(S,\overline{S}) - f(S,S) - f(\overline{S},S) & \text{Por observación} \\ &= \boxed{f(S,\overline{S}) - f(\overline{S},S)}\ (\star) & \end{aligned}$$

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

$$V(f) \overset{Por\ (\star)}{=} \underbrace{f(S,\overline{S}) - \underbrace{f(\overline{S},S)}_{\geq 0}}_{\leq 0}$$

$$\Rightarrow V(f) \leq f(S,\overline{S}) \leq C(S,\overline{S}) = \text{Cap(S)}$$

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

Supongamos que V(f) = Cap(S). Sea g un flujo cualquiera y T un corte cualquiera.

- $V(g) \overset{Por\ a)}{\leq} Cap(S) = V(f) \Rightarrow$ f es maximal
- $Cap(T) \overset{Por\ a)}{\leq} V(f) = Cap(S) \Rightarrow$ S es minimal

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

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

¿$t \in S$?
<u>Si t estuviera en S:</u> existiría un camino aumentante entre s y t.
Por el teorema del camino aumentante podemos construir un flujo g tal que:

$$V(g) = V(f) + \epsilon \text{ para algun } \epsilon > 0$$
$$\therefore V(g) > V(f) \textbf{ Absurdo} \text{ pues } f \text{ es maximal.}$$

$\therefore t \notin S \Rightarrow S$ es corte.

Solo resta ver que: $V(f) = Cap(S)$

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

Analicemos (1) y (2)

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

$x \in S \Rightarrow \exists$ camino aumentante $x_0 = s,\ x_1, \dots x_r = x$.

$y \in \overline{S} \Rightarrow \nexists$ camino aumentante entre s y y. En particular $x_0 = s, x_1 = x, \dots y$ no es camino aumentante.

$$\Rightarrow f(\overrightarrow{xy}) = Cap(\overrightarrow{xy}) \qquad \forall x \in S, \forall y \in \overline{S} : \overrightarrow{xy} \in E.$$

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

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

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

$y \in S \Rightarrow \exists$ camino aumentante $y_0 = s,\ y_1, \dots y_r = y$.

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

$\Rightarrow f(\overline{S}, S) = 0$

Luego de (1) y (2)

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

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:

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

   Idem para las columnas.

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

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

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

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

   Si lo hacems 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:

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

Referencias:

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

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

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:

$$\text{Complejidad(1 Extensión)} = \underbrace{\mathcal{O}(n)}_{\#CM} * \underbrace{\mathcal{O}(n^2)}_{Compl(CM)} + \underbrace{\mathcal{O}(n^2)}_{Busqueda\ n\ filas\ x\ n\ columnas}$$

$$\therefore \text{Complejidad(Húngaro)} = \underbrace{\mathcal{O}(n^2)}_{Matching\ inicial} + \underbrace{\mathcal{O}(n)}_{\#extensiones} * \underbrace{\mathcal{O}(n^3)}_{Compl(extension)}) = \mathcal{O}(n^4)$$

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 oberservemos que M induce una función inyectiva de X a Y.

$$f(x) = \text{único y : xy} \in \text{M}.$$

1. Si $S \subseteq X \Rightarrow |S| = |\Gamma(S)|$.

   Además por definición de f, $f(x) \in \Gamma(x)$.

2. Si $x \in S \Rightarrow f(x) \in \Gamma(S) \Rightarrow f(S) \subseteq \Gamma(S)$.

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 $\Rightarrow \exists\ S \subseteq X : |S| > |\Gamma(S)|$.

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

Sean:

- S_0 = filas sin matchear.
- $T_1 = \Gamma(S_0)$, 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 = filas etiquetadas por las columnas de T_1.
- $T_2 = \Gamma(S_1) - T_1$, columnas etiquetadas por las filas de S_1.

En general:

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

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. $\Gamma(S_0 \cup S_1 \cup \ldots S_j) = T_1 \cup T_2 \cup \ldots T_{j+1}$
   Por inducción en j:
   - Caso Base: $j = 0$ vale.
   - Caso Inductivo: Supongamos que vale para j, veamos para *j+1*.

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

3. Por construcción, los S_i y T_i son todos distintos.
   Sea $S = S_0 \cup S_1 \cup \ldots S_k$

$$
\begin{aligned}
|\Gamma(S)| &= |\Gamma(S_0 \cup S_1 \cup \ldots S_k)| \\
&= |T_1 \cup T_2 \cup \ldots \underbrace{T_{k+1}}_{=\emptyset}| \qquad \text{Por 3} \\
&= |T_1 \cup T_2 \cup \ldots T_k| \\
&= |T_1| + |T_2| + \ldots |T_k| \qquad \text{Por 2} \\
&= |S_1| + |S_2| + \ldots |S_k| \qquad \text{Por 1} \\
&= |S| - |S_0| \qquad \text{Por 2} \\
&< |S| \qquad \text{Absurdo!}
\end{aligned}
$$

2.5. Teorema del matrimonio

Teorema: Todo grafo regular bipartito tiene un matching perfecto.
Prueba: Sea $G = (X \cup Y, E) bipartito regular \forall w \subseteq V(G)$, definimos:

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

Como G es regular, $\exists\ \Delta = \delta > 0 : d(z) = \Delta \forall\ z \Rightarrow \forall w \in\ W \exists$ al menos un lado wv.

Supongamos que $w \subseteq X$ (de igual forma para $w \subseteq Y$)

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

Es decir, a cada w le corresponden Δ lados distintos. En particular $|E_x| = \Delta * |x|$ y $|E_y| = \Delta * |y|$ pero $E_x = E = E_y$ pues G es bipartito.

Por lo tanto:
$|E_x| = |E_y| \Rightarrow \Delta * |x| = \Delta * |y| \Rightarrow |x| = |y|$ $\therefore$ Todo matching completo es perfecto.
Basta ver que existe un matching comple de X a Y, *i.e* que se cumple la condición de Hall.

Sea $S \subseteq X$:

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

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. 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:

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

Prueba: Sean:

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

Propiedades de $\overline{G}$:

1. $\overline{G}$ es bipartito, sus partes son:
   - $X \cup Y^{\star}$
   - $X \cup Y$

   No existen lados entre $x \leftrightarrow x,\ y^{\star} \leftrightarrow y^{\star},\ x \leftrightarrow y^{\star}$.

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

   Sea v $\in$ V tal que $d_G(v) < \Delta \Rightarrow$ forma parte de un nuevo lado.
   $\therefore d_{\overline{G}}(v) = d_{\overline{G}}(v^{\star)} = d_G(v) + 1$
   Conclusión: $\Delta(\overline{G}) = \Delta(G)\ y\ \delta(\overline{G}) = \delta(G) + 1$

   Repitiendo este proceso, $i.e\ G \to \overline{G} \to \overline{\overline{G}} \to \overline{\overline{\overline{G}}} \ldots$ evenatualmente 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 $\overset{Por\ 1.}{\Rightarrow} \exists$ un matching perfecto en H, llamado M. Coloriemos 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 $\overset{Por\ 1.}{\Rightarrow} \exists$ un matching perfecto en H_1, llamado M_1. Coloriemos 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. $\therefore \chi'(H) = \Delta$

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

2.7. Teorema 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 corrigue, entonces:

$$
\begin{aligned}
|C| &\leq \frac{2^n}{1 + n + \binom{n}{2} + \ldots \binom{n}{t}} \\
&= \sum_{i=1}^{t} \binom{n}{i}
\end{aligned}
$$

Prueba:

2.8. Sea H una matriz de chequeo de un código C, pruebe que:

2.8.1. $\delta(C) =$ mínimo número de columnas linealmente dependientes de H

2.8.2. Si H no tiene la columna cero ni columnas respetidas $\Rightarrow$ C corrige al menos un error

2.9. Sea C un código cíclico de dimensión k y longitud n y sea $g(x)$ su polinomio generador, probar que:

2.9.1. C está formado por los múltiplos de $g(x)$ de grado menor a n

2.9.2. El grado de $g(x)$ es $n - k$

2.9.3. $g(x)$ divide a $1 + x^n$

Capítulo 3

Parte C

3.1. 4-COLOR $\leq_p$ SAT

3.2. 3-SAT es NP-Completo

3.3. 3-COLOR es NP-Completo

Bibliografía

[1] Curto Agustín , «Matemática Discreta II, apuntes de clase», *FaMAF, UNC*.

[2] Maximiliano Illbele, «Resumen de Discreta II, 16 de agosto de 2012», *FaMAF, UNC*.