Teoremas Matemática Discreta 2

(1) Probar que 3-COLOR es NP-completo

Veremos que 3-SAT $\leq_p$ 3-COLOR. Para ello, dada una instancia *B* de 3-SAT, es decir, una expresión booleana en CNF con exactamente 3 literales en cada disjunción, crearemos polinomialmente una instancia *G* 3-COLOR, es decir, un grafo *G*, tal que *B* sea satisfacible si y sólo si $\chi(G) \leq 3$

Prueba:

Definimos *G* como un triángulo si *B* es satisfactible y como K_4 si no.

Supongamos que las variables de *B* son x_1,....., x_n., y que *B* = $D_1 \wedge \ldots \wedge D_m$., con las disjunciones $D_j = l_{1,j} \vee l_{2,j} \vee l_{3,j}$ ($l_{k,j}$ literales). Construiremos el grafo *G* .dando sus vértices y sus lados.

Los vértices son:

1) 2n vértices v_l, uno por cada literal l , es decir, por cada variable x_i hay dos vértices v_x y $v_{\bar{x}i}$

2) 6 m vértices $\{e_{k,j}\}_{k=1,2,3\,;\,j=1,\ldots,m} \cup \{a_{k,j}\}_{k=1,2,3\,;\,j=1,\ldots,m}$, es decir, para cada
j = 1, 2, …., m, seis vértices $e_{1,j}$, $e_{2,j}$,$e_{3,j}$, $a_{1,j}$,$a_{2,j}$,$a_{3,j}$

3) Dos vértices especiales, *s* y *t*.

Los lados son:

1) El lado *st*

2) 2n lados tv_l, uno por cada literal l

3) n lados $v_x v_{\bar{x}}$, uno por cada variable x

4) 3m lados $e_{k,j}$, $a_{k,j}$, (k = 1, 2, 3, j = 1, 2, ….., m)

5) 3m lados $a_{1,j}$, $a_{2,j}$; $a_{1,j}$, $a_{3,j}$; $a_{3,j}$, $a_{2,j}$ (j = 1, 2, ….., m)

6) 3m lados $se_{k,j}$(k = 1, 2, 3, j = 1, 2, ……, m)

7) 3m lados $e_{k,j}$, $v_{l_{k,j}}$ (k = 1, 2, 3, j = 1, 2, ……, m)

Como *G* tiene triángulos, entonces $\chi(G) \geq 3$. A si que $\chi(G) \leq 3$ si y sólo si $\chi(G) = 3$. Entonces probaremos que *B* es satisfactible si y solo si $\chi(G) = 3$. Vamos a necesitar distinguir entre variables y los asignamientos de valores a las variables. Un asignamiento de valores es un vector de bits en $\{0, 1\}^n$. Lo denotaremos por *b.* Entonces x(*b*) es el valor que asume la variable x en ese asignamiento. Similarmente denotaremos por l(*b*) el valor que asume el literal l. Y D_j(*b*) el valor que asume toda disjunción y *B*(*b*) el valor que asume toda la expresión booleana.

Veamos primero la implicación *B* satisfactible $\Rightarrow \chi(G) = 3$. Debemos dar un coloreo

propio con 3 colores de *G*, a si que debemos colorear todos los vértices de *G*, uno por uno. Tenemos que usar que hay un asignamiento de valores a las variables de *B* que la vuelve verdadera. Es decir, existe un $b \in \{0,1\}^n$ tal que *B*(*b*) = 1. Coloreamos los vértices v_l por medio de c(v_l) = l(*b*). Al ir coloreando los vértices, iremos chequeando que el coloreo sea propio, lo cual para cada lado donde los dos vértices ya hayan sido coloreados, los extremos tengan colores distintos.
Ya tenemos coloreados los vértices v_l y $v_x v_{\bar{x}}$ forman lado, a si que tenemos que chequearlos. Pero c($v_{\bar{x}}$) = $\bar{x}$(*b*) = 1 - x(b) ≠ x(*b*) = c(v_x) así que $v_x v_{\bar{x}}$ no crea problemas. Coloreamos ahora c(*s*) = 1 y c(*t*) = 2. Como c(*s*) ≠ c(*t*) entonces *st* no crea problemas. Como c(*t*) = 2 y c(v_l) ∈ {0, 1}, entonces tv_l no crea problemas. Ahora debemos colorear los *e* y *a*. Acá tenemos que usar que *B*(*b*) = 1. Como *B*(*b*) = 1 y *B* = $D_1 \wedge \ldots \wedge D_m$. entonces D_j(*b*) = 1 ∀j. Como $D_j = l_{1,j} \vee l_{2,j} \vee l_{3,j}$, entonces por lo de arriba para todo j existe al menos un k_j tal que $l_{k_j,j}$(*b*) = 1. (‡)

Si hay más de un tal k_j elegimos uno solo, por ejemplo el primero. Coloreamos, para cada j:

→ c($a_{k_j,j}$) = 2

→ Para los $a_{r,j}$ con r ≠ k_j, qué son los dos que quedan, coloreamos un de ellos con 1, y el otro con 0

Esto significa que el triángulo formado por los $a_{k,j}$ tiene los tres vértices de distinto color, por los tanto no crea problemas.

Quedan los *e*, que están unidos con *s*, algunos v_l y los *a*. Coloreamos, para cada j, c($e_{k_j,j}$) = 0 y c($e_{r,j}$) = 2 para los r ≠ k_j. Como c($e_{k_j,j}$) = 0 y c($a_{k_j,j}$) = 2 entonces $e_{k_j,j}$, $a_{k_j,j}$ no crea problemas. Como c($e_{r,j}$) = 2 y c($a_{r,j}$) ∈ {0,1} entonces para r ≠ k_j tenemos que $e_{r,j}$, $a_{r,j}$ no crea problemas. Como c($e_{i,j}$) ∈ {2,0} y c(*s*) = 1, entonces para r ≠ k_j tenemos c($e_{r,j}$) = 2 y c($v_{l_r,j}$) ∈ {0,1} así que $e_{r,j}$, $v_{l_r,j}$ no crea problemas.

Para el caso k_j tenemos c($v_{l_{k_j,j}}$) = $l_{k_j,j}$(*b*) = 1 (por ‡). Por lo tanto, como c($e_{k_j,j}$) = 0 y c($v_{l_{k_j,j}}$) = 1, entonces $e_{k_j,j}$, $v_{l_{k_j,j}}$ no crea problemas. Hemos terminado de colorear todos los vértices con 3 colores y chequeado que el coloreo es propio. Por lo tanto, hemos completado la implicación *B* satisfactible ⇒χ(G) = 3.
Veamos la vuelta. Supongamos que existe un coloreo propio *c* de *G* con tres colores.

Debemos definir un $b \in \{0, 1\}^n$ tal que $B(b) = 1$. Definimos b usando el coloreo c de la siguiente forma: $b_i = 1$ si $c(v_{x_i}) = c(s)$ y $b_i = 0$ si $c(v_{x_i}) \neq c(s)$. Sea ahora $j \in \{1, 2, ..., m\}$. Como los $a_{k,j}$ forman un triángulo, y c colorea G con tres colores, entonces los tres colores deben estar representados en ese triángulo. En particular, existe un q tal que $c(a_{q,j}) = c(t)$.

Analicemos el color posible de $e_{q,j}$

1) $e_{q,j}, a_{q,j} \in E \Rightarrow c(e_{q,j}) \neq c(a_{q,j}) = c(t)$
2) $e_{q,j}, s \in E \Rightarrow c(e_{q,j}) \neq c(s)$

Por lo tanto, $e_{q,j}$ debe tener el "tercer" color, es decir, el color que no es color de s ni el color de t.

$v_{l_{q,j}}, e_{q,j} \in E \Rightarrow c(v_{l_{q,j}}) \neq c(e_{q,j})$

$v_{l_{q,j}}, t \in E \Rightarrow c(v_{l_{q,j}}) \neq c(t)$

Como $e_{q,j}$ tiene el tercer color, vemos que $v_{l_{q,j}}$ no puede tener ni el tercer color ni el color de t, por lo tanto, sólo le queda tener el color de s: $c(v_{l_{q,j}}) = c(s)$.

$l_{q,j}$ es un literal, a si que es una variable o una negación de una variable, analicemos los dos casos:

1) Si $l_{q,j}$ es una variable: $\exists i$ con $l_{q,j} = x_i$
   a) $c(v_{l_{q,j}}) = c(s)$ implica entonces $c(v_{x_i}) = c(s)$. Esto es por la definición de b implica que $b_i = 1$. Entonces $l_{q,j}(b) = x_i(b) = b_i = 1$
2) Si $l_{q,j}$ es una negación de una variable: $\exists i$ con $l_{q,j} = \overline{x_i}$
   a) $c(v_{l_{q,j}}) = c(s)$ implica entonces $c(v_{\overline{x_i}}) = c(s)$. Como $v_{x_i}, v_{\overline{x_i}} \in E$ entonces $c(v_{x_i}) \neq c(v_{\overline{x_i}})$. Así, $c(v_{x_i}) \neq c(s)$ lo cual implica que $b_i = 0$. Entonces $l_{q,j}(b) = \overline{x_i}(b) = 1 - x_i(b) = 1 - b_i = 1 - 0 = 1$.

En cualquiera de los dos casos hemos probado que $l_{q,j}(b) = 1$.

Como $D_j = l_{1,j} \vee l_{2,j} \vee l_{3,j}$, entonces $l_{q,j}(b) = 1$ implica que $D_j(b) = 1$. El j era cualquiera en {1, 2,...., m}, es decir, hemos probado que $D_j(b) = 1$ para todo j. Como $B = D_1 \wedge ... \wedge D_m$ lo anterior implica que $B(b) = 1$. Así B es satisfactible.

---------- **PARTE A** ----------

(2) Complejidad del algoritmo de Edmonds-Karp

La complejidad del algoritmo de Edmonds-Karp es $O(nm^2)$.

Prueba:

Supondremos que si en el network está el lado xy(→), entonces no está el lado yx (→). Como la búsqueda y construcción de cada camino aumentante se hace con BFS, cada incremento del flujo tiene complejidad *O*(m). Si f_0, f_1, f_2,....,etc son flujos parciales producidos al correr Edmonds-Karp, entonces queremos ver que hay una cantidad finita de ellos, y dar una cota para ese número. Para ello se define ***lados críticos***:

||| Diremos que un lado xy(→) **se vuelve crítico** durante la construcción de uno de los flujos intermedio (digamos, f_{k+1}) si para la construcción de f_{k+1} para una de las dos cosas siguientes:

1) Se usa el lado en forma forward, saturandolo (es decir $f_k(xy\rightarrow) < c(xy\rightarrow)$, si no no se podría usar, pero luego $f_{k+1}(xy\rightarrow) = c(xy\rightarrow)$).

2) O se usa el lado en forma backward, vaciandolo (es decir $f_k(xy\rightarrow) > 0$ pero $f_{k+1}(xy\rightarrow) = 0$). |||

Supongamos que podemos probar que el número de veces que un lado puede volverse crítico está acotado por un número *B*. Entonces como cada camino aumentante que se usa en Edmonds-Karp tiene al menos un lado crítico, el total de flujos intermedios estaría acotado por *mB*, pues hay *m* lados y cada uno se puede volver crítico a los sumo *B* veces. Como vimos que cada construcción de un flujo tiene complejidad *O*(*m*), concluiriamos que la complejidad de Edmonds-Karp es *O*(*m*) * *O*(*mB*) = $O(Bm^2)$. Para terminar la prueba necesitamos probar que *B* = *O*(*n*).

Definición: dados vértices *x*, *z* y flujo *f* definimos a **la distancia entre *x* y *z* relativa a *f* como la longitud del menor *f*-camino aumentante entre *x* y *z***, si es que existe tal camino, o infinito si no existe o 0 si *x* = *z*. **La denotaremos como** $d_f(x,z)$

Notación: dado un vértice *x* denotamos $d_k(x) = d_{f_k}(s,x)$ y $b_k(x) = d_{f_k}(x,t)$

Definición: dado un flujo *f* y un vértice *x*, diremos que un vértice *z* es un vecino *f*FF de *x* si pasa alguna de las siguientes condiciones:

- xz(→) ∈ E y *f*(xz→) < c(xz→)
- zx(→) ∈ E y *f*(zx→) > 0

Observación: **si *z* es un $f_k FF$vecino de *x*, entonces** $d_k(z) \leq d_k(x) + 1$

Prueba de la observación:

Si no existe f_k-camino aumentante entre *s* y *x*, entonces $d_k(x) = \infty$ y el lema es obvio. Supongamos entonces que existen f_k-caminos aumentantes entre *s* y *x*, y tomemos de entre todos ellos alguno con longitud mínima, y llamemosle *P*. Por lo tanto, la longitud de *P* es $d_k(x)$. Como *z* es$f_k FF$vecino de *x*, si tomamos *Q* el camino que consiste en agregar *z* al final de *P*, tenemos que *Q* es un f_k-camino aumentante

entre *s* y *z*. La longitud de *Q* es igual a la longitud de *P* más uno.

Q es UN f_k-camino aumentante entre *s* y *z*, por lo tanto la MENOR longitud posible entre TODOS los f_k-caminos aumentantes entre *s* y *z* será menor o igual que la longitud de *Q*. Dado que la menor longitud posible entre todos los f_k-caminos aumentantes entre *s* y *z* es $d_k(z)$ y la longitud de *Q* es $d_k(x) + 1$, hemos probado que $d_k(z) \leq d_k(x) + 1$.

Las distancias no disminuyen: las distancia definidas anteriormente no disminuyen a medida que *k* crece. Es decir, $d_k(x) \leq d_{k+1}(x)$y $b_k(x) \leq b_{k+1}(x)\ \forall x$

Continuando con la prueba de complejidad de Edmonds-Karp.

Ahora estamos en condiciones para acotar cuantas veces puede un lado volverse crítico. Supongamos que xy(→) se vuelve crítico en el paso *k* y luego en el paso *r* con *r* > *k*. Tenemos que analizar dos casos: se vuelve crítico en el paso *k* porque se saturó, o se vuelve crítico en el paso *k* porque se vació. Si se saturó entonces para que se vuelva crítico en el paso *r*, deben pasar una de dos cosas:

- Se vuelve crítico en el paso *r* porque se vacía
- Se vuelve crítico en el paso *r* porque vuelve a saturarse

Para que ocurra el segundo caso, debe haberse vaciado ANTES aunque sea un poco, si no es imposible que vuelva a saturarse. Entonces, en cualquiera de los casos, deducimos que existe $l > k$ tal que el flujo en xy(→) disminuye al pasar de f_l a f_{l+1}.

Entonces:

1) En el paso *k*, se satura: esto implica que para construir f_{k+1}se usa un f_k-camino aumentante de la forma *s...xy...t*. **Como estamos usando Edmonds-Karp**, ese camino es de longitud mínima, por lo tanto $d_k(y) = d_k(x) + 1$ (i)

2) En el paso $l$el flujo disminuye, ya sea vaciandose completamente o un poco: esto implica que para construir f_{l+1}se usa un $f_l$camino aumentante de la forma *s...yx...t*. **Como estamos usando Edmonds-Karp**, ese camino es de longitud mínima, por lo tanto, $d_l(x) = d_l(y) + 1$(ii)

Entonces:

$$
\begin{aligned}
d_l(t) &= d_l(x) + b_l(x) \\
&= d_l(y) + 1 + b_l(x) \quad \text{por (ii)} \\
&\geq d_k(y) + 1 + b_k(x) \quad \text{porque las distancias no disminuyen} \\
&= d_k(x) + 1 + 1 + b_k(x) \quad \text{por (i)}
\end{aligned}
$$

= $d_k(t)$ + 2

Este análisis era si el lado xy(→) se volvía crítico en el paso *k* porque se saturaba. Si en vez de eso se vuelve crítico en ese paso porque se vacía, el análisis es similar. Como se vacía, existe un camino (de longitud mínima) de la forma *s...yx...t* que se usa para pasar de f_k a f_{k+1}. Por lo tanto $d_k(x) = d_k(y) + 1$(iii)

Para poder volver a ser crítico, debe llenarse de vuelta, ya sea completamente o parcialmente. Así que debe haber un $l > k$ para el cual se manda flujo a través de él. Entonces, existe un camino (de longitud mínima) de la forma *s...xy...t* que se usa para pasar de f_l a f_{l+1}. Por lo tanto $d_l(y) = d_l(x) + 1$. (iv)

Observemos que (iii) y (iv) son iguales a (i) y (ii), sólo que con *x* e *y* intercambiados. Por lo tanto podemos deducir $d_l(t) \geq d_k(t) + 2$. Simplemente intercambiando *x* e *y* en la prueba. Entonces, como $d_r(t) \geq d_l(t) \geq d_k(t) + 2$, concluimos que para que un lado vuelva a volverse crítico, la distancia entre *s* y *t* debe aumentar en al menos 2. Como la distancia entre *s* y *t* puede ir desde un mínimo de 1 a un máximo de n-1, concluimos que un lado puede volverse crítico un máximo de *O*(*n*) veces. Así que el "*B*" del principio de la prueba en *n*.

(3) Dados vértices x, z y flujo f definimos a la distancia entre x y z relativa a f como la longitud del menor f-camino aumentante entre x y z, si es que existe tal camino, o infinito si no existe o 0 si x = z, denotandola por df (x, z), y definimos dk(x) = dfk (s, x), donde fk es el k-ésimo flujo en una corrida de Edmonds-Karp, entonces dk(x) ≤ dk+1(x).

Si no existe f_k-camino aumentante entre *s* y *x*, entonces $d_k(x) = \infty$ y el lema es obvio. Supongamos entonces que existen f_k-caminos aumentantes entre *s* y *x*, y tomemos de entre todos ellos alguno con longitud mínima, y llamemosle *P*. Por lo tanto, la longitud de *P* es $d_k(x)$. Como *z* es$f_k FF$vecino de *x*, si tomamos *Q* el camino que consiste en agregar *z* al final de *P*, tenemos que *Q* es un f_k-camino aumentante entre *s* y *z*. La longitud de *Q* es igual a la longitud de *P* más uno.

Q es UN f_k-camino aumentante entre *s* y *z*, por lo tanto la MENOR longitud posible entre TODOS los f_k-caminos aumentantes entre *s* y *z* será menor o igual que la longitud de *Q*. Dado que la menor longitud posible entre todos los f_k-caminos aumentantes entre *s* y *z* es $d_k(z)$ y la longitud de *Q* es $d_k(x) + 1$, hemos probado que $d_k(z) \leq d_k(x) + 1$.

(4) Probar que la distancia en networks auxiliares sucesivos aumenta

Prueba:

Supondremos que en el network original si está el lado xy(→) entonces no está el lado yx(→). Sea *NA* un network auxiliar y *NA'* el network auxiliar siguiente. Sea *d* la distancia (de Ford-Fulkerson) de un vértice a *s* en el network original cuando construimos *NA* y *d'* similar pero cuando construimos *NA'*. Sabemo que $d(t) \leq d'(t)$

por la prueba de Edmonds-Karp. Queremos probar que vale el < ahí.

Si *NA'* no tiene a *t* entonces $d'(t) = \infty > d(t)$y ya está. Asumamos entonces t ∈ *NA'*. Entonces existe un camino dirigido $x_0x_1..x_{r-1}x_r$ entre *s* y *t* **en *NA'***. Como para pasar de *NA* a *NA'* debemos saturar todos los caminos entre *s* y *t* de *NA*, vemos que ese puede ser un camino **en *NA***. Pues si lo fuera habría al menos un lado saturado de ese camino, y no podría ser un camino en *NA'*.

Como $x_0x_1..x_{r-1}x_r$ **no es** un camino en *NA*, entonces pasa una de las dos cosas:

1) Algún vértice x_i no está en *NA*

2) Están todos los x_i en *NA* pero falta algún lado $x_ix_{i+1}(\rightarrow)$.

Veamos estos dos casos. Supongamos primero que algún vértice x_i no está en *NA*.

Caso 1

Como *t* está en *NA*, entonces $x_i \neq t$. Todos los vértices $\neq t$ que estén a distancia mayor o igual que *t* no se incluyen, pero todos los que tengan distancia menor a *t* están pues construimos *NA* con BFS. Entonces la única forma en que x_i no este en *NA* es que $d(t) \leq d(x_i)$. (1)

Como probamos en Edmonds-Karp que $d \leq d'$, tenemos: $d(x_i) \leq d'(x_i)$. (2)

Ahora, como $x_0x_1..x_{r-1}x_r$ es un camino en *NA* y *NA'* es un network por niveles, concluimos que $d'(x_i) = i$ para todo i. (3)

Además vimos que $x_i \neq t$, a si que $i < r$. (4)

Entonces: $d(t) \leq_{(1)} d(x_i) \leq_{(2)} d'(x_i) =_{(3)} i <_{(4)} r = d'(t)$. Y probamos este caso

Caso 2

Asumimos ahora que están todos los x_i en *NA* pero falta algún lado $x_ix_{i+1}(\rightarrow)$, y tomamos el primer *i* para el cual pasa eso. Por Edmonds-Karp, sabemos que $d(x_{i+1}) \leq d'(x_{i+1})$. Así que tenemos dos subcasos: que ese ≤ sea <, o que sea =.

Subcaso A caso 2:

$d(x_{i+1}) < d'(x_{i+1})$(5)

Usando *b*, *b'* para las distancias de un vértice a *t* en los networks respectivos y recordando que $b \leq b'$, y que $d(t) = d(x) + b(x)$ para todo x y similar para *d'*, *b'*, tenemos:

$$\begin{aligned} d(t) &= d(x_{i+1}) + b(x_{i+1}) \\ &\leq d(x_{i+1}) + b'(x_{i+1}) \quad \text{(lema de Edmonds-Karp)} \\ &<_{(5)} d'(x_{i+1}) + b'(x_{i+1}) = d'(t) \end{aligned}$$

Y hemos probado $d(t) < d'(t)$ en este subcaso

Subcaso B de caso 2

Supongamos ahora que $d(x_{i+1}) = d'(x_{i+1}) = i + 1$. Como *i* es el primer índice para el cual $x_ix_{i+1}(\rightarrow)$ no está en *NA*: entonces la porción del camino $x_0x_1..x_i$ **si está**

en *NA*. Esto implica (al se *NA* un network por niveles) que $d(x_i) = i$.

Entonces, en *NA*, x_i está en el nivel *i*, y x_{i+1}en el nivel *i+1*. En particular, concluimos no solo no está en *NA* el lado $x_i x_{i+1}(\rightarrow)$ sino que tampoco está el lado x_i $x_{i+1}(\rightarrow)$ (pues los niveles no son legales para que ese lado esté).

Como $d(x_i) = i$, $d(x_{i+1}) = i + 1$, entonces x_i, x_{i+1} están a distancia "legal" para que pueda existir el lado $x_i x_{i+1}(\rightarrow)$. Pero ese lado no está en *NA*. Por lo que concluimos que:

1) $x_i x_{i+1}(\rightarrow)$ es el lado del network original pero está saturado

2) $x_{i+1} x_i(\rightarrow)$ es el lado del network original pero está vacío

Pero $x_i x_{i+1}(\rightarrow)$ si es un lado en *NA'*, a si que la saturación es:

1) $x_i x_{i+1}(\rightarrow)$ es el lado del network original, estaba saturado al construir *NA* pero des-saturado al construir *NA'*

2) $x_{i+1} x_i(\rightarrow)$ es el lado del network original, estaba vacío al construir *NA* pero no vacío al construir *NA'*.

Ya sea para des-saturar $x_i x_{i+1}(\rightarrow)$ en el primer caso o para mandar flujo por $x_{i+1} x_i(\rightarrow)$ en el segundo, vemos que si o si debemos haber usado un lado $x_{i+1} x_i(\rightarrow)$ **en *NA***. Pero habíamos visto que ese lado no está en *NA*, absurdo.

(5) Complejidad de Dinic

<u>Prueba complejidad Dinic-Even</u>

Si denotamos AVANZAR por A, RETROCEDER por R e INCREMENTAR+inicializar por I, entonces vemos que Dinic-Even es una sucesión de As, Rs, Is. P. Las preguntas que debemos responder son

1) ¿Cuántas palabras hay?
2) ¿Cuál es la "complejidad" de cada palabra?

<u>¿Cuántas palabras hay?</u>

Cada palabra termina en un x = (R o I). R borra el lado por el cual retrocede. I manda flujo por un camino en el cual se satura al menos un lado, y borra todos los lados saturados. Concluimos que X, sea R o I, borra al menos un lado. Por lo tanto la cantidad de palabras es a lo sumo m.

<u>¿Cuál es la complejidad de cada palabra?</u>

Vimos que R y A son $O(1)$. I es INCREMENTAR, que es $O(n)$ más un inicializar, que es $O(1)$, así que I es $O(n)$. Así que si una palabra A…..AX tiene *r* As, la complejidad de la palabra será $O(r+1) = O(r)$ si X es R y $O(r+n)$ si X es I. Pero A mueve el extremo del camino donde estamos parados un nivel para adelante. Como puede haber a lo sumo *n* niveles, entonces $r \leq n$. Por lo tanto la complejidad de A….AX es $O(n)$ si X=R y $O(n+n) = O(n)$ si X=I. Es decir, $O(n)$ en cualquiera de los dos casos.

Entonces hemos probado que hay a lo sumo *m* palabras, y la complejidad de cada palabra es $O(n)$. Por lo tanto la complejidad total del paso de encontrar un flujo

bloqueante es (números de palabras) * (complejidad de c/palabra) = $O(mn)$. Y por lo tanto, como vimos al principio de la prueba, la complejidad total es

$$O(n) * O(mn) = O(mn^2)$$

Prueba complejidad de Dinitz original

Es similar a Dinic-Even, sólo que no tiene Rs, pero tiene PODARs, que denotaremos por *P*. Así que la estructura luce como A...AIPiA...AIPi… donde "i" es inicializar. La complejidad de cada AA...AI es $O(n+n) = O(n)$ pues acá hay exactamente *n* As e I es $O(n)$. Al igual que el análisis anterior, I borra al menos un lado, así que hay a lo sumo *m* de estos A….AI. La complejidad total de todos los A...AI es entonces $O(mn)$. Hay un i por cada I, así que hay *m* is, y cada i es $O(1)$, así que la complejidad total de los is es $O(m)$. Quedan los Ps.

La complejidad de cada *P* es variable, pues *P* borra vértices, que en realidad se traduce en borrar todos los lados que llegaban al vértice y podría ser que haya que borrar uno o incluso ningún lado, o tener que borrar casi todos, así que en el peor caso es $O(m)$. La clave está en no contar la complejidad de cada *P* y cuántos hay, sino la complejidad de TODOS los Ps en CONJUNTO. Primero se observa que *P* está dividido en dos partes:

1) Va chequeando vértices mirando si tienen lados de salida o no
2) Si tiene lados de salida, no hace nada. Si no, borra todos los lados de entrada

Llamaremos a la primer parte PV, es decir, PV recorre todos los vértices, y llamaremos $B(x)$ a borrar todos los lados de entrada de un vértice *x*.

Análisis PVs

La complejidad de cada PV es $O(n)$, pues chequea todos los vértices. Hay un PV luego de cada I, así que hay a lo sumo *m* en total. La complejidad total de todos ellos es $O(nm)$. Para los $B(x)$ es donde debemos hacer un análisis distinto ya que no sabemos cuántos $B(x)$s se hacen luego de un PV.

Análisis de $B(x)$

Observemos que la complejidad de un $B(x)$ es $O(d(x))$. También, si se "activa" la llamada a $B(x)$ no se vuelve a hacer B(x) para ese vértice *x* nunca más, porque ya le borramos todos los lados. Entonces, independientemente de cuántos $B(x)$s hay luego de UN PV, lo que sabemos es que al final de todo, habrá a lo sumo un $B(x)$ por cada vértice. Como la complejidad de $B(x)$ es $O(d(x))$, entonces la complejidad del conjunto $B(x)$s es la suma de los grados de todos los vértices, que es $O(m)$. (lema del apretón de manos)

Resumiendo, la complejidad total de los AA...AI es $O(nm)$, la de los is es $O(m)$, la de los PVs es $O(nm)$ y la de todos los $B(x)$s es $O(m)$. Entonces la complejidad total del paso bloqueante es

$$O(nm) + O(m) + O(nm) + O(m)$$

y la del algoritmo completo, $O(mn^2)$, como vimos. Fin

(6) Complejidad algoritmo de Wave

La complejidad de Wave es $O(n^3)$

La complejidad de Wave es n veces la complejidad de encontrar un flujo bloqueante en un network auxiliar. Así que basta con demostrar que la complejidad

de encontrar un flujo bloqueante en un network auxiliar es $O(n^2)$.

Para calcular esa complejidad, observemos que para encontrar un flujo bloqueante, Wave hace una serie de ciclos de olas hacia adelante-olas hacia atrás, y en cada ola hacia adelante hacemos una serie de *BalanceoHaciaAdelante*(x) y en cada ola hacia atrás una serie de *BalanceoHaciaAtras*(x).

En cada uno de esos, revisamos una serie de *ys*, ya sea en $\Gamma^+(x)$o en *M*(*x*), y "procesamos" el lado xy(→) ó yx(→), de acuerdo sea el caso.

Cada uno de esos "procesamientos" de lado es *O*(1) (pues hay que calcular cuánto mandar/devolver, restarlo de *D*(*x*), sumarlo a *D*(*y*), cambiar cuánto vale g en el lado correspondiente, y hacer algún IF).

Por lo tanto la complejidad de encontrar un flujo bloqueante con Wave es simplemente la cantidad total de estos "procesamientos" de lados.

Para poder calcular cuantos de estos "procesamientos" hacemos, debemos dividir estos "procesamientos" en categorías:

Cuando estamos procesando un lado *xy*(→) para mandar flujo de *x* a *y*, pueden pasar dos cosas:

1) Luego de procesarlo, el lado *xy*(→) queda saturado
2) No queda saturado

Sea S la cantidad total sobre todas las olas hacia adelante, de procesamientos de la categoría 1) arriba, y P la cantidad total, sobre todas las olas hacia adelante, de procesamientos de la categoría 2) arriba.

Similarmente, cuando estamos procesando un lado yx(→) para que *x* le devuelva flujo a *y*, pueden pasar dos cosas:

I) Luego de procesarlo, el lado yx(→) queda vacío
II) No queda vacío

Sea *V* la cantidad total sobre todas las olas hacia atrás, de procesamientos de la categoría I) arriba, y *Q* la cantidad total de procesamientos sobre todas las olas hacia atrás, de lados de la categoría II) arriba.

Entonces la complejidad del paso bloqueante de Wave es *S* + *P* + *V* + *Q*.

Calculemos esos números.

En cada *BalanceoHaciaAdelante*(x) buscamos vecinos de x y les mandamos todo el flujo que podamos: $Min\{D(x), c(\rightarrow xy) - g(\rightarrow xy)\}$. Si ese mínimo es igual a $c(\rightarrow xy) - g(\rightarrow xy)$ el lado *xy*(→) queda saturado, lo retiramos de $\Gamma^+(x)$ y continuamos con otro. Entonces, de entre todos los vecinos de *x* hay A LO SUMO uno solo tal que ese mínimo es *D*(*x*). Es decir, al hacer *BalanceoHaciaAdelante*(x), todos los lados que procesamos, salvo a lo sumo uno, se saturan.

Concluimos entonces que en cada *BalanceoHaciaAdelante* hay a lo sumo UN lado procesado como parte de *P.* Entonces *P* está acotado superiormente por la cantidad total de *BalanceoHaciaAdelante*.

Ahora bien, en cada ola hacia adelante, hay a lo sumo *n* − *2* *BalanceoHaciaAdelante* así que sólo necesitamos calcular cuántas olas hacia adelante hay. Pero en cada ola hacia adelante, salvo la última, AL MENOS UN vértice es bloqueado. ¿Por qué?, pues porque si una ola hacia adelante no bloquea ningún vértice, esto significa que los balanceó a todos, y si los balancea a todos, el

algoritmo termina, así que debe ser la última ola.

Entonces, como los vértices NUNCA SE DESBLOQUEAN, puede haber a lo sumo *n* olas hacia adelante. (*n−2 + 1 = n−1* si nos queremos poner estrictos, pero es irrelevante). Por lo tanto, tenemos a lo sumo n−2 *BalanceoHaciaAdelante* por cada ola hacia adelante, y tenemos *O*(*n*) olas hacia adelante, significa que tenemos en total $O(n^2)$*BalanceoHaciaAdelante* y por lo tanto, la cantidad de procesamientos de *P* también es $O(n^2)$.

Similarmente, al hacer un *BalanceoHaciaAtras*, a lo sumo un lado no se vacia, asi que *Q* es la cantidad total de *BalanceoHaciaAtras* ,que son *n − 2* por cada ola hacia atrás, y como el número de olas hacia atrás es igual al número de olas hacia adelante, y vimos que estas están acotadas por n, tenemos que *Q* también es $O(n^2)$

Nos queda calcular *S* y *V* . Observemos que una vez que un lado *xy*(→) se satura, NUNCA MÁS se satura. Es decir, los lados se saturan una sola vez, ¿por qué?. Pues porque para poder saturarse otra vez, primero y debe DEVOLVERLE flujo a *x*. Pero la única forma en que y le puede devolver flujo a *x* es si *y* esta bloqueado, y si está bloqueado, *x* no puede mandarle mas flujo a *y*, y en particular *xy*(→) no puede volver a saturarse.

Por lo tanto, *S* está acotado por el número total de lados, *m*.

Similarmente, un lado *yx*(→) sólo puede vaciarse una sola vez, porque para poder vaciarse otra vez, primero y deber´ıa enviarle flujo a *x*. Pero si *yx*(→) se vació es porque *x* le devolvió flujo a *y,* y esto sólo puede pasar si *x* está bloqueado. Pero si *x* está bloqueado, y no puede mandarle flujo, y por lo tanto *yx*(→) no puede vaciarse otra vez porque nunca puede llenarse.

Así que *V* también está acotado por el número total de lados, *m*.

Por lo tanto la complejidad de hallar un flujo bloqueante en un network auxiliar con Wave es igual a $S + P + V + Q = O(m) + O(n^2) + O(m) + O(n^2) = O(n^2)$.

Como se dijo, esto implica que la complejidad total de Wave es $O(n^3)$.

---------- PARTE B ----------

(7) Probar que si f es un flujo, las siguientes afirmaciones son equivalentes:

1) f es maximal.

2) Existe un corte S tal que v(f) = cap(S). (y en este caso, S es minimal)

3) No existen f-caminos aumentantes

Para probar la equivalencia de 1,2,3 probaremos que 1 ⇒ 3 ⇒ 2 ⇒ 1

 # 1 ⇒3

Si existiese un f-camino aumentante, podríamos mandar un ε a través de él, y obtener un flujo f* tal que *v*(*f**) = *v*(*f*) + ε. Esto diría que *v*(*f**) > *v*(*f*) lo cual contradice

que *f* sea maximal

 # 2 ⇒1

Sea *g* un flujo cualquiera. tenemos que $v(g) \leq cap(S)$. Pero por hipótesis (2) tenemos que $cap(S) = v(f)$. Concluimos que $v(g) \leq v(f)$ y por lo tanto *f* es maximal. Si *T* es un corte, $cap(T) \geq v(f) = cap(S)$, es decir, *S* es minimal

 # 3 ⇒2

Necesitamos construir un *S* que sea corte con $cap(S) = v(f)$.

S = {*s*} ∪ {*x* ∈ *V*: exista un f-camino aumentante desde *s* a *x*}

Como estamos suponiendo que vale (3), entonces $t \notin S$. Como $s \in S$ por definición, y vimos que $t \notin S$, entonces *S* es un corte. Por lo tanto $v(f) = f(S, \overline{S}) - f(\overline{S}, S)$. Calculemos $f(S, \overline{S})$ y $f(\overline{S}, S)$.

$f(S, \overline{S}) = \sum_{x,y} f(xy \rightarrow)[x \in S][y \notin S][xy(\rightarrow) \in E]$. Consideremos un par *x*, *y* de los que aparecen en esa suma. Como x ∈ *S*, entonces existe un f-camino aumentante entre *s* y *x*, digamos $s = x_0, x_1, \ldots, x_r = x$. Pero como $y \notin S$, entonces no existe un f-camino aumentante entre *s* e *y*. En particular $s = x_0, x_1, \ldots, x_r = x, y$ NO ES un f-camino aumentante. Pero $xy(\rightarrow) \in E$, a si que PODRÍA serlo. Entonces la única razón por la cual no es un f-camino aumentante es que $f(xy \rightarrow) = c(xy \rightarrow)$. Esto es cierto para cualquier *x*, *y* que aparezcan en esa suma. Entonces:

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

$$= \sum_{x,y} c(xy \rightarrow)[x \in S][y \notin S][xy(\rightarrow) \in E]$$

$$= c(S, \overline{S}) = cap(S)$$

Veamos ahora $f(\overline{S}, S)$.

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

Consideremos un par *x*, *y* de los que aparecen en esa suma. Como $y \in S$, entonces existe un f-camino aumentante entre *s* e *y*, digamos $s = x_0, x_1, \ldots, x_r = y$. Pero como $x \notin S$, entonces no existe un f-camino aumentante entre *s* y *x*. En particular $s = x_0, x_1, \ldots, x_r = x, y$ NO ES un f-camino aumentante. Pero $xy(\rightarrow) \in E$, a si que PODRÍA serlo, usando *y*, *x* como lados backward. Entonces la única razón por la cual no es un f-camino aumentante es que $f(xy \rightarrow) = 0$. Esto es cierto para cualquier *x*, *y* que aparezcan en esa suma. Entonces:

$$f(\bar{S}, S) = \sum_{x,y} f(xy \rightarrow)[x \notin S][y \in S][xy(\rightarrow) \in E]$$

$$= \sum_{x,y} 0\,[x \notin S][y \in S][xy(\rightarrow) \in E]$$

$$= 0$$

Con esto y lo que probamos antes, concluimos:

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

$$= cap(S) - 0 = cap(S)$$

con lo cual hemos probado (2). Fin

(8) Enunciar el teorema de la cota de Hamming y probarlo.

Sea *C* un código de longitud *n*, $\delta = \delta(C)$ y $t = [\frac{\delta-1}{2}]$. Entonces:

$$\#\, C \leq \frac{2^n}{1+n+\ldots+ C_{n,t}}$$

Sea $A = \bigcup_{v \in C} D_t(v)$.

C corrige *t* errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.Por lo tanto la unión que forma a *A* es una unión disjunta, lo cual nos permite calcular la cardinalidad de *A* como: $\#A = \sum_{v \in C} \#D_t(v)$. Para calcular la cardinalidad de los D_ts, definamos los siguientes conjuntos:

$$S_r(v) = \{w \in \{0,1\}^n : d_H(v,w) = r\}$$

Recordemos que $D_t(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq t\}$, por lo tanto $D_t(v) = \bigcup_{r=0}^{t} S_r(v)$. Y como no podemos tener $d_H(v,w) = r$ y al mismo tiempo $d_H(v,w) = r'$ para $r' \neq r$, entonces esa unión es disjunta.

A si que $\#\, D_t(v) = \sum_{r=0}^{t} \#S_r(v)$. Necesitamos calcular $\#S_r(v)$. Un $w \in S_r(v)$es una palabra que difiere de *v* en exactamente *r* de los *n* lugares posibles, por lo tanto podemos asociar cada $w \in S_r(v)$ un subconjunto de cardinalidad *r* {1, 2,..., *n*} ; el conjunto de índices *i* tales que $w_i \neq v_i$. Como estamos trabajando con códigos binarios, dado un subconjunto de cardinalidad *r* de {1,2,...,*n*}, podemos crear un $w \in S_r(v)$. Pues dado un conjunto de *i*s, simplemente cambiamos los bits de *v*en esos *i*s.

Formalmente si $\Xi_r = \{Z \subseteq \{1, 2,..., n\} : \#Z = r\}$. Definimos $\psi : S_r(v) \rightarrow \Xi_r$ por medio de $\psi(w) = \{i : w_i \neq v_i\}$. Y definimos $\Phi : \Xi_r \rightarrow S_r(v)$ por medio de $\Phi(Z)$ = el único *w* tal que $w_i = v_i$ si $i \notin Z$ y $w_i = 1 \oplus v_i$ si $i \in Z$. Entonces ψ, Φ son inversas una

de la otra. Por lo tanto #$S_r(v)$= # Ξ_r. La cardinalidad de Ξ_r es el número combinatorio $C_{n,r}$.

Por lo tanto, $\#S_r(v) = C_{n,r}$.

Y $\#D_t(v) = \sum_{r=0}^{t} C_{n,r}$.

Y #A = $\sum_{v \in C} (\sum_{r=0}^{t} C_{n,r})$

La suma interior no depende de *v*, a si que queda:

#A = $(\sum_{r=0}^{t} C_{n,r}) \times \#C$. Por lo tanto:

$$\#C = \frac{\#A}{\sum_{r=0}^{t} C_{n,r}} \leq \frac{2^n}{\sum_{r=0}^{t} C_{n,r}}$$

Solo resta observar que como *A* es un subconjunto de $\{0,1\}^n$, su cardinalidad es menor o igual que 2^n, lo cual finaliza la prueba.

(9) Probar que si H es matriz de chequeo de C, entonces:
δ(C) = Min{ j : ∃ un conjunto de j columnas LD de H}

Prueba:

Denotando por $H^{(j)}$la columna j-ésima de *H*, sea $\{H^{(j_1)}, H^{(j_2)}, ..., H^{(j_s)}\}$ un conjunto LD de columnas de *H*. Esto significa que existes c_1,...., c_s **no todos nulos** tales que:

$c_2H^{(j_1)}+c_2H^{(j_2)}+...+c_sH^{(j_s)}$ = 0.

Sea *w* = $c_1e_{j_1} + c_2e_{j_2} + ... + c_se_{j_s}$ donde e_j es el vector con todos 0 salvo 1 en la coordenada *j*. Entonces:

$$Hw^t = H(c_1e_{j_1} + c_2e_{j_2} + ... + c_se_{j_s})^t$$
$$= c_1He^t_{j_1} + c_2He^t_{j_2} + ... + c_sHe^t_{j_s}$$

$$= c_1 H^{j_1} + c_2 H^{j_2} + \ldots + c_s H^{j_s}$$
$$= 0$$

Por lo tanto, $w \in C$, pues $C = Nu(H)$. Pero su peso es $\leq s$, pues es suma de a lo sumo s e_j. Y $w \neq 0$, pues no todos los c_j son 0.

Así: $\delta(C) = Min\{|v|: v \in C, v \neq 0\} \leq |w| \leq s$

Esto vale para cualquier conjunto de columnas LD de H. Concluimos que si m = Min{ j : ∃ un conjunto de j columnas LD de H}, entonces $\delta(C) \leq m$.

Ahora veamos la otra desigualdad:

Sea $v \in C$tal que $\delta(C)$= |v|. Esto último implica que existen i_1,...., $i_{\delta(C)}$ con

v = $e_{i_1} + e_{i_2} + \ldots + e_{i_{\delta(C)}}$. Como $v \in C$, entonces Hv^t = 0.

Con el mismo cálculo de antes, concluimos que

$H^{(i_1)}+H^{(i_2)}+\ldots+H^{(i_{\delta(C)})} = Hv^t$ = 0. Pero $H^{(i_1)}+H^{(i_2)}+\ldots+H^{(i_{\delta(C)})}$ = 0 dice que { $H^{(i_1)}+H^{(i_2)}+\ldots+H^{(\delta(C))}$} es un conjunto LD de columnas de H. Por lo tanto m = Min{ j : ∃ un conjunto de j columnas LD de H} $\leq \delta(C)$. Fin

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

1) C está formado por los múltiplos de g(x) de grado menor que n: C = {p(x) : gr(p) < n&g(x)|p(x)}

2) C = {v(x) ⊙ g(x) : v es un polinomio cualquiera}

3) gr(g(x)) = n − k

4) g(x) divide a 1 + x^n

<u>Prueba:</u>

Sea $C_1\{p(x): gr(p) < n\&g(x)|p(x)\}$y

$C_2 = \{v(x) \odot g(x): v\ es\ un\ polinomio\ cualquiera\}$

$C_2 \subseteq C$, pues $g(x) \in C$. Sea $p(x) \in C$. Dividamos $p(x)$ por $g(x)$, obteniendo polinomios $q(x)$ y $r(x)$, con $gr(r) < gr(g)$ tal que $p(x) = q(x)g(x) + r(x)$. Por lo tanto $r(x) = p(x) + q(x)g(x)$. Cómo $gr(r) < gr(g)$, entonces $r(x) = r(x)$ mod $(1 + x^n)$. Como $p(x) \in C$, entonces $gr(p) < n$, y $p(x) = p(x)$ mod $(1 + x^n)$. Entonces:

$r(x) = r(x)$ mod $(1 + x^n)$
$= (p(x) + q(x)g(x))$ mod $(1 + x^n)$
$= p(x)$ mod $(1 + x^n) + (q(x)g(x))$ mod $(1 + x^n)$
$= p(x) + q \odot g$

Como $p(x) \in C$ y $q \odot g \in C$ y C es lineal, concluimos que $r \in C$. Pero $gr(r) < gr(g)$ que es el polinomio **no nulo** de **menor** grado de C. Concluimos $r = 0$. Por lo tanto $p(x) = q(x)g(x) + r(x) = q(x)g(x) \in C_1$

Entonces concluimos que $C \subseteq C_1$ y habíamos visto que $C_2 \subseteq C$, sólo nos resta ver que $C_1 \subseteq C_2$. Pero esto es obvio, pues si $gr(p) < n$ y $p(x) = q(x)g(x)$, entonces:

$p(x) = p(x) \bmod (1 + x^n)$ (pues $gr(p) < n$)

Y por lo tanto $p(x) = q(x)g(x) \bmod (1 + x^n) = q \odot g \in C_2$. Con esto se probó las partes 1) y 2) del teorema. Vamos a la 3)

Sea t el grado de $g(x)$. Por 1), $p(x) \in C$ sii es de la forma $q(x)g(x)$ para algún polinomio $q(x)$. Pero como el grado de los elementos de C es menor que n, entonces el grado de $q(x)g(x)$ debe ser menor que n. Por lo tanto el grado de $q(x)$ debe ser menor que $n - t$. Así, para cada polinomio de grado menor que $n - t$ corresponde un polinomio de C, y viceversa. Por lo tanto la cardinalidad de C es igual a la cardinalidad del conjunto de polinomios de grado menor que $n - t$. Los polinomios de grado menor que $n - t$ tienen $n - t$ coeficientes (los de los términos de grado 0, 1, …., $n - t - 1$). Cada uno de esos coeficientes puede ser 1 o 0, así que cada uno tiene dos posibilidades. Como son $n - t$, el total de polinomios posibles es 2^{n-t}. Entonces hemos probado que la cardinalidad de C es 2^{n-t}. Pero como C es lineal, sabemos que su cardinalidad es 2^k. A si que $2^k = 2^{n-t}$, por lo tanto $k = n - t$, y el grado de $g(x)$ es $t = n - k$. Fin de la parte 3).

La parte 4) se puede probar de varias formas. Veamos una:

Dividimos $1 + x^n$ por $g(x)$, obteniendo $q(x)$, $r(x)$ con $gr(r) < gr(g)$ tal que $1 + x^n = q(x)g(x) + r(x)$. Por lo tanto $r(x) = 1 + x^n + q(x)g(x)$. Cómo $gr(r) < gr(g) < n$, $r(x) = r(x) \bmod (1 + x^n)$.

Así: $r(x) = (1 + x^n + q(x)g(x)) \bmod (1 + x^n) = q \odot g \in C$. Como $r \in C$ y $gr(r) < gr(g)$, entonces $r = 0$ y $g(x) / (1 + x^n)$ (divide).

(11) Enunciar y probar el Teorema de Hall.

Si G es un grafo bipartito con partes X e Y y $|S| \leq |\Gamma(S)|$ para todo $S \subseteq X$, entonces existe un matching completo sobre X.

Prueba:

Supongamos que el teorema no es cierto, es decir que $|S| \leq |\Gamma(S)|$ para todo $S \subseteq X$ pero que no existe un matching completo sobre X. Probaremos que el hecho de que no exista un matching completo sobre X implica que existe $S \subseteq X$ con $|S| > |\Gamma(S)|$ lo cual contradice la hipótesis.

Como estamos suponiendo que no existe matching completo sobre *X*, entonces corriendo el algoritmo para hallar matching maximal, usando Edmonds-Karp, obtenemos un matching maximal que NO cubre todos los vértices de *X*. Sea S_0 aquellos vértices de *X* que, al terminar el algoritmo, han quedado sin cubrir ($S_0 \neq \varnothing$ porque estamos suponiendo que no hemos cubierto a todos los vértices de *X*). Es decir, $x \in S_0$ si y solo si no existe $y \in Y$ tal que *xy* sea lado del matching. Esto significa que $f(xy\rightarrow) = 0$ para todo *y*, donde *f* es el flujo maximal encontrado. Por lo tanto $out_f(x) = 0$ para todo $x \in S_0$. Como *f* es flujo, eso significa que $in_f(x) = 0$ para todo $x \in S_0$.

Viceversa, supongamos que *x* es un vértices tal que $in_f(x) = 0$. Entonces $out_f(x) = 0$ y por lo tanto $f(xy\rightarrow) = 0$ para todo *y* lo cual significa que *x* no forma parte del matching. Concluimos que $S_0 = \{x \in X : in_f(x) = out_f(x) = 0\}$. En el último paso del algoritmo, Edmonds-Karp encuentra un corte minimal.

Digamos que el corte es *C*. Este corte incluye a *s*, y los otros elementos de *C* son vértices de *G*, que pueden estar en *X* o en *Y*. Así que existen $S \subseteq X$, $T \subseteq Y$ tal que $C = \{s\} \cup S \cup T$. Recordemos que *C* se obtiene al hacer la última cola BFS: *C* consiste de todos los vértices que en algún momento están en esa última cola. Como $x \in S_0 \Leftrightarrow in_f(x) = 0 \Leftrightarrow f(sx\rightarrow) = 0$, entonces os vértices de S_0 son precisamente todos los vértices que *s* agrega a la cola. Por lo tanto $S_0 \subseteq S$.

Todos los otros elementos de *S* deben haber sido agregados a la cola por algún elemento distinto de *s*. Pero un vértice solo puede agregar a algún vecino, ya sea vecino hacia adelante o hacia atrás. Esto quiere decir que los vértices de *S* no pueden haber sido agregados por otros vértices de *S*, pued como $S \subseteq X$, no hay lados entre ellos. Así que los vértices de *S* que no están en S_0 deben haber sido agregados por vértices de *T*. Pero $T \subseteq Y$ y todo vértice *y* de *Y* tiene $\Gamma^+(y) = \{ t \}$. Así que *T* sólo puede agregar a un vértice de *X* a la cola si lo agrega por medio de un lado **backward**. Para que *y* agregue a *x* como backward, debe pasar que $f(xy\rightarrow) > 0$. Esto implica $f(xy\rightarrow) = 1$ y que *xy* sea parte del matching. Por lo tanto, cada *y* que pueda agregar a la cola a algún vértice en forma backward, puede agregar UN SOLO tal vértice.

Entonces cada vértice de *T* puede agregar un solo vértice a la cola. Pero además, cada vértice de *T* efectivamente agrega un vértice a la cola. ¿Por qué?. Supongamos que no, que exista $y \in T$ que no agrega a nadie a la cola. Como *C* es un corte, $t \notin C$, a si que *y* no debería poder agregar a *t* a la cola. Con lo cual, debemos tener que $f(yt\rightarrow) = 1$. Entonces $out_f(y) = 1$ y por lo tanto $in_f(y) = 1$ y debe existir $x \in X$ tal que $f(xy\rightarrow) = 1$. Pero entonces *x* es “candidato” a ser agregado a la cola como backward por *y*.

Como estamos suponiendo que *y* no agrega a nadie a la cola, debe haber una razón para que *y* no agregue a *x*. La única razón posible es que *x* ya esté en la cola. No lo puede haber agregado alguien de *T* pues si fuese así *y* no lo agrega a si que debería ser otro vértice *z* de *T*, agregandolo backward, pero esto significaría que tanto *xz* como *xy* están en el matching, absurdo. Entonces la única posibilidad que queda es que haya sido agregado por *s*, es decir, que *x* esté en S_0. Pero si $x \in S_0$, entonces $out_f(x) = 0$ lo que contradice que *f*(*xy*→) = 1. Concluimos entonces que todo *y* en *T* agrega a alguien a la cola, y por lo que vimos antes, agrega **exactamente un** tal alguien a la cola. Esos elementos agregados son los de $S - S_0$, así que concluimos que $|S - S_0| = |T|$. Analicemos ahora los vértices de *T*, los cuales deben haber sido agregados por alguien de *X* de la cola, es decir, por alguien de *S*. Esto solo puede pasar si cada vértice de *T* es un vecino de algún vértice de *S*. Por lo tanto, $T \subseteq \Gamma(S)$. Si no son iguales, entonces existe $y \in \Gamma(S)$ tal que $y \notin T$. Como $y \in \Gamma(S)$ entonces existe $x \in S$ con $xy \in E$. Como $x \in S$, *x* está en *C*. Pero $y \notin T$, a si que *y* no está en *C*. A si que *x* nunca agregó a *y*. Pero $xy \in E$ a si que para que *x* no agregue a *y*, debe pasar que *f*(*xy*→) = 1. Pero no se sabe quien agregó a *x*: *y* no fue porque nunca estuvo en la cola, y no puede ser ningún otro elemento de *T* pues *f*(*xy*→) = 1 implica que no existe ningún otro *z* con *f*(*xz*→) = 1. Debe haber sido agregado por *s*, pero esto implica que $out_f(x) = 0$ absurdo pues *f*(*xy*→) = 1. Concluimos que $T = \Gamma(S)$. Entonces $|\Gamma(S)| = |T| = |S - S_0| = |S| - |S_0| < |S|$ (la última desigualdad pues $S_0 \neq \varnothing$. Hemos probado que $|\Gamma(S)| < |S|$, lo que contradice que $|S| \leq |\Gamma(S)|$ por hipótesis. Fin

(12) Enunciar y probar el teorema del matrimonio de König.

Todo grafo bipartito regular tiene un matching perfecto

Dado un conjunto $W \subseteq V$, definamos $E_W = \{zw \in E \mid w \in W\}$. Sean *X* e *Y* las partes del grafo bipartito. Supongamos $W \subseteq X$. Entonces:

$|E_W| = |\{zw \in E \mid w \in W\}| = \sum_{w \in W} |\{z : zw \in E\}|$. La última igualdad pues como $W \subseteq X$ entonces no estamos contando un lado *zw* dos veces, pues *zw* ∈ *E*implica que *z* ∈ *Y* y por lo tanto no puede estar en *W*. Entonces

$|E_W| = \sum_{w \in W} |\{z : zw \in E\}| = \sum_{w \in W} d(w)$. Como *G* es regular, *d*(*w*) = Δ para todo *w*, a si que tenemos $|E_W| = \sum_{w \in W} d(w) = \sum_{w \in W} \Delta = \Delta \cdot |W|$. También tenemos que si $W \subseteq Y$ entonces también vale que $|E_W| = \Delta \cdot |W|$. Si tomamos *W* = *X* tendremos que $|E_X| = \Delta \cdot |X|$. Pero *G* es bipartito, a si que $E_X = \{xy \in E \mid x \in X\} = E$. Concluimos que $|E| = \Delta \cdot |X|$.

Tomando *W* = *Y* concluimos $|E_Y| = \Delta \cdot |Y|$. Pero $E_Y = \Delta \cdot |Y|$. Entonces $|E|$ es igual a $\Delta \cdot |X|$y a $\Delta \cdot |Y|$. Por lo tanto $\Delta \cdot |X| = \Delta \cdot |Y|$ a si que $|X| = |Y|$. Entonces, como $|X| = |Y|$, para probar que existe un matching perfecto basta probar que existe un matching completo sobre *X*, y para esto usaremos el teorema de Hall.
Sea entonces $S \subseteq X$, y sea $l \in E_S$, entonces existe $x \in S$, $y \in Y$ tal que l es de la forma $l = xy$. Por lo tanto $y \in \Gamma(x) \subseteq \Gamma(S)$ (pues $x \in S$). Entonces *l*es de la forma *xy* con $y \in \Gamma(S)$, lo cual implica que $l \in E_{\Gamma(S)}$. Como l era cualquier elemento de E_S esto dice que $E_S \subseteq E_{\Gamma(S)}$.Por lo tanto $|E_S| \leq |E_{\Gamma(S)}|$.
Como $S \subseteq X$, entonces lo que probamos al principio de la prueba dice que $|E_S| = \Delta \cdot |S|$. Como $\Gamma(S) \subseteq Y$, entonces $|E_{\Gamma(S)}| = \Delta \cdot |\Gamma(S)|$. Como se probó que $|E_S| \leq |E_{\Gamma(S)}|$, concluimos que $\Delta \cdot |S| \leq \Delta \cdot |\Gamma(S)|$. Por lo tanto $|S| \leq |\Gamma(S)|$y como esto vale para cualquier $S \subseteq X$, Hall nos dice que existe un matching completo sobre *X* lo cual completa la prueba.