

Teoremas Para el Final

Matemática Discreta II

FaMAF – UNC 2024

1. Complejidad de Edmonds-Karp..... 1

2. Las distancias de Edmonds-Karp no disminuyen..... 3

3. Complejidad de Dinic (ambas versiones)..... 4

4. Complejidad de Wave (tarjan)..... 5

5. Max flow, Min cut..... 6

6. Baby Brooks..... 7

7. 2-COLOR es polinomial..... 8

8. Teorema de Hall..... 9

9. Teorema del Matrimonio de König..... 10

10. Cota de Hamming..... 11

11. $\delta(G) = \min\{\text{\#columnas LD de } H\}$ 12

12. Propiedades de un código cíclico..... 13

13. 3-SAT es NP-completo..... 15

14. 3-COLOR es NP-completo..... 17

15. Matrimonio3D es NP-completo..... 20

Complejidad de Edmonds-Karp

Teorema:

Edmonds-Karp termina siempre con complejidad $O(m^2n)$

Demostración:

Edmonds-Karp construye una sucesión de caminos aumentantes (de s a t) utilizando BFS a través de los cuales aumenta el flujo inicial, hasta que no existan más caminos. Por lo tanto su complejidad está dada por:

$$\text{compl}(\text{Edmonds Karp}) = O(\text{Compl}(\text{Construir CamAum} + \text{Aumentar Flujo}) * \#\text{Caminos Aumentantes})$$

Notar que $\text{compl}(\text{Aumentar Flujo}) = O(n)$ pues la longitud máxima de un camino formado por BFS es de $n-1$ lados y aumentar el flujo es $O(1)$ por cada lado. Mientras que $\text{compl}(\text{Construir CamAum}) = O(m)$ pues la construcción se realiza con BFS. Así,

$$\text{compl}(\text{Edmonds Karp}) = O(m * \#\text{Caminos Aumentantes})$$

Veremos que $\#\text{Caminos Aumentantes} = m * n$ y $\therefore \text{compl}(\text{Edmonds Karp}) = O(m * m * n)$

Para ello, definimos:

Un lado se vuelve **crítico** en un paso dado si es utilizado para construir el siguiente flujo en modo forwards y se satura, o en modo backwards y se vacía. De esta forma, cada camino aumentante vuelve crítico a al menos un lado.

Definimos:

Dado un flujo f en N y dos vértices x, z :

$$d(x, z) = 0, \text{ si } x = z$$

$$d(x, z) = \infty, \text{ si no existe un } f \text{ camino aumentante entre } x \text{ y } z$$

$$d(x, z) = \min\{\text{longitud de } f \text{ camino aumentante entre } x \text{ y } z\}, \text{ si existe}$$

Si f_0, f_1, \dots son los flujos generados por una corrida de *Edmonds Karp*, definimos:

$$d_k(x) = d_{f_k}(s, x) \text{ y } b_k(x) = b_{f_k}(x, t)$$

Proposición (no hace falta demostrarla):

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

Observación: $d_k(t) = d_k(x) + b_k(x)$ si ambas son finitas.

Si acotamos la cantidad de veces que un lado puede volverse crítico, acotaremos entonces la cantidad de caminos aumentantes posibles.

Sea $x \rightarrow y$ un lado que se vuelve crítico en el paso k . Esto puede ocurrir porque se saturó o se vació.

Si $x \rightarrow y$ se saturó:

Entonces se utilizó un f_k - camino aumentante de la forma $s \dots xy \dots t$ y este es de longitud mínima por utilizar Edmonds-Karp. Si el lado xy se vuelve a volver crítico en un paso $j > k$, entonces es por que se vacía (y se utiliza backwards en el paso j) o porque se vuelve a saturar en el paso j , lo que implica que se utilizó backwards devolviendo flujo en algún paso i con $k < i < j$. Es decir que para que un lado que se saturó en el paso k se vuelva a volver crítico en un paso j , debe ser utilizado backwards en un paso i con $k < i \leq j$. Por lo tanto se utilizó un f_i -camino aumentante de la forma $s \dots yx \dots t$ que también es de longitud mínima por utilizar Edmonds-Karp. Pero ahora

$$d_j(t) \geq d_i(t) = d_i(x) + b_i(x) = d_i(y) + 1 + b_i(x)$$

$$\geq d_k(y) + 1 + b_k(x) = d_k(x) + 1 + 1 + b_k(x) = d_k(t) + 2$$

Esto significa que la distancia de s a t debe aumentar en al menos 2 para que un lado se pueda volver a volver crítico una vez que se satura.

Si $x \rightarrow y$ se vació:

Entonces se utilizó un f_k -camino aumentante de la forma $s \dots yx \dots t$ y este es de longitud mínima por utilizar Edmonds-Karp. Si el lado $x \rightarrow y$ se vuelve a volver crítico en un paso $j > k$, entonces es por que se satura (y se utiliza forwards en el paso j) o porque se vuelve a vaciar en el paso j , lo que implica que se utilizó forwards mandando algo de flujo en algún paso i con $k < i < j$. Es decir que para que un lado que se vació en el paso k se vuelva a volver crítico en un paso j , debe ser utilizado forwards en un paso i con $k < i \leq j$.

Por lo tanto se utilizó un f_i -camino aumentante de la forma $s \dots xy \dots t$ que también es de longitud mínima por utilizar Edmonds-Karp. Pero ahora

$$d_j(t) \geq d_i(t) = d_i(y) + b_i(y) = d_i(x) + 1 + b_i(y) \\ \geq d_k(x) + 1 + b_k(y) = d_k(y) + 1 + 1 + b_k(y) = d_k(t) + 2$$

Esto significa que la distancia de s a t debe aumentar en al menos 2 para que un lado se pueda volver a volver crítico una vez que se vacía.

Por lo tanto la cantidad de veces que un lado puede volverse crítico es $O(n)$, pues la longitud de los caminos varía entre 1 y $n-1$ y cada vez que se vuelve crítico debe aumentar en 2.

Como cada camino aumentante de Edmonds-Karp vuelve crítico al menos un lado, hay m lados y cada lado puede volverse crítico $O(n)$ veces, entonces $\# \text{Camino Aumentantes} = O(m * n)$.

Las distancias de Edmonds-Karp no disminuyen

Teorema:

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

Demostración:

Conjunto A = vértices que no cumplen la regla

Sup $A \neq \text{vacío}$

Elijo el x con menor d_{k+1} en A

Existe f_{k+1} -camino aumentante $s \dots zx$ en el paso $k+1$ ($d_{k+1}(x) < d_k(x) \leq \text{inf}$) y s no está en $A \Rightarrow$

hay un z anterior a x y además $d_{k+1}(z) = d_{k+1}(x) - 1$

Como $d_{k+1}(z) < d_{k+1}(x) \Rightarrow z$ no está en A entonces $d_k(z) \leq d_{k+1}(z)$

$d_k(z) \leq d_{k+1}(z) < \text{inf} \Rightarrow$ hay un f_k -camino aumentante $p_k: s \dots z$ en el paso k

Si zx es un lado entonces lo podríamos agregar en modo forwards al camino $s \dots z$ en el paso k pero esto cumpliría

$$d_k(x) = d_k(z) + 1 \leq d_{k+1}(z) + 1 \leq d_{k+1}(x) \text{ absurdo pues } x \text{ está en } A$$

Luego el lado zx debe estar saturado, pero en el paso $k+1$ ya no lo está pues hay un camino aumentante $s \dots zx$ entonces se debe haber utilizado backwards entre f_k y f_{k+1} (en el paso k) con un f_k -camino aumentante de la forma $s \dots xz \dots t$

pero esto implica $d_k(z) = d_k(x) + 1 > d_{k+1}(x) + 1 = d_{k+1}(z) + 1 + 1 \geq d_k(z) + 2$ **absurdo**

Si xz es un lado podríamos agregar x backwards al camino $s \dots z$ en el paso k pero esto cumpliría

$$d_k(x) = d_k(z) + 1 \leq d_{k+1}(z) + 1 \leq d_{k+1}(x) \text{ absurdo pues } x \text{ está en } A$$

Luego el lado xz debe estar vacío pero no lo está más en el paso $k+1$

\Rightarrow se usó forwards en el paso k con un camino de la forma $s \dots xz \dots t$

lo cual cumple $d_k(z) = d_k(x) + 1 > d_{k+1}(x) + 1 = d_{k+1}(z) + 1 + 1 \geq d_k(z) + 2$ **absurdo**

Luego A es vacío.

Complejidad de Dinic (ambas versiones)

Teorema:

La complejidad del algoritmo de dinic, tanto “original” como “occidental” es $O(n^2m)$

Demostración:

$O(\text{cant NA (construir + hallar flujo bloqueante)})$

$O(N * (m + ?))$

versión original

? = DFS en $O(n)$ y cada DFS bloquea un lado $\Rightarrow O(m)$ veces $\Rightarrow O(mn)$

PODAR \Rightarrow hay que podar luego de cada camino ($O(m)$ podar donde se ven en $O(1)$ los $O(n)$ vertices)

podar es $O(mn)$ + eliminar los lados de los vertices = $O(m)$ pues $\sum \{\text{grado}(v)\} = O(m)$

versión occidental con pseudo código del paso de hallar flujo bloqueante:

```

g = 0, flag = 1;
while (flag)
  p = [s]; x = s;
  while (x != t && flag)
    if (hay vecinos de adelante de x)
      y = un vecino
      agrego y al camino p          “A”  $O(1)$ 
      x = y
    else
      if (x != s)
        y = anterior a x en p
        eliminar x de P y el lado yx del NA    “R”  $O(1)$ 
        x = y
      else
        flag = 0
  if (x == t)
    aumentar flujo g y eliminar los lados saturados del NA    “C”  $O(n)$ 
return g

```

Una corrida de este algoritmo será de la forma A...ARA...ACA...AR...

el paso A es $O(1)$ pues simplemente agrega a una lista

el paso R es $O(1)$ pues simplemente elimina de una lista

el paso C es $O(n)$ pues aumentar flujo es $O(n)$ y no puede haber más de n lados en $p[]$

Cada R elimina un lado y cada C elimina al menos un lado \Rightarrow sólo puede haber $O(m)$ A...AX en la corrida del algoritmo, con $X = C$ o R

Cuántas veces ocurre A en cada paso? Cada A avanza un nivel en el NA, y hay $O(n)$ niveles por lo tanto hay $O(n)$ A's por cada paso

Así, hallar flujo bloqueante = $\text{compl}(O(m) * \text{compl}(A...AX))$

$\text{compl}(A...AX) = \text{compl}(A...A) + \text{compl}(X) = O(n)*O(1) + O(1)*O(n)$ en el peor caso donde $X=C$

finalmente $\text{compl}(\text{hallar flujo bloqueante}) = O(m * n)$

Complejidad de Wave (tarjan)

Teorema:

El algoritmo wave de tarjan tiene complejidad $O(n^3)$

Demostración:

$$\begin{aligned}\text{compl}(\text{wave}) &= O(\text{cant NA} * (\text{construirlos} + \text{hallar bloqueante})) \\ &= O(N * (m + ?))\end{aligned}$$

notar que $O(m) + O(n^2) = O(n^2)$ pues $m < nC2 = n^2 - n / 2$

conjuntos

ola forwards, enviando flujo cada lado se puede saturar o no

$S = \{\text{casos donde los lados se saturan}\}$

$P = \{\text{casos donde no se saturan}\}$

ola backwards, devolviendo flujo cada lado se puede vaciar o no

$V = \{\text{casos donde los lados se vacían}\}$

$Q = \{\text{casos donde no se vacían}\}$

Análisis de S

Hay $O(m)$ lados, cuántas veces se podrían saturar? supongamos que un lado se saturó, para que se vuelva a saturar, entonces tendrá que haber devuelto flujo (para poder enviar y saturar de nuevo), pero si devolvió flujo entonces está bloqueado y no podrá volver a mandar, ni saturarse

$$S = O(m)$$

Análisis de V

Hay $O(m)$ lados, cuántas veces se podrían vaciar? supongamos que un lado se vació, para que se vuelva a vaciar, entonces tendrá que haber enviado flujo (para poder devolver y vaciarse de nuevo), pero si devolvió flujo la primera vez entonces está bloqueado y no podrá volver a mandar, ni vaciarse otra vez

$$V = O(m)$$

Análisis de P

En cada ola hacia adelante, para cada vértice, se saturan todos sus lados salvo quizás el último, y enviar flujo por el es $O(1)$

$$P = O(n) * \# \text{olas hacia adelante}$$

Analisis de Q

En cada ola hacia atrás, para cada vértice, se vacían todos sus lados salvo quizás el último, y devolver flujo por el es $O(1)$

$$Q = O(n) * \# \text{olas hacia atrás}$$

olas hacia adelante = # olas hacia atrás y en cada ola hacia adelante, salvo quizás en la última, se desbalancea y por lo tanto se bloquea un vértice que no se vuelve a desbloquear, por lo tanto hay $O(n)$ olas hacia adelante.

$$\text{compl}(P) = \text{compl}(Q) = O(n * n)$$

$$\text{Luego compl}(\text{hallar flujo bloqueante en cada NA}) = O(2 * m + 2 * n^2) = O(n^2)$$

Max flow, Min cut

Teorema:

El valor de todo flujo es menor o igual a la capacidad de todo corte
 flujo $f \Rightarrow f$ es maximal $\Leftrightarrow v(f) = \text{cap}(S)$ para algún corte S (y el corte es minimal)

Demostración:

Lema importante: $v(f) = f(S, \text{complemento}(S)) - f(\text{complemento}(S), S)$

Con ese lema es fácil ver la parte 1

parte 2, f flujo

f maximal \rightarrow construyo $S = \{s\} \cup \{x : \text{existe un } f\text{-camino aumentante entre } s \text{ y } x\}$

$s \in S$

t no está en S porque podría aumentar el flujo y no sería maximal

usando lema, $v(f) = f(S, \text{complemento}(S)) - f(\text{complemento}(S), S)$

$f(S, \text{complemento}(S)) = \text{suma } f(xy)$ y existe un camino aumentante hasta x , pero no hasta y , pero existe el lado xy entonces xy debe estar saturado $\Rightarrow f(xy) = c(xy)$ para todo $xy \Rightarrow f(S, \text{complemento}(S)) = \text{cap}(S)$

$f(\text{complemento}(S), S) = \text{suma } f(xy)$ donde existe un camino aumentante hasta y pero no hasta x , pero existe el lado xy que se podría usar backwards para hacer un camino aumentante hasta x es decir xy debe estar vacío $\Rightarrow f(xy) = 0$ para todo $xy \Rightarrow f(\text{complemento}(S), S) = 0$

luego $v(f) = \text{cap}(S)$

S corte y f flujo con $v(f) = \text{cap}(S)$

en la parte 1 vimos que si g es un flujo entonces $v(g) \leq \text{cap}(S) = v(f)$ por lo tanto f es maximal

Notar que S es minimal pues también por 1 $\text{cap}(T) \geq v(f) = \text{cap}(S)$ para todo T y f

Baby Brooks

Teorema:

Si G es un grafo conexo no regular, $\chi(G) \leq \Delta(G)$

Demostración:

Colorear con Greedy utilizando el orden DFS(x) inverso donde x es el de menor grado $d < D$.

Utilizar greedy garantiza que el coloreo es propio, veamos que siempre habrá un color disponible entre 1 y D para todos los vértices con este orden.

En DFS, cada vértice desde x agrega a todos sus vecinos posteriores i.e todos los vértices $\neq x$ tienen al menos un vecino anterior.

En el **orden inverso**, este vecino será un vecino **posterior** y por lo tanto aún no coloreado por greedy.

Luego los vértices distintos de x tienen como máximo tienen **$D - 1$ vecinos ya coloreados**

\Rightarrow habrá un color disponible en $\{1, \dots, D\}$ para ellos.

Cuando llegamos a x , no tiene vecinos posteriores (es el último en el orden inverso), pero $d(x) = d < D$

\Rightarrow tendrá un color disponible que ninguno de sus vecinos utilice en $\{1, \dots, D\}$

Luego el coloreo es propio y utiliza a lo sumo D colores $\Rightarrow \chi(G) \leq D$

2-COLOR es polinomial

Teorema:

Se puede determinar que un grafo G es bipartito o “2-coloreable” determinísticamente en tiempo polinomial sobre el tamaño del grafo.

Demostración:

Corremos BFS(x) a partir de algún vértice x de G en tiempo $O(m)$ y obtenemos los niveles de cada vértice en BFS.

Coloreamos G con el coloreo c tal que $c(v) = \text{NIVEL}_{\text{BFS}}(v) \bmod 2$

Chequeamos si es propio, esto es, para cada vértice, chequear que todos sus vecinos tengan colores distintos a él. Se hace en $O(\text{suma de grados}) = O(m)$

Si es propio $\Rightarrow X(G) \geq 2$ fin.

Si no es propio, existe un lado vw con $\text{NIVEL}_{\text{BFS}}(v) \bmod 2 = \text{NIVEL}_{\text{BFS}}(w) \bmod 2$

Es decir que sus niveles tienen la misma paridad y por lo tanto la suma de sus niveles sería un número PAR.

Por lo obtenido en BFS, existen en el grafo caminos $x \dots v$ y $x \dots w$ los cuales se separan a partir de algún vértice z , es decir que son de la forma $x \dots z \dots v$ y $x \dots z \dots w$ lo cual implica la existencia de un ciclo

$z \dots vw \dots z$ en G , ¿Cuántos lados tiene?

$z \dots v$ tiene $\text{NIVEL}_{\text{BFS}}(v) - \text{NIVEL}_{\text{BFS}}(z)$ lados

vw es 1 lado

$w \dots z$ tiene $\text{NIVEL}_{\text{BFS}}(w) - \text{NIVEL}_{\text{BFS}}(z)$ lados

En total, el ciclo tiene $\text{NIVEL}_{\text{BFS}}(v) - \text{NIVEL}_{\text{BFS}}(z) + 1 + \text{NIVEL}_{\text{BFS}}(w) - \text{NIVEL}_{\text{BFS}}(z)$

Pero la suma de los niveles de v y w es un valor par como dijimos antes, $2 \cdot \text{NIVEL}_{\text{BFS}}(z)$ es par, y al sumarle 1 significa que el ciclo de G tiene una cantidad IMPAR de lados por lo tanto $X(G) \geq 3$.

Teorema de Hall

Teorema:

En un grafo G bipartito con partes X e Y , existe un matching “completo” de X a $Y \Leftrightarrow$ para todo S contenido en X , $|\Gamma(S)| \geq |S|$

Demostración:

ida: el matching induce una biyección f entre los elementos de X con sus vecinos en Y , por lo que $|f(S)| = |S|$ para todo S contenido en X y $f(S)$ está contenido en $\Gamma(S)$ por lo que $|\Gamma(S)| \geq |S|$ para todo S contenido en X

vuelta: suponemos la condicion de hall, para todo S contenido en X , $|\Gamma(S)| \geq |S|$

Supongamos que al correr el algoritmo para hallar matching maximal, hallamos un matching M con $|E(M)| < |X|$, es decir que no cubre a X

Corremos el algoritmo en su forma matricial de nuevo sobre M , obteniendo un conjunto S de filas etiquetadas y un conjunto T de columnas etiquetadas.

Definimos

$S_0 = \{ \text{filas etiquetadas con } * \text{ en el primer paso} \}$

Y para $i = 1, \dots, k$

$T_i = \{ \text{columnas etiquetadas por } S_{i-1} \}$

$S_i = \{ \text{filas etiquetadas por } T_i \}$

donde k es el paso en el que frena el algoritmo, de esta manera es claro que

$S = \text{unión (disjunta) de } S_i$

$T = \text{unión (disjunta) de } T_i$

Notar que el algoritmo nunca frena al pasar de un T_i a un S_i con $i=1, \dots, k$ pues una columna etiquetada está libre y extiende el matching (lo cual no ocurre en este caso), o etiqueta a la fila con la que está matcheada.

Por ello, existen la misma cantidad de estos conjuntos (k) sin contar el S_0 .

Notar también que cada columna de T_i etiqueta únicamente a una fila (agrega únicamente una fila) al conjunto S_i y por lo tanto $|S_i| = |T_i|$ para todo $i = 1, \dots, k$

Luego como las uniones son disjuntas,

$$\begin{aligned} |S| &= |S_0| + |S_1| + \dots + |S_k| \\ &= |S_0| + |T_1| + \dots + |T_k| \\ &= |S_0| + |T| \\ &> |T| \end{aligned}$$

Veamos ahora que $T = \Gamma(S)$

T son las columnas etiquetadas, y las columnas son etiquetadas por filas de S , pero cada fila de S etiqueta únicamente a columnas vecinas, por lo que $x \in T \Rightarrow x$ es vecino de alguna fila de $S \Rightarrow x \in \Gamma(S)$

De esta forma, S es un subconjunto de X que cumple $|S| > |\Gamma(S)|$ (violando la condición de hall que supusimos cierta) por lo cual esto es un absurdo de asumir que $|E(M)| < |X|$.

Finalmente $|E(M)| = |X|$ pues tampoco puede ser mayor.

Teorema del Matrimonio de König

Teorema:

Todo grafo bipartito regular tiene un matching perfecto.

Demostración:

Sea G un grafo bipartito regular con $d = D = \delta(G)$ con partes X e Y

Definimos, para un conjunto W contenido en V , $E_w = \{ xy \in E(G) : x \in W \}$

Luego, tomamos un subconjunto de vértices $S \subseteq X$

$xy \in E_s \Rightarrow y \in \Gamma(x) \Rightarrow y \in \Gamma(S) \Rightarrow xy \in E_{\Gamma(S)}$

Por lo tanto E_s contenido en $E_{\Gamma(S)} \Rightarrow |E_s| \leq |E_{\Gamma(S)}|$

¿Cuál es la cardinalidad de E_w si $W \subseteq X$ ó $W \subseteq Y$?

$xy \in E_w \Rightarrow y$ no está en W

pues $W \subseteq X \Rightarrow y \in Y$ y $W \subseteq Y \Rightarrow y \in X$

Luego $E_w =$ unión disjunta de lados de cada uno de los vértices de w

$|E_w| =$ suma de grados de vértices de W , pero todos tienen el mismo grado por lo que
 $= D * |W|$

Luego utilizando $|E_s| \leq |E_{\Gamma(S)}|$ obtenemos $D * |S| \leq D * |\Gamma(S)|$ y por lo tanto
 $|S| \leq |\text{vecinos}(S)|$ para todo S contenido en X .

Luego el teorema de Hall aplica indicando que existe un matching completo de X a Y .

Notar que $|X| \leq |\Gamma(X)| \leq |Y|$, pero la elección de X sobre Y fue arbitraria por lo tanto también se cumple la condición de Hall para todo S contenido en Y , lo cual implica que $|Y| \leq |X|$

Finalmente $|X| = |Y|$ y el matching “completo” es “perfecto”.

Cota de Hamming

Teorema:

Sea C un código binario en $\{0,1\}^n$, de longitud N , $d = d(c)$ y $t = \text{floor}((d-1)/2)$

luego $|C| \leq 2^n / \sum(nCr)_{r=1,\dots,t}$

Demostración:

Definimos A contenido en $\{0,1\}^n$ como la unión de los discos de radio t al rededor de las palabras de C , esto es:

$A = \text{union } \{ D_t(v) \} \text{ para } v \text{ en } C$

Como C corrige t errores, la unión es disjunta y

$|A| = \text{suma de } |D_t(v)| \text{ para todo } v \text{ en } C$

Observamos que $D_t(v) = \text{unión } \{ S_r(v) \} \text{ para } r = 0, \dots, t \text{ donde } S_r(v) = \{ x \text{ en } \{0,1\}^n : d_H(v, x) = r \}$

Y la unión es disjunta pues no hay dos palabras iguales que estén a distintas distancias de v para ningún v .

Luego

$|D_t(v)| = \text{suma de } |\{ S_r(v) \}| \text{ para } r = 0$

Como $S_r(v)$ son las palabras que difieren en r bits de v , hay una biyección entre los conjuntos de subconjuntos de r bits y los elementos de $S_r(v)$

Así,

$|S_r(v)| = |\text{conjunto de subconjuntos de } r \text{ bits de los } n \text{ posibles}| = nCr$

Juntando todo,

$|A| = \text{suma de } |D_t(v)| \text{ para todo } v \text{ en } C$

$|D_t(v)| = \text{suma de } |\{ S_r(v) \}| \text{ para } r = 0, \dots, t$

$|S_r(v)| = nCr \leftarrow \text{independiente de } v$

luego

$|D_t(v)| = \text{suma de } |nCr| \text{ para } r = 0, \dots, t \leftarrow \text{independiente de } v$

$|A| = \text{suma de (suma de } nCr \text{ para } r=0,\dots,t) \text{ para todo } v \text{ en } C \text{ pero al ser independiente de } v \text{ entonces}$

$|A| = |C| * \text{suma de } nCr \text{ para } r=0,\dots,t$

Así,

$|C| = |A| / \text{suma de } nCr \text{ para } r=0,\dots,t \text{ y } |A| \leq 2^n$

\Rightarrow

$|C| \leq 2^n / \text{suma de } nCr \text{ para } r=0,\dots,t$

$$\delta(G) = \min\{\# \text{columnas LD de } H\}$$

Teorema:

Si H es la matriz de chequeo de un código C , entonces $\delta(G) = \min\{\# \text{columnas LD de } H\}$

Demostración:

$$m \leq d$$

$d \Rightarrow$ una palabra x distinta de 0 en C tiene d unos pues $d = \min\{|x| : x \in C, x \neq 0\}$

$$\Rightarrow Hx^t = H(e_{i_1} + \dots + e_{i_d})^t = H(e_{i_1})^t + \dots + H(e_{i_d})^t = \text{suma de } d \text{ columnas} = 0$$

$$\Rightarrow d \text{ columnas LD} \Rightarrow m \leq d$$

$$d \leq m$$

$$m \Rightarrow \text{suma de } m \text{ columnas} = 0 \Rightarrow H(e_{i_1} + \dots + e_{i_m})^t = 0$$

\Rightarrow la palabra $(e_{i_1} + \dots + e_{i_m})$ está en el código pues está en $\text{Nu}(H)$ y es distinta de 0 pues $m \geq 1$

$$\Rightarrow \text{como } d = \min\{|x| : x \in C, x \neq 0\}, \text{ entonces } d \leq m$$

Propiedades de un código cíclico

Teorema:

Si C es un código cíclico de longitud n y dimensión k , y g es su polinomio generador, entonces:

- I. $C = \{p(x) : \text{gr}(p) < n \text{ \& } g(x) \mid p(x)\}$
- II. $C = \{v(x) * g(x) \bmod (1+x^n) : v(x) \text{ es un polinomio cualquiera}\}$
- III. $\text{gr}(g) = n - k$
- IV. $g(x) \mid (1+x^n)$

Demostración:

Definimos:

$$C_1 = \{p(x) : \text{gr}(p) < n \text{ \& } g(x) \mid p(x)\}$$

$$C_2 = \{v(x) * g(x) \bmod (1+x^n) : v(x) \text{ es un polinomio cualquiera}\}$$

Para demostrar i) e ii), veremos que $C_1 \subseteq C_2 \subseteq C \subseteq C_1$ y por lo tanto todos serán iguales.

$$C_1 \subseteq C_2$$

Dada p en C_1 , sabemos que $\text{gr}(p) < n$ por lo que $p(x) \bmod (1+x^n) = p(x)$
 como $g(x) \mid p(x)$ entonces existe un polinomio $q(x)$ tal que
 $p(x) = g(x)q(x)$
 tomando módulo,
 $p(x) \bmod (1+x^n) = g(x)q(x) \bmod (1+x^n)$
 $p(x) = g(x)q(x) \bmod (1+x^n)$ está en C_2

$$C_2 \subseteq C$$

Dada p en C_2 , sabemos que $p(x) = v(x) * g(x) \bmod (1+x^n)$ para algún polinomio $v(x)$
 luego
 $p(x) = (v_0 + v_1x + v_2x^2 + \dots + v_dx^d) * g(x) \bmod (1+x^n)$ para algún d
 $= v_0 * g(x) + v_1x * g(x) + \dots + v_dx^d * g(x) \bmod (1+x^n)$
 $= v_0 * g(x) + v_1 \text{rot}(g) + \dots + v_d \text{rot}^d(g)$
 y todos los términos son palabras de C por lo tanto $p(x)$ está en C .

$$C \subseteq C_1$$

Dada p en C , sabemos que $\text{gr}(p) < n$ por lo que $p(x) \bmod (1+x^n) = p(x)$
 dividir p por g nos da un $q(x)$ y $r(x)$, con $\text{gr}(r) < \text{gr}(g)$ tal que
 $p(x) = g(x)q(x) + r(x)$
 tomando módulo,
 $p(x) \bmod (1+x^n) = g(x)q(x) + r(x) \bmod (1+x^n)$
 $p(x) = g(x)q(x) \bmod (1+x^n) + r(x)$
 $r(x) = p(x) + g(x)q(x) \bmod (1+x^n)$

pero $p(x)$ en C y $g(x)q(x) \bmod (1+x^n)$ en $C_2 \subseteq C \Rightarrow r(x)$ en C pero $\text{gr}(r) < \text{gr}(g)$

$\Rightarrow r(x) = 0$ pues g es el único polinomio de menor grado en C

$\Rightarrow \text{gr}(p) < n \text{ \& } g(x) \mid p(x)$

$\Rightarrow p(x)$ en C_1

Veamos ahora iii)

Como C es lineal de dimension k , $|C| = 2^k$

Por definición, C_1 incluye las palabras de grado menor a n que sean múltiplos de $g(x)$. Esto induce una biyección entre las palabras de C_1 y los polinomios de grado $< n - \text{gr}(g)$, pues para cada una de las palabras de C_1 existe un $q(x)$ con grado $< n - \text{gr}(g)$ tal que $g(x)q(x) = p(x)$ sea de grado $< n$

Luego $|C| = |C_1| = |\text{polinomios de grado } < n - \text{gr}(g)| = 2^{n - \text{gr}(g)}$

$$\Rightarrow 2^k = 2^{n - \text{gr}(g)}$$

$$\Rightarrow k = n - \text{gr}(g)$$

$$\Rightarrow \text{gr}(g) = n - k$$

Veamos, finalmente, iv)

Dividir $(1+x^n)$ por $g(x)$ nos da dos polinomios

$q(x)$ y $r(x)$ con $\text{gr}(r) < \text{gr}(g)$ tal que $1+x^n = g(x)q(x) + r(x)$

Tomando módulo,

$$1+x^n \bmod (1+x^n) = g(x)q(x) + r(x) \bmod (1+x^n)$$

$$0 = g(x)q(x) \bmod (1+x^n) + r(x)$$

$$r(x) = g(x)q(x) \bmod (1+x^n)$$

El término de la derecha es una palabra de C_2 i.e una palabra de C , por lo tanto $r(x)$ en C y $\text{gr}(r) < \text{gr}(g)$, como vimos anteriormente, significa que $r(x) = 0$ por las propiedades de g .

Finalmente concluimos que $g(x) \mid (1+x^n)$

3-SAT es NP-completo

Teorema:

3-SAT es NP-Completo (3-SAT = 3-CNF-SAT)

Demostración:

Para demostrar que 3-SAT es NP completo, mostraremos que CNF-SAT se reduce polinomialmente a 3-SAT, y como CNF-SAT es NP-completo, entonces 3-SAT también lo será.

Para ello, debemos construir una instancia $A(B)$ de 3-SAT a partir de una instancia B de CNF-SAT, en tiempo polinomial sobre el tamaño de B .

Una instancia B con variables x_1, \dots, x_n de CNF-SAT es de la forma $B = D_1 \& \dots \& D_m$, donde $D_j = l_{j1} \vee \dots \vee l_{j(r_j)}$ donde l_{jk} son literales i.e. variables o negaciones de variables.

Queremos transformarla en $A(B) = E_1 \& \dots \& E_m$ donde cada E_j será una conjunción de disyunciones de 3 variables i.e. una instancia de 3-CNF-SAT.

Además queremos, dado $b = (b_1, \dots, b_n)$ vector de asignación de variables x_1, \dots, x_n
 $B(b) = 1 \Leftrightarrow$ Existe d (asignaciones de y_{jk}) tal que $[A(B)](b)(d) = 1$

i.e B es satisfacible $\Leftrightarrow A(B)$ lo es

Notar que como B es una conjunción de D_j 's, $B(b) = 1 \Leftrightarrow D_j(b) = 1$ para $j=1, \dots, m$

Para obtener $A(B)$, transformaremos cada D_j en un E_j de la siguiente manera:

Si $r_j = 1 \rightarrow E_j = (l_{rj} \vee y_{j1} \vee y_{j2}) \& (l_{rj} \vee \neg y_{j1} \vee y_{j2}) \& (l_{rj} \vee y_{j1} \vee \neg y_{j2}) \& (l_{rj} \vee \neg y_{j1} \vee \neg y_{j2})$

Si $r_j = 2 \rightarrow E_j = (l_{j1} \vee l_{j2} \vee y_{j1}) \& (l_{j1} \vee l_{j2} \vee \neg y_{j1})$

Si $r_j = 3 \rightarrow E_j = D_j$

Hasta aquí, es claro que $E_j(b) = 1 \Leftrightarrow D_j(b) = 1$ pues las variables agregadas **no pueden** hacer todos los términos verdaderos a la vez i.e. lo deben hacer las variables de D_j

Si $r_j \geq 4 \rightarrow E_j = (l_{j1} \vee l_{j2} \vee y_{j1}) \& (\neg y_{j1} \vee y_{j2} \vee l_{j3}) \& (\neg y_{j2} \vee y_{j3} \vee l_{j3}) \& \dots \& (\neg y_{j(r_j-3)} \vee y_{j(r_j-2)} \vee l_{j(r_j-2)}) \& (\neg y_{j(r_j-2)} \vee l_{j(r_j-1)} \vee l_{j(r_j)})$

Notar que la construcción de cada E_j es en tiempo polinomial sobre el tamaño de cada D_j ya que la cantidad de variables agregadas es lineal en la cantidad de literales en D_j . Luego la construcción de $A(B)$ completa es en tiempo polinomial sobre el tamaño de B .

Veamos ahora que dado b vector de asignación de variables y $r_j \geq 4$,

$D_j(b) = 1 \Leftrightarrow$ Existe d (asignaciones de y_{jk}) tal que $E_j(b)(d) = 1$

(\Rightarrow)

Dado un j en $\{1, \dots, m\}$ y $b = b_1, \dots, b_n$ asignación de variables

$D_j(b) = 1 \Rightarrow$ Existe un literal $l_{j(kj)}$ tal que $l_{j(kj)}(b) = 1$, si hay más de uno tomamos el primero.

Construimos $d = (d_1, \dots, d_r)$ con $d_1 = \dots = d_{kj-2} = 1$ y $d_{kj-1} = \dots = d_{rj-2} = 0$, donde $y_{jh}(d) = d_h$

$$\begin{aligned}
 E_j = & (l_{j1} \vee l_{j2} \vee y_{j1})(b)(d) \\
 & \& (\neg y_{j1} \vee y_{j2} \vee l_{j3})(b)(d) \\
 & \& \dots \\
 & \& (\neg y_{j1} \vee y_{j2} \vee l_{j3})(b)(d) \leftarrow \text{Todos los términos anteriores a este son 1 pues tienen un } d_h \text{ con } 1 \leq h \leq kj-2 \\
 & \& (\neg y_{j(kj-2)} \vee y_{j(kj-1)} \vee l_{jkj})(b)(d) \leftarrow \text{Este término es } =1 \text{ pues } l_{jkj}(b) = 1 \\
 & \& (\neg y_{j(kj)} \vee \text{---})(b)(d) \leftarrow \text{Desde aquí hasta el final, todos los términos son 1 pues comienzan con} \\
 & \& \dots \qquad \qquad \qquad \text{un } \neg d_h = 1 \text{ con } jk \leq h \leq rj-2 \\
 & \& (\neg y_{j(rj-3)} \vee y_{j(rj-2)} \vee l_{j(rj-2)})(b)(d) \\
 & \& (\neg y_{j(rj-2)} \vee l_{j(rj-1)} \vee l_{j(rj)})(b)(d)
 \end{aligned}$$

Luego queda demostrado que dado b que hace verdadero a D_j existe un d tal que junto con el b dado hacen verdadero al E_j correspondiente.

(\Leftarrow) Veamos ahora que dado $b = (b_1, \dots, b_n)$, $d = (d_1, \dots, d_r)$ vectores de asignaciones de x_i e y_i respectivamente,

$$E_j(b)(d) = 1 \Rightarrow D_j(b) = 1$$

Supongamos que no, es decir que existen $b = (b_1, \dots, b_n)$ y $d = (d_1, \dots, d_r)$ tal que $E_j(b)(d) = 1$ pero $D_j(b) = 0$

$$D_j(b) = 0 \Rightarrow l_{jh}(b) = 0 \text{ para todo } h=1, \dots, rj$$

luego

$$\begin{aligned}
 E_j(b) = 1 = & (l_{j1} \vee l_{j2} \vee y_{j1}) \& (\neg y_{j1} \vee y_{j2} \vee l_{j3}) \& (\neg y_{j2} \vee y_{j3} \vee l_{j3}) \& \dots \& (\neg y_{j(rj-3)} \vee y_{j(rj-2)} \vee l_{j(rj-2)}) \& (\neg y_{j(rj-2)} \vee l_{j(rj-1)} \vee l_{j(rj)}) \\
 = & (0 \vee 0 \vee y_{j1}) \& (\neg y_{j1} \vee y_{j2} \vee 0) \& (\neg y_{j2} \vee y_{j3} \vee 0) \& \dots \& (\neg y_{j(rj-3)} \vee y_{j(rj-2)} \vee 0) \& (\neg y_{j(rj-2)} \vee 0 \vee 0) \\
 = & y_{j1} \& (\neg y_{j1} \vee y_{j2}) \& \dots \& (\neg y_{j(rj-3)} \vee y_{j(rj-2)}) \& (\neg y_{j(rj-2)}) \\
 = & 0 \text{ pues si fuese } 1, y_{j1} = 1 \Rightarrow y_{j(rj-2)} = 1 \text{ pero } (\neg y_{j(rj-2)}) = 1, \text{ absurdo}
 \end{aligned}$$

Concluimos entonces por (\Rightarrow) y (\Leftarrow) que D_j es satisfacible $\Leftrightarrow E_j$ lo es y por lo tanto B es satisfacible $\Leftrightarrow A(B)$ lo es.

3-COLOR es NP-completo

Teorema:

3-COLOR es NP-Completo

Demostración:

Para demostrar que 3-COLOR es NP-Completo, veremos que 3(-CNF)-SAT se reduce polinomialmente a 3-COLOR, y como 3-SAT es NP-Completo, entonces 3-COLOR también lo será.

Para ello, debemos dar un algoritmo A polinomial en el tamaño de una instancia B de 3-SAT de manera tal que $A(B) = G$ sea una instancia de 3-COLOR y que B sea satisfacible $\Leftrightarrow X(G) \leq 3$

Como B es una instancia de 3-SAT, es de la forma $B = D_1 \& \dots \& D_m$ con $D_j = l_{j1} \vee l_{j2} \vee l_{j3}$ donde cada l_{jk} es una variable en $\{x_1, \dots, x_n\}$ o su negación.

Construimos $A(B) = G$:

Vértices:

$$V(G) = \{u_i, w_i\} \cup \{a_{j1}, a_{j2}, a_{j3}, e_{j1}, e_{j2}, e_{j3}\} \cup \{s, t\} \text{ para } i = 1, \dots, n; j = 1, \dots, m$$

Lados:

$$\begin{aligned} E(G) = & \{ a_{j1}a_{j2}, a_{j2}a_{j3}, a_{j3}a_{j1} \} \leftarrow \text{Triángulos de } a\text{'s} \\ & \cup \{ a_{jr}e_{jr} \} \leftarrow \text{"garras" de los triángulos de } a\text{'s} \\ & \cup \{ tu_i, tw_i, u_iw_i \} \leftarrow \text{Triángulos de } u\text{'s, } w\text{'s y } t \\ & \cup \{ st \} \\ & \cup \{ se_{jr} \} \\ & \cup \{ e_{jr}v(l_{jr}) \} \end{aligned}$$

para todo $j = 1, \dots, m$; $i = 1, \dots, n$; $r = 1, 2, 3$;

donde $v(l_{jr}) = u_i$ si Existe i tal que $l_{jr} = x_i$
 $= w_i$ si Existe i tal que $l_{jr} = \neg x_i$

Es claro que la construcción de G es en tiempo polinomial pues las cantidades de lados y vértices son lineales en n, m y r (tamaño de B).

Veamos ahora que B es satisfacible $\Leftrightarrow X(G) \leq 3$

Notar primero, que

B es satisfacible

\Leftrightarrow

Existe $b = (b_1, \dots, b_n)$ asignaciones de variables tal que $B(b) = 1$

\Leftrightarrow

$D_j(b) = 1$ para todo $j = 1, \dots, m$

(\Rightarrow) Asumimos que B es satisfacible i.e Existe $b = (b_1, \dots, b_n)$ tal que $D_j(b) = 1$ para todo $j = 1, \dots, m$

Queremos ver que $X(G) \leq 3$, para ello, daremos un coloreo c propio de G con 3 colores.

Colorearemos el grafo G paso a paso, asegurándonos de que nunca deje de ser un coloreo propio.

Como st es un lado $\Rightarrow c(s) \neq c(t)$

Asignamos $c(s) = \text{SALMON}$ y $c(t) = \text{TURQUESA}$ y de esta manera el lado st mantiene lo propio del coloreo.

Dado un j en $\{1, \dots, m\}$ Como $D_j(b) = 1$ entonces existe un k_j tal que $I_{j(k_j)}(b) = 1$. Si hay más de uno tomamos uno solo. Ahora coloreamos:

$c(a_{j(k_j)}) = \text{TURQUESA}$

y para $j \neq k_j$, $c(a_{j_r}) = \text{uno SALMON y el otro NEGRO}$

De esta forma los "triángulos de a 's" utilizan los 3 colores i.e mantienen lo propio del coloreo.

Seguimos con

$c(e_{j(k_j)}) = \text{NEGRO}$

y para $j \neq k_j$, $c(e_{j_r}) = \text{TURQUESA}$

de esta forma el lado $a_{j(k_j)}e_{j(k_j)}$ mantiene lo propio del coloreo (uno TURQUESA y otro NEGRO)

y los lados $e_{j_r}a_{j_r}$ para $r \neq k_j$ también mantienen lo propio del coloreo pues e es TURQUESA y los a serán SALMON o NEGRO.

Definimos ahora,

$c(u_i) = \text{SALMON si } b_i = 1 \text{ y NEGRO si } b_i = 0$

$c(w_i) = \text{NEGRO si } b_i = 1 \text{ y SALMON si } b_i = 0$

Revisemos los lados

$e_{j_r}v(l_{j_r})$ para $r \neq k_j$ mantienen lo propio del coloreo pues $c(e_{j_r}) = \text{TURQUESA}$ y $l_{j_r} = w_{\text{algo}}$ o u_{algo} utiliza el color NEGRO o SALMON.

Nos queda ver $e_{j(k_j)}v(l_{j(k_j)})$, recordemos que $I_{j(k_j)}(b) = 1$

Si $l_{j(k_j)}$ es una variable, Existe i tal que $l_{j(k_j)} = x_i$ y por lo tanto $v(l_{j(k_j)}) = u_i$

Ahora, $1 = I_{j(k_j)}(b) = x_i(b) = b_i \Rightarrow c(u_i) = \text{SALMON}$

como $c(e_{j(k_j)}) = \text{NEGRO}$, este lado no crea problemas

Si $l_{j(k_j)}$ es una negación de variable, Existe i tal que $l_{j(k_j)} = \neg x_i$ y por lo tanto $v(l_{j(k_j)}) = w_i$

Ahora, $1 = I_{j(k_j)}(b) = \neg x_i(b) = 1 - b_i \Rightarrow b_i = 0 \Rightarrow c(u_i) = \text{NEGRO}$

Como $w_i u_i$ es un lado, $c(w_i) \neq \text{NEGRO}$ y $c(e_{j(k_j)}) = \text{NEGRO}$

por lo que el lado $e_{j(k_j)}v(l_{j(k_j)}) = e_{j(k_j)}w_{j(k_j)}$ mantiene lo propio del coloreo.

Luego B satisfacible \Rightarrow Existe un coloreo propio de $A(B) = G$ con 3 colores $\Rightarrow X(G) \leq 3$.

(\leq) Asumimos $X(G) \leq 3$, como $G = A(B)$ tiene triángulos (K_3) formados por a 's, entonces $X(G) \geq 3$.
Esto nos indica que $X(G) = 3$

Sea c un coloreo propio de G con 3 colores, construiremos $B = (b_1, \dots, b_n)$ tal que $B(b) = 1$

Definimos, para ello, $b_i = 1$ si $c(u_i) = c(s)$ y $b_i = 0$ en caso contrario.

Veamos ahora que $B(b) = 1$, para ello, necesitamos que $D_j(b) = 1$ para todo $j = 1, \dots, m$ y por lo tanto necesitamos que exista, para cada j , un k_j tal que $I_{j(k_j)}(b) = 1$

Fijamos un j en $\{1, \dots, m\}$

Los triángulos formados por a 's utilizan los 3 colores, por lo tanto elegimos k_j tal que $c(a_{j(k_j)}) = c(t)$

Como $a_{j(k_j)}e_{j(k_j)}$ es un lado, $c(e_{j(k_j)}) \neq c(t)$

Como $se_{j(k_j)}$ es un lado, $c(e_{j(k_j)}) \neq c(s)$

y como ts es un lado $c(t) \neq c(s) \Rightarrow c(e_{j(k_j)}) = 3er\ color$

$e_{j(k_j)}v(I_{j(k_j)})$ un lado y $c(e_{j(k_j)}) = 3er\ color \Rightarrow c(v(I_{j(k_j)})) \neq 3er\ color$

$v(I_{j(k_j)}) = w_i$ o u_i para algún $i \Rightarrow tv(I_{j(k_j)})$ es un lado $\Rightarrow c(v(I_{j(k_j)})) \neq c(t)$

Luego $c(v(I_{j(k_j)})) = c(s)$

Si $I_{j(k_j)}$ es una variable x_i entonces $I_{j(k_j)}(b) = x_i(b) = b_i$ y $v(I_{j(k_j)}) = u_i$

$\Rightarrow c(v(I_{j(k_j)})) = c(u_i) = c(s)$

$\Rightarrow I_{j(k_j)}(b) = b_i = 1$

Si $I_{j(k_j)}$ es una negación de variable $\neg x_i$ entonces $I_{j(k_j)}(b) = \neg x_i(b) = 1 - b_i$

y $v(I_{j(k_j)}) = w_i$

$\Rightarrow c(w_i) = c(v(I_{j(k_j)})) = c(s)$ y $w_i u_i$ lado

$\Rightarrow c(u_i) \neq c(s)$

$\Rightarrow b_i = 0$

$\Rightarrow I_{j(k_j)}(b) = 1 - b_i = 1$

Por lo tanto $X(G) \Rightarrow$ Existe b tal que $B(b) = 1 \Rightarrow B$ es satisficible.

Matrimonio3D es NP-completo

Teorema:

Matrimonio3D es NP-COMPLETO

Demostración:

Para demostrarlo, mostraremos que 3(-CNF)-SAT se reduce polinomialmente a Matrimonio3D y como 3-SAT es NP-Completo entonces Matrimonio3D también lo será.

Es decir, a partir de una instancia B con variables x_1, \dots, x_n de la forma $B = D_1 \& \dots \& D_m$ con $D_j = l_{j1} \vee l_{j2} \vee l_{j3}$ construiremos en tiempo polinomial sobre n y m , un 3-hipergrafo $A(B) = H$ tal que Existe un matching perfecto en $H \Leftrightarrow B$ es satisfacible.

Construcción de H:

Vértices:

$$X = \{a_{ij}\}_{ij} \cup \{s_j\}_j \cup \{g_k\}_k$$

$$Y = \{b_{ij}\}_{ij} \cup \{t_j\}_j \cup \{h_k\}_k$$

$$Z = \{u_{ij}, w_{ij}\}_{ij}$$

para $i = 1, \dots, n$; $j = 1, \dots, m$ y $k = 1, \dots, m(n-1)$

De esta manera, $|X| = mn + m + m(n-1) = 2mn = |Y| = |Z|$

Lados: $E_0 \cup E_1 \cup E_2 \cup E_3$

$$E_0 = \{ \{ a_{ij}, b_{ij}, u_{ij} \} \}_{ij}$$

$$E_1 = \{ \{ a_{i(j+1)}, b_{ij}, w_{ij} \} \}$$

$$E_2 = \{ \{ g_k, h_k, z \} : z \in Z \}$$

y definimos $v(l_{jr}) = u_{ij}$ si existe i tal que $l_{jr} = x_i$
 $= w_{ij}$ si existe i tal que $l_{jr} = \neg x_i$

con ello,

$$E_3 = \{ \{ s_j, t_j, v(l_{jr}) \} \}$$

para todos los lados, $i = 1, \dots, n$; $j = 1, \dots, m$ y $k = 1, \dots, m(n-1)$ y $r = 1, 2, 3$

Veamos ahora si

Existe un matching perfecto en $H \Leftrightarrow B$ es satisfacible

(\Rightarrow) Asumimos que existe un matching M perfecto en H

Como M es perfecto, entonces cubre todos los vértices. En particular, debe cubrir a los vértices a_{ij} por lo tanto existe un lado L en $E(M)$ tal que L en E_0 ó L en E_1

Si L en $E(M)$ n $E_0 \Rightarrow L$ es de la forma $\{ a_{ij}, b_{ij}, u_{ij} \}$. Como M es un matching (\cdot : una biyección), el lado b_{ij} no puede aparecer en otro lado de $E(M) \Rightarrow \{ a_{i(j+1)}, b_{ij}, w_{ij} \}$ no está en $E(M)$. Pero $a_{i(j+1)}$ **debe estar** en $E(M)$ lo cual significa que $\{ a_{ij}, b_{ij}, u_{ij} \}$ para todo $j = 1, \dots, m$. Llamamos esta situación el “CASO 0” para i .

Similarmente, si L en $E(M)$ n $E_1 \Rightarrow L$ es de la forma $\{ a_{i(j+1)}, b_{ij}, w_{ij} \}$. Como M es un matching (\cdot : una biyección), el lado b_{ij} no puede aparecer en otro lado de $E(M) \Rightarrow \{ a_{ij}, b_{ij}, u_{ij} \}$ no está en $E(M)$. Pero a_{ij} **debe estar** en $E(M)$ lo cual significa que $\{ a_{i(j+1)}, b_{ij}, w_{ij} \}$ para todo $j = 1, \dots, m$. Llamamos esta situación el “CASO 1” para i .

Definimos entonces un vector de asignación de variables $b = (b_1, \dots, b_n)$ de la siguiente manera:

$b_i = 0$ si se da el **CASO 0** para i

$= 1$ si se da el **CASO 1** para i

Veamos ahora que $B(b) = 1$ y por lo tanto que B es satisficible.

Notar que $B(b) = 1 \Leftrightarrow D_j(b) = 1$ para todo $j=1, \dots, m$

Fijando un j , $D_j(b) = 1 \Leftrightarrow$ Existe un r tal que $l_{jr}(b) = 1$

Como M es un matching perfecto, **debe** cubrir a los lados $\{s_j, t_j, v(l_{jr})\}$ para algún valor de $r = r_j$ pues son los únicos en donde aparecen los vértices s_j y t_j . Tomamos este valor r_j y veremos que $l_{j(r_j)}(b) = 1$

Si $l_{j(r_j)}$ es una variable x_i entonces $l_{j(r_j)}(b) = x_i(b) = b_i$

entonces $v(l_{j(r_j)}) = u_{ij} \Rightarrow \{s_j, t_j, v(l_{j(r_j)})\} = \{s_j, t_j, u_{ij}\}$ por lo que u_{ij} no puede estar en otro

lado del matching $\Rightarrow \{a_{ij}, b_{ij}, u_{ij}\}$ no está en el matching \Rightarrow **CASO 1** para $i \Rightarrow b_i = 1$

luego $l_{jr}(b) = x_i(b) = b_i = 1$.

Si $l_{j(r_j)}$ es una negación de variable $\neg x_i$ entonces $l_{j(r_j)}(b) = \neg x_i(b) = 1 - b_i$

entonces $v(l_{j(r_j)}) = u_{ij} \Rightarrow \{s_j, t_j, v(l_{j(r_j)})\} = \{s_j, t_j, w_{ij}\}$ por lo que w_{ij} no puede estar en otro

lado del matching $\Rightarrow \{a_{i(j+1)}, b_{ij}, w_{ij}\}$ no está en el matching \Rightarrow **CASO 0** para $i \Rightarrow b_i = 0$

luego $l_{jr}(b) = x_i(b) = 1 - b_i = 1$.

(\Leftarrow) Asumimos que B es satisficible, debemos ver que existe un matching perfecto en H . Para ello, construiremos un matching M con $E(M) = F_0 \cup F_1 \cup F_2 \cup F_3$ y mostraremos que **cubre** a todos los vértices de H .

$F_0 = \{\{a_{ij}, b_{ij}, u_{ij}\} : j = 1, \dots, m \text{ y } b_i = 0\}$

$F_1 = \{\{a_{i(j+1)}, b_{ij}, w_{ij}\} : j = 1, \dots, m \text{ y } b_i = 1\}$

Es claro que todos los lados de F_0 y F_1 son disjuntos pues i no puede ser 0 y 1 a la vez.

Como $B(b) = 1 \Rightarrow D_j(b) = 1$ para todo $j=1, \dots, m$

\Rightarrow Para todo $j=1, \dots, m$ Existe un r_j tal que $l_{j(r_j)}(b) = 1$

$F_3 = \{\{s_j, t_j, v(l_{j(r_j)})\}\}$

Es claro que son disjuntos entre sí pues hay un único r_j por cada j .

Veamos que los lados de F_3 son disjuntos con todos los lados anteriores:

Si no lo fueran, el problema lo causaría $v(l_{j(r_j)})$ pues s y t no forman parte de ningún lado anterior.

Si $l_{j(r_j)}$ es una variable $x_i \Rightarrow$

$1 = l_{j(r_j)}(b) = x_i(b) = b_i = 1$

\Rightarrow el vértice u_{ij} no está en F_0 para ningún j

pero $v(l_{j(r_j)}) = u_{ij}$

\Rightarrow Todos los lados de F_3 donde $l_{j(r_j)}$ es una variable son disjuntos con los de con F_0 y (obviamente) con los de F_1 pues no tienen posibilidad de coincidir.

Si $l_{j(r_j)}$ es una negación de variable $\neg x_i \Rightarrow$

$1 = l_{j(r_j)}(b) = \neg x_i(b) = 1 - b_i \Rightarrow b_i = 0$

\Rightarrow el vértice w_{ij} no está en F_1 para ningún j (para ese i)

pero $v(l_{j(r_j)}) = w_{ij}$

\Rightarrow Todos los lados de F_3 donde $l_{j(r_j)}$ es una negación variable son disjuntos con los de con F_1 y (obviamente) con los de F_0 pues no tienen posibilidad de coincidir.

Finalmente, definimos $N = \{ z \text{ en } Z : z \text{ **no está** cubierto por } M \}$

$|N|$? Veamos $|Z - N| = |\{ z \text{ en } Z : z \text{ **está** cubierto por } M \}|$

F_3 cubre m vértices en Z , pues cubre a uno por cada $j = 1, \dots, m$

Si $p = \#\{i : b_i = 0\}$ y $q = \#\{i : b_i = 1\}$ entonces

F_0 cubre $p \cdot m$ vértices en Z

F_1 cubre $q \cdot m$ vértices en Z

Obs: $p + q = n$

Luego $|Z - N| = m + p \cdot m + q \cdot m = (1 + p + q) \cdot m = m \cdot (n+1)$

Por lo que $|N| = |Z| - |Z - N| = 2mn - m \cdot (n+1) = m \cdot (2n - n - 1) = m(n-1)$

Lo cual nos indica que existe una biyección f entre $\{ 1, \dots, m(n-1) \}$ y los lados de N

Definimos entonces $F_2 = \{ \{g_k, h_k, f(k)\} \}$ los cuales son **claramente** disjuntos de todos los lados anteriores pues g y h no estaban cubiertos, y $f(k)$ en $N \Rightarrow$ tampoco estaba cubierto. También son disjuntos entre sí pues f es una biyección.

De esta forma y como todos los conjuntos dados son disjuntos,

$$\begin{aligned} |E(M)| &= |F_0| + |F_1| + |F_2| + |F_3| \\ &= pm + qm + m(n-1) + m \\ &= nm + m(n-1) + m \\ &= 2mn = |X| = |Y| = |Z| \end{aligned}$$

Y por lo tanto M es un matching perfecto.