

Demonstraciones.

3) Complejidad algoritmo Dinic-Original vs Dinic-Even.

- La $d(s,t)$ entre NA consecutiva aumenta, nos a lo sumo Δ NA

- Se $CFB \geq 2$ complejidad de encontrar flujo bloqueante en un NA.

Un algoritmo "tipo" Dinic tiene complejidad.

$$(N^{\Delta} M^{\Delta})^*(C_{\text{cada NA}} + C_{FB}) = O(n^*(m + C_{FB}))$$

Hay que ver que $C_{FB} = O(mn)$

Original

- Cada camino entre s y t se encuentra con DFS, pero si el vértice v no tiene backtracking (vertice con lado entrante "tiene lado saliente")
- para mantener esto se debe ir revisando los NA, "Podar"
- debemos encontrar complejidad de: encontrar caminos + podar.

Construir un camino dirigido desde s a t es muy fácil:

- Tomar $p = s$.
- WHILE($p \neq t$)
 - Tomar q cualquier vecino de p . (*)
 - Agregar pq al camino.
 - Tomar $p = q$
- ENDWHILE

- como cada camino sativa y poda tanto borra o meno un lado, nos a lo sumo m caminos.

Complejidad encontrar caminos = $O(mn)$

Complejidad PODAR:

podar consiste en recorrer los vértices y eliminar los que no tienen salida,

$\sim PV$: recorrer vérticos = $O(n)$ chequea todos los vértices una sola vez

$\sim B(n)$: eliminar lados de entrada de un vértice X . \int por vértice.

hay un PV en cada PODAR ("podar por") a lo sumo $m+1$ PV.

Complejidad todos PV $O(nm)$

la construcción depende de la cantidad de niveles, entonces es $O(n)$. Luego se borran los lados saturados en el network

$O(n)$ de vueltas.

Si lo tanto borra o meno

Complejidad $B(x) = \Theta(d(x))$, y se hace una sola vez por x .

Complejidad del conjunto $B(X) = \Theta(\sum_x d(x)) = \Theta(2m) = \Theta(m)$

Compactidad CFB:

encontrar - caminar + poder

$$\Theta(nm) + \Theta(\partial V) + \Theta(B(X)) = \Theta(nm) + \Theta(nm) + \Theta(m) \\ = \Theta(nm)$$

Complejidad Dinic-Original = $\Theta(n \times (m + nm))$

$$= \Theta(mn^2)$$

Dinic-Even

- No tiene poder, enodo DFJ puede tener que hacer backtrace.

Tenemos que ver el pseudocódigo con el que se encuentran flujos bloquantes en una NA.

$g = 0$

STOPFLAG:=1//para saber cuando parar

WHILE (STOPFLAG) //while externo

| $p = [s], x = s$ // Inicialización inicial de x y del camino p

| WHILE (($x \neq t$) AND (STOPFLAG)) //while interno

| | IF $\Gamma^+(x) \neq \emptyset$ THEN AVANZAR(x)

| | ELSE IF ($x \neq s$) THEN RETROCEDER(x)

| | ELSE STOPFLAG=0

| IF ($x == t$) THEN INCREMENTAR

RETURN(g)

Avanzar > retroceder, $\Theta(1)$.

Incrementar $\Theta(n)$, largo max del camino es N e incrementa flujo a lo largo del camino y borra las saturaciones.

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.

podemos dividir la suc de palabras en
AA, AI x
ROI

Cada palabra termina en ROI, y tanto ROI
borran o meno un lado, cont de palabras a lo sumo m.

Complejidad de cada palabra:

R, A $\Theta(1)$, I $\Theta(n)$, si la palabra AA \times tiene k A's, la complejidad sera $\Theta(k+1) = \Theta(k)$ o $\Theta(k+n) = \Theta(n)$, pero como cada A significa aumentar 1 nivel en la NA, entonces $k \leq n$, por lo que la ROI la complejidad de la palabra es $\Theta(n)$, y como hay a lo sumo m palabras, entonces la complejidad de encontrar un flujo bloquante es $\Theta(nm)$, la de Dinic-Even $\Theta(mn^2)$

4) Completidad Wave.

- Sabemos que hay ≥ 10 sumos en NA y la cantidad de niveles aumenta.
- hay que ver que la complejidad de encontrar un flujo bloqueante en una NA es $\mathcal{O}(n^2)$ entonces la la complejidad de wave es $\mathcal{O}(n)$. $\mathcal{O}(n^2) = \mathcal{O}(n^3)$.

Prueba:

en Wave hacemos serie de Olos hacia adelante y hacia atrás.

Llámalo hacia adelante hacer una serie de FB(x)

Llámalo hacia atrás hacer una serie de BB(x)

por cada uno verifiquemos $\Gamma^+(x)$ o $M(x)$.

en general cuando miramos un $y \in \Gamma^+(x)$ pueden pasar 2 cosas:

- 1- \overrightarrow{xy} se satura, S cont total sobre todas las olas hacia adelante.
- 2- no se satura, P cont total sobre todas las olas hacia adelante.

Cuando miramos $y \in M(x)$ pueden darse 2 cosas 1- \overrightarrow{xy} se vacia 2- no se vacia.

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.

S.

- Supongamos que un lado \overrightarrow{xy} se satura entonces lo borramos del NA.
Como nunca mas lo agregamos, nunca mas lo saturaremos. (se saturan una vez)

S acotado por m.

No seguimos de que es correcto borrarlo porque para poder usarlo otra vez primero debe des-saturarse, y le devuelvo flujo a x.

y solo le pude devolver flujo a x solo si y esta bloquedado, pero si el flujo bloqueado x no puede mandarle mas flujo a y, entonces \overrightarrow{xy} no puede volver a ser usado.

V

Supongamos que \overrightarrow{yx} se vacia.

Pero yx sólo se puede vaciar si x esta bloqueado pues de otra forma no podria devolverle flujo a y.

Pero si x esta bloqueado, el vértice "y" nunca mas puede mandarle flujo.

Si yx nunca mas puede recibir flujo, entonces menos aun va a poder volver a vaciarse.

Asi que V tambien està acotado por el numero total de lados, m.

En cada $FB(x)$ buscamos vecinos de x y les mandamos todo el flujo que podamos:

$$\min\{D(x), c(\vec{xy}) - g(\vec{xy})\}.$$

Si ese mínimo es igual a $c(\vec{xy}) - g(\vec{xy})$ el lado \vec{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 sólo tal que ese mínimo es $D(x)$.

Es decir, al hacer $FB(x)$, todos los lados \vec{xy} que miramos, salvo a lo sumo uno, se saturan.

Concluimos entonces que en cada FB hay a lo sumo UN lado procesado como parte de P .

Por lo tanto, tenemos a lo sumo $n - 2$ FB por cada ola hacia adelante, y tenemos $O(n)$ olas hacia adelante, significa que tenemos en total $O(n^2)$ FB y por lo tanto, la cantidad de procesamientos de P es $O(n^2)$.

Similarmente, al hacer un BB , a lo sumo un lado no se vacía, así que Q es la cantidad total de BB 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 $O(n^2)$.

por lo que decimos que Φ está acotado superiormente, por la const de FB .

en cada ola hay a lo sumo $n - 2$ FB .

Cuantas olas hace adelante hoy?

en cada ola menos la última al menos un vértice se bloquea, si no bloquea, entonces quedan 2 todos, y en la última, entonces hay a lo sumo n olas hacia adelante.

entonces la complejidad de para bloquear es

$$S + P + R + Q = O(m) + O(n^2) + O(m) + O(n^2) \approx O(n^2).$$

implica que la complejidad de WAVE es $O(n^3)$

5) La distancia en NA sucesivo aumenta

la cantidad de niveles de un NA es menor a la cantidad de niveles de un network posterior.

Prueba:

- Sea NA un network auxiliar y NA' un network sucesivo.

- Sea f el flujo (del network original) inmediatamente anterior a NA y f' el inmediatamente anterior a NA' .

- Sea $d(rx) = d_f(r, x) > d'(rx) = d_{f'}(r, x)$, sabemos que $d(t) \leq d'(t)$ por la prueba de EK.

Queremos ver que vale a <.

Si $t \in NA' \Rightarrow d'(t) = \infty > d(t) \Rightarrow$ ya esto, asumir $t \in NA'$

$\Rightarrow \exists$ C-dicho x_0, x_1, \dots, x_r entre $r > t$ en NA' , pero no tiene st camino en NA .

- Si lo fuera, al construir f' habriamos saturado ese camino. (es decir, saturar al menos un lado).
- Pues para terminar con NA y pasar de f a f' debemos saturar todos los caminos de NA .
- Pero si quedó saturado, no podría ser un camino en NA' .

Sí el c-dikigido no está en NA se debe a 2 cosas.

1. Algun x_i NO está en NA.

2. Existe **Todos** los x_i pero no algun lado $\overrightarrow{x_i x_{i+1}}$.

1. Como $t \in NA \Rightarrow x_i \neq t \wedge$ todos los vértices $\neq t$ que están a distancia mayor o igual que t no se incluyen, pero todos los que tienen menor están por NA se construye con BFS.

- La única forma que x_i no esté en NA es que $d(t) \leq d(x_i)$ (1)

> Como sabemos que $d \leq d'$ $\Rightarrow d(x_i) \leq d'(x_i)$ (2)

Como $x_0 x_1 \dots x_{i-1} x_i$ es un camino en NA y NA' es un net por niveles, concluimos que:

• $d'(x_i) = i \quad \forall i$. (3), Además $x_i \neq t \Rightarrow i < t$ (4)

Entonces:

$$d(t) \stackrel{(1)}{\leq} d(x_i) \stackrel{(2)}{\leq} d'(x_i) \stackrel{(3)}{=} i < t \stackrel{(4)}{=} d'(t)$$

> Probaremos que la distancia entre t y t aumenta para el caso?

2.

Todos los $x_i \in NA$ pero $\overrightarrow{x_i x_{i+1}} \notin NA$.

- Sabemos que $d(x_{i+1}) \leq d'(x_{i+1})$, tenemos 2 subcasos que sea $\subset 0 =$

• $d(x_{i+1}) < d'(x_{i+1})$ (5), Sea $b(x) = b_f(x, t)$, $b'(x) = b_{f'}(x, t)$

Por EFK sabemos $b = b'$ y $d(t) = d(x) + b(x)$, igual para $d' \sim b'$

$$d(t) = d(x_{i+1}) + b(x_{i+1})$$

$$\leq d(x_{i+1}) + b'(x_{i+1}) \quad (\text{EFK})$$

$$\stackrel{(5)}{\leq} d'(x_{i+1}) + b'(x_{i+1}) = d'(t), \quad \text{probamos } d(t) < d'(t)$$

• Supongamos $d(x_{i+1}) = d'(x_{i+1}) = i+1$, como i es el primer i para el cual $\overrightarrow{x_i x_{i+1}} \notin NA$.

Entonces al camino $x_0 x_1 \dots x_i$ Si está en NA $\Rightarrow d(x_i) = i$

Entonces, en NA, x_i está en nivel i y x_{i+1} en nivel $i+1$.

Podemos concluir que ambos lados $\overrightarrow{x_i x_{i+1}}$ y $\overrightarrow{x_{i+1} x_i}$ NO ESTAN EN NA.

Como $d(x_i) = i$, $d(x_{i+1}) = i+1$. Entonces x_i, x_{i+1} están a distancia "legitima" para que exista en lado $\overrightarrow{x_i x_{i+1}}$

■ Pero ese lado no está en NA. Por lo que concluimos que:

- 1 $\xrightarrow{x_i x_{i+1}}$ es lado del network original pero está saturado, o:
- 2 $\xrightarrow{x_{i+1} x_i}$ es lado del network original pero está vacío.

■ Pero $x_i x_{i+1}$ si es un lado en NA' .

■ Así que la situación es:

- 1 $\xrightarrow{x_i x_{i+1}}$ es lado del network original, estaba saturado al construir NA pero des-saturado al construir NA' , o
- 2 $\xrightarrow{x_{i+1} x_i}$ es lado del network original, estaba vacío al construir NA pero no vacío al construir NA' .

La única forma en que pase [1] es que al pasar de f a f' dessaturamos el lado $\xrightarrow{x_i x_{i+1}}$, es decir, devolvemos flujo, lo que dice que usamos en algún momento el lado como backward.

Como pasamos de f a f' usando NA , esto significa que en NA debemos haber usado el $\xrightarrow{x_{i+1} x_i}$.

Esto es un absurdo pues habíamos visto que ese lado no puede estar en NA , pues estaríamos yendo del nivel $i+1$ al i .

La única forma en que pase [2] es que al pasar de f a f' mandamos algo de flujo por el lado $\xrightarrow{x_{i+1} x_i}$.

Así que ese lado también debe ser un lado en NA , lo cual como dijimos es un absurdo.

Fin Prueba



6) El valor de todo flujo es menor o igual que la capacidad de todo corte
que si f es un flujo, entonces las sig afirmaciones son eq:
i) \exists corte S tq $v(f) = \text{cap}(S)$ y en este caso S es minimal
ii) f es maximal
iii) No existen f -camino aumentantes.

Primero probamos $v(f) \leq \text{cap}(S)$, $\forall f, S$: f flujo y S corte, usando que $v(f) = f(S, \bar{S}) - f(\bar{S}, S)$.

Luego probamos $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$

Prueba:

- $v(f) \leq \text{cap}(S)$, $\forall S$ corte

$$\begin{aligned} v(f) &= f(S, \bar{S}) - f(\bar{S}, S) \\ &\leq f(S, \bar{S}) \\ &\leq c(S, \bar{S}) \\ &= \text{cap}(S). \end{aligned}$$

$$\therefore v(f) \leq \text{cap}(S)$$

1 \Rightarrow 2. Sean f, S como en $1 \Rightarrow 3$ en flujo cuando

sabemos que $v(g) \leq \text{cap}(S) \quad \forall S$, entonces

$$v(g) \leq \text{cap}(S) = v(f) \quad \text{para CUALQUIER FLUJO } g$$

entonces f es maximal, tambien

Sea T un corte, sabemos $\text{cap}(T) \geq v(f) = \text{cap}(S)$
entonces S es minimal.

2 \Rightarrow 3. Dsuminuir 2 veremos 3.

Sea f un flujo maximal, si existe un f -camino aumentante, entonces podríamos mandar $\epsilon > 0$ a través de él, y obtener un flujo f' tal que $v(f') = v(f) + \epsilon \Rightarrow v(f)$ pero es **aburrido**, pues f es maximal por hipótesis.

3 \Rightarrow 1

No existen f -caminos aumentantes, entonces existe un S tal que $\text{cap}(S) = v(f)$.

definimos un S como: $S = \{x \in V : \exists f\text{-ca de } S \ni x\}$
como visto 3 $\Rightarrow T \not\subseteq S$, S es un corte.

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

calculemos

$$f(S, \bar{S}) = \sum f(x, y) [x \in S] [y \notin S] [x \in E], \text{ veamos que el } f\text{-ca de } S \ni x \text{ existe}$$

pero a y no, esto significa que a lazo x está saturado, entonces

$$f(x, y) = c(x, y)$$

$$f(\bar{S}, S) = \sum f(x, y) [x \notin S] [y \in S] [x \in E], \text{ veamos que como } y \in S \text{ y } x \notin S,$$

el camino $S = x_0, x_1, \dots, x_k = y$, x **no** es aumentante, pero podía serlo si se usara el lazo x como backward, pero si no lo usara, o porque $f(x, y) = 0$, entonces.

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

$$\text{Luego } v(f) = \text{cap}(S) - 0$$

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



fin prueba \rightarrow queda demostrado que las tres afirmaciones son equivalentes.

7) Probar que 2-color es polinomial.

dado un grafo G , tomar un vértice $x \in G$ y lo colorear con 0. Luego hacer BFS, colorear con 0 si el dist de x es par, 1 impar. Luego veremos si el colorado es propio. Hay que hacerlo con un x , si al terminar BFS faltan vértices por colorear hacerlo al mismo con algún x' que parte hasta terminar.

Complejidad:

- elegir un vértice $\mathcal{O}(1)$
- BFS $\mathcal{O}(m)$
- colorear $\mathcal{O}(1)$.
- en general si es propio = recorrer todos los lados $\Rightarrow \mathcal{O}(m)$.

2-colorable o polinomial, hay que checarlo la correctitud.

Prueba:

Si al terminar encontramos que un vértice v, z con el mismo color, debe significar que existe un ciclo impar de la forma $w \dots v \dots z \dots w$. Y este ciclo tiene distancia

$$d(w,v) + d(v,z) + d(z,w) = d(w,v) + 1 + d(z,w) \quad (1)$$

Como $C(v) = C(z)$ entonces $d(x,v) \equiv d(x,z) \pmod{2}$.

Tomar a ver que $d(w,v) \equiv d(w,z) \pmod{2}$.

$$\begin{aligned} d(x,v) &\equiv d(x,z) \pmod{2} \\ d(x,v) + d(w,v) &\equiv d(x,z) + d(w,z) \pmod{2} \\ d(v,w) &\equiv d(w,z) \pmod{2} \quad (2) \end{aligned}$$

Ahora veremos que el ciclo formado es impar.

$$\begin{aligned} &= d(w,v) + 1 + d(z,w) \quad (1) \\ &= d(w,v) + 1 + d(z,w) \pmod{2} \text{ tomar } \pmod{2} \\ &= (d(w,v) \pmod{2}) + 1 + (d(z,w) \pmod{2}) \pmod{2} \\ &= (d(w,v) \pmod{2}) + 1 + (d(v,w) \pmod{2}) \pmod{2} \\ &= 2(d(w,v) \pmod{2}) + 1 \pmod{2} \\ &= 1 \pmod{2} \end{aligned}$$

Y el ciclo es impar, por ende G no es bipartido \Rightarrow no es 2-colorable.



8) Enunciar y probar el teorema de Hall

Teorema de Hall: Sea $G = (X \cup Y, E)$ un grafo bipartito,

$$|S| \leq |\Gamma(v)|, \forall v \in X \implies \exists \text{ un matching completo}$$

Prueba.

Supongamos que no es cierto, es decir que $|S| > |\Gamma(N)|$ y $S \subseteq X$ pero no existe un matching completo sobre X .

Veamos que el hecho de que no existe matching completo sobre X implica $|S| > |\Gamma(S)|$.

Sea S_0 aquellos vértices de X , que a término de algún vértice y en Y .

- $S_0 = \{x \in X : \text{in}(x) = \text{out}(x) = 0\}$

$x \in S_0 \implies \exists y \in Y : \vec{xy} \in E(M)$, significa que $f(xy) = 0 \quad \forall y$, donde f es el flujo máximo. $\implies \text{out}(x) = 0 \rightarrow$ por ende $\text{in}(x) = 0$

También vale que si x es vértice tal que $\text{in}(x) = 0$
- para $\text{out}(x) = 0$ por lo tanto $f(xy) = 0$.

y x no forma parte del matching.

- Sea C el resto mínimo obtenido al restar $E - T$, obtendremos que ese resto incluye a S . Y los otros elementos son vértices que quedan entre X y Y .

$$C = \{S\} \cup S \cup T, \quad S \subseteq X, \quad T \subseteq Y.$$

Como $x \in S_0 \implies \text{in}(x) = 0 \implies f(\vec{sx}) = 0$, todos los vértices de S_0 son todos los vértices que se agregan a la cola. Por lo tanto $S_0 \subseteq S$.

Esto nos dice que tener los demás elementos (que no están en S_0) no deben haber sido agregados por s . Esto quiere decir que a los demás vértices de S no se agregó ningún vértice de T , pues $S \subseteq X$ y no hay边 entre S .

Pero en T $\forall y \in T$ tiene $\Gamma^+(y) = \emptyset$, así que un vértice de T no puede agregar vértices de X a la cola por medio de un lado **backward**.

Deberá pasar $f(\vec{xy}) > 0$, y en matching implica $f(\vec{xy}) = 1$. Y $\vec{xy} \in E(M)$ por lo tanto cada y que pueda agregar un vértice a la cola en forma **backward**, solo puede agregar **UN VÉRTICE**.

- Vamos que efectivamente cada vértice de T agrega un vértice a la cola.

- Supongamos que no y que $\exists y \in T$ tal que no agrega a nadie
- Como C es resto $T \not\subseteq C$, tenemos $f(yT) = 1$, por lo tanto $\text{out}(y) = 1 \implies \text{in}(y) = 1$. Entonces debe existir $x \in X$ tal que $f(\vec{xy}) = 1$. Es decir suponemos que x puede ser agregado a la cola como **backward** por y .

- Pero como y NO agrega, entonces debemos suponer que x y y estén en la cola.
- Algunas de T NO lo agregan porque significa que $\exists z \in T$ tal que $f(xz) = 1$, lo que implica que xy y xz están en el matching, absurdo.
- La única posibilidad es que x esté en S_0 y lo agregas, pero si $x \in S_0 \implies \text{out}(x) = 0$ y ya habíamos visto $f(\vec{xy}) = 1$.

CONCLUIMOS que todo vértice T agrega **exactamente** a alguien en la cola. Y los elementos agregados son los de $S - S_0 \implies |T| = |S - S_0|$.

Todos los vértices de T tienen hijos solo agregados por alguien de X , y esto solo para y ; cada vértice de T es vecino de alguno de S . $T \subseteq \Gamma(S)$.

- $T = \Gamma(S)$?

Supongamos que no, entonces $\exists y \in \Gamma(S)$ tal que $y \notin T$.

- como $y \in \Gamma(S)$ $\exists x \in S$ con $xy \in E$, $y \in C$
- pero $y \notin T \Rightarrow y \notin C$, x nunca agrado a y .
- para $x \in S$ debe pasar que $f(x,y) = 1$.

Veremos que si un agrado a x , no puede ser agrado de T porque $f(x,y)$ implica que no existe $f(x,z) = 1$.

y tampoco lo agrado x porque eso significa que $x \in S_0$ y $\text{out}(x) > 0$ y es absurdio.

entonces $T = \Gamma(S)$.

y tenemos

$$|\Gamma(S)| = |T| = |S - S_0| = |S| - |S_0| \leq |S| \text{ por } S_0 \neq \emptyset.$$

probaremos que $|\Gamma(S)| < |S|$ lo que contradice nuestra hipótesis de que $|S| \leq |\Gamma(S)| \wedge S \subseteq X$.

Fin 

9) Teorema del matrimonio.

Todo grafo bipartido regular tiene Matching perfecto.

Prueba:

Sea $G = (X \cup Y, E)$ bipartito regular con $E \neq \emptyset \wedge \emptyset$
 $\wedge W \subseteq V(G)$ definir,

$$E_W = \{xy \in E(G) : x \in W \vee y \in W\}$$

$\hookrightarrow \{\text{lados con un extremo en } W\}$

Supongamos que $W \subseteq X$ (lo mismo para $W \subseteq Y$). Ademas como G es regular, $\Delta = \delta \geq 0 : d(z) = \Delta, \forall z$

$$\begin{aligned} |E_W| &= |\{xy \in E : x \in W\}| \\ &= \sum_{x \in W} |y : xy \in E| \\ &= \sum_{x \in W} d(x) \\ &\geq \Delta * |W| \end{aligned}$$

Ento vale para cualquier W subconjunto de $Y \cup X$.

entonces si todos los lados son de la forma XY , vale que $|E_x| = |E|$ y $|E| = |\Delta^* X|$, tambien para Y .

tambien,

$$|E_x| = |\Delta^* X|$$

$$|E_y| = |\Delta^* Y| \text{ pero } |E_x| = |E| = |E_y| \text{ para } G \text{ es bipartito.}$$

$$\begin{aligned} |E_x| &= |E_y| && \text{por lo que si encontramos} \\ |\Delta^* X| &= |\Delta^* Y| && \text{un matching completo} \\ |X| &\leq |Y| && \text{tambien es perfecto.} \end{aligned}$$

Vemos que se cumple la condición de Hall.

Consideremos $S \subseteq X$, tenemos que para todo lado $l \in E_S$ este lado es de la forma XY , donde $X \in S$, $Y \in \Gamma(S)$ y por lo tanto $l \in E_{\Gamma(S)}$. Vemos que $E_S \subseteq E_{\Gamma(S)}$

es decir:

$$E_S \subseteq E_{\Gamma(S)}$$

$$|E_S| \leq |E_{\Gamma(S)}|$$

$$\Delta|S| \leq \Delta|\Gamma(S)|$$

$$|S| \leq |\Gamma(S)|$$

Como se cumple el Teorema de cota de Hall entonces el matching es perfecto

Fin



10) Si G es bipartito entonces $\chi'(G) = \Delta$

Supongamos G regular. Y usaremos inducción en Δ

- S: $\Delta = 1$, G es una colección de lados distintos, se piden colores todos con el mismo color.

- Supongamos $\Delta(G) > 1$ y que el tco vale para todo grafo bipartito regular con grado max $\Delta(G)-1$

Alando el tco del matrimonio, tenemos un matching perfecto, sea E_1 los lados de ese matching.

Eliminamos E_1 de G y obtenemos G^* con $\Delta(G^*) = \Delta(G)-1$. Y por HI $\chi'(G^*) = \Delta(G^*) = \Delta(G)-1$.

Si a los lados de E_1 les damos un color distinto de esos $\Delta(G^*)$ colores, entonces tenemos colores para los lados de G con $\Delta(G^*)+1$ colores, lo cual prueba la inducción.

Caso general: queremos ver que todo grafo bipartito G puede incluirse en un grafo bipartito H regular tal que $\Delta(H) = \Delta(G)$, entonces

$\Delta(G) \leq \chi'(G) \leq \chi'(H) = \Delta(H) = \Delta(G)$, probaremos la existencia de H .

• Sea $G = (X \cup Y, E)$ y definimos X^+ , Y^+ como copiar de X, Y . Y tomamos

$$X^+ = X \cup Y^*, Y^+ = Y \cup X^*$$

$$\tilde{E} = E \cup \{x^*y^* : x \in E\} \cup \{xx^* : d(x) < \Delta(G)\} \cup \{yy^* : d(y) < \Delta(G)\}$$

Tenemos un grafo \tilde{G} que tiene como subgrafo a G pero también para cada vértice de G su grado es $< \Delta(G)$, su grado aumenta en 1. Y también es bipartito.

Concluimos que existe H regular con $\Delta(H) = \Delta(G)$ con $G \subseteq H$, por lo tanto podemos colorear G con $\Delta(G)$ colores.

11) Complejidad Hungaro. $\mathcal{O}(n^4)$

- Restar el minimo de cada fila y cada columna es $\mathcal{O}(m)$
- Calcular el matching inicial es \mathcal{O}_r o $\mathcal{O}(n^2)$

Mas que revisar n filas \rightarrow n columnas.

Como en nuestra implementacion no tenemos que cambiar ningun numero auxiliar al cambiar la matriz.

Y ademas como no perdemos ningun cero del matching perfecto y solo devolvamos los lugares donde hay nuevos ceros de esta forma continuamos con varios cambios de matriz en el medio pero solamente agregamos UN LADO al matching.

por lo que para conseguir un matching perfecto, haremos esto a lo sumo $\mathcal{O}(n)$ veces.

entonces **Complejidad Hungaro:**

$$\mathcal{O}(n^2) + \mathcal{O}(n) \cdot \begin{matrix} \text{extender} \\ \text{matching} \end{matrix}$$

para extender el matching en un lado debemos,

1- La complejidad de extender un lado (si lo hace) segun los de O mas BFS ($\mathcal{O}(m)$, $m=n^2 \Rightarrow \mathcal{O}(n^2)$) o si se usan matrices, en el peor de los casos se deben revisar todas las filas y sus vecinos tambien $\mathcal{O}(n^2)$.

2- Y para cambiar la matriz \Rightarrow Extender matching = $\mathcal{O}(n^2) + C.M.T$

C.M. seguirse

- Calcular m, calcular min elemento de $S \times \Gamma(S)$ o $\mathcal{O}(n)$
- Restar m de S y sumarlo a $\Gamma(S)$
Luego restar en cada fila en $\mathcal{O}(n) \Rightarrow$ restar todo la fila $\mathcal{O}(n) |S| = \mathcal{O}(n^2)$

L similarmente para restar es $\mathcal{O}(n^2)$

por lo tanto $C.M. = \mathcal{O}(n) + \mathcal{O}(n^2) = \mathcal{O}(n^2)$

T = Cuantos veces cambiar la matriz.

Sabemos que al correr el algoritmo obtener un nuevo matching o convergir un nuevo con $|S_n| > |\Gamma(S_n)| + |S_n| - |S|$

Lo probemos. Sea $x \in S$, $v \in \Gamma(x)$ tq $m = c_{x,v}$

con C matriz $\Rightarrow m = \min\{c_{x,v} : x \in S, v \in \Gamma(x)\}$.

Al restar m de C , la nueva matriz tiene un cero en $c_{x,v}$.
Como $x \in S$, al continuar el algoritmo, x se agregará a v a la cola.

Lv no está en la cola, por lo tanto $v \notin S$.

V puede formar o no parte del matching parcial.

LS: no forma parte entonces $f(v) = 0 \Rightarrow$ llegamos a + y tenemos un nuevo f-camino aumentante y extenderemos el matching.

LS: forma parte, entonces la columna v entra matcheada con alguna fila z .

- v es vecino de z y como $v \notin \Gamma(z)$, entonces z NO puede estar en S .
y ahí agregarán z al nuevo S (etiquetando o agregándolo a la cola).

Luego el algoritmo continua y si podemos extender el matching, bien. Si no terminaremos con un nuevo S que tendrá los mismos elementos mas AL MENOS z .

Así que $|S_{nuevo}| > |S_{actual}|$

Como S puede crecer a lo sumo $O(n)$ veces, entonces luego de a lo sumo $O(n)$ "recorridos" de S deberán si o si tener que poder extender el matching.

entonces $T \leq n \Rightarrow O(n)$

Complicación Hungaro:

$$\begin{aligned} O(n^2) + O(n) \cdot (CM \cdot T) &= O(n^2) + O(n) \cdot (O(n^2) O(n)) \\ &= O(n^2) + O(n) \cdot O(n^3) \\ &= O(n^4) \end{aligned}$$



Idea para reducirlo a $O(n^3)$, es que CM sea $O(n)$

definir un array con los mínimos de los elementos de una columna que están en las filas de S . Resumiendo CM tendrá 2 pasos:

1) Calcular mínimo, es $O(n)$ en un array. Y donde ento el $O(n)$ de Crearlos.

en el algoritmo cuando agregamos una fila x a S la recorremos buscando ceros, apreciamos ese momento para recorrer los elementos de fila x y columna v y decidir si es el mínimo de esa col o no y agregarlo. de esa forma escuchamos ese $O(n)$ en algo que demora $O(n^2)$.

2) Sumar y restar m a S y $\Gamma(x)$

la idea es nunca sumar o restar a la matriz, sino abstractear esas cuentas (sólo arrays RF[x] y SC[x] que nos dicen cuánto + / - los elementos de $F \setminus x$.

entonces en vez de cambiar todo la matriz, actualizaremos los arrays, que esto es $O(n)$.

El bucle entra en el momento de buscar ceros no preguntados
 $\text{if } (0 == C[x][v]) \text{ si no if } (0 == C[x][v] - RF[x] + SC[v])$
 tambien a la hora de agregar los minimos al arreglo utilizamos esa forma.

Haciendo esto se cambia CM para ser
 $O(n^2) + O(n^2) \geq O(n) + O(n)$ y

la complejidad de hungryo es $O(n^2) + O(n) \cdot (O(n), O(n)) = O(n^3)$

FIN 

12) Enunciado de cota de Hamming y probando.

Sea C un código de longitud n y corrige t errores

$$\#C \leq \frac{2^n}{\sum_{k=0}^t \binom{n}{k}} \quad \text{con } t = \left[\frac{s-1}{2} \right]$$

dcf:

Sea v una palabra $v \in \{0,1\}^n$ y un número natural $r \geq 0$, definimos el disco de radio r alrededor de v como.

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

luego C detecta t errores si $D_t(v) \cap C = \{v\} \quad \forall v \in C$, entonces

$$D_t(v) \cap D_t(w) = \emptyset \quad \forall v, w \in C : v \neq w.$$

por lo tanto si tenemos $A = \bigcup_{v \in C} D_t(v)$ A es una unión disjointa entonces $\#A = \sum_{v \in C} \#D_t(v)$

Calcularemos $\#D_t$

Sea $S_r(v) = \{w \in \{0,1\}^n : d_H(v,w) = r\}$, 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 esta unión es disjointa.

$$\#D_t(v) = \sum_{r=0}^t \#S_r(v) \quad \text{verificar } \#S_r(v) \text{ es una palabra que difiere de } v \text{ en exactamente } r \text{ bits. entonces}$$

$$\#S_r(v) = \binom{n}{r}$$

$$\Rightarrow \#D_t(v) = \sum_{r=0}^t \binom{n}{r} \quad \text{y } \#A = \sum_{v \in C} \left(\sum_{r=0}^t \binom{n}{r} \right) = \#A = \sum_{r=0}^t \binom{n}{r} \cdot \#C.$$

entonces $\#C = \frac{\#A}{\sum_{r=0}^t \binom{n}{r}}$ y como $\#A$ es un conj de palabras $\{0,1\}^n$ si 0 si era excedido por 2^n , entonces llegamos a:

$$\#C = \frac{\#A}{\sum_{r=0}^t \binom{n}{r}} \leq \frac{2^n}{\sum_{r=0}^t \binom{n}{r}} \quad \text{y queda demuestra el teorema.}$$

Fin 

13) Probemos que si H es matriz de chequeo de C , entonces

$$S = \min\{j : \exists \text{ const } j \text{ columna LD de } H\}$$

Sea $S = \min\{j : \exists \text{ const } j \text{ columna LD de } H\}$
probaremos $\delta(C) \leq S \Rightarrow S \leq \delta(C)$

$$\delta(C) \leq S$$

por def existe un const $LD = \{H^{(1)}, H^{(2)}, \dots, H^{(r)}\}$ entonces existen $c_1, \dots, c_r \in \mathbb{Q}$
tales numeros tales qve.

$$c_1 H^{(1)} + \dots + c_r H^{(r)} = 0, \text{ con } w = c_1 e_1 + \dots + c_r e_r,$$

$$w \neq 0.$$

Luego

$$\begin{aligned} H w^t &= H(c_1 e_1 + \dots + c_r e_r) \\ &= c_1 H e_1^t + \dots + c_r H e_r^t \\ &= c_1 H^{(1)} + \dots + c_r H^{(r)} \end{aligned}$$

$= 0$ entonces $w \in C$ pero $C = \text{Nu}(H)$
pero el peso de $w \leq r$, pero en la suma de \geq la suma se ej. $\Rightarrow w \neq 0$.

Luego tenemos

$$\delta(C) = \min\{|v| : v \in C, v \neq 0\} \leq |w| \leq S.$$

$$S \leq \delta(C)$$

Sea $v \in C$ tq $\delta(C) = |v| \Rightarrow \exists i_1, \dots, i_{\delta(C)}$ con $v = e_{i_1} + e_{i_2} + \dots + e_{i_{\delta(C)}}$

y como $v \in C \Rightarrow H v^t = 0 \Rightarrow$ con el mismo calculo de arriba concluimos
que.

$$H^{(i_1)} + H^{(i_2)} + \dots + H^{(i_{\delta(C)})} = H v^t = 0 \quad \{H^{(i_1)}, \dots, H^{(i_{\delta(C)})}\} \text{ que es LD}$$

por lo tanto $S = \min\{j : \exists \text{ const } j \text{ columna LD de } H\} \leq \delta(C)$

y concluimos qve

$$S \leq \delta(C) \wedge S \geq \delta(C) \Rightarrow S = \delta(C) \quad \text{y terminamos la prueba}$$

Fin. 

14) Sea C un código cíclico de dimensión k y longitud n . Sea $g(x)$ su polinomio generador probá que:

- C es formado por los múltiplos de $g(x)$ de gr $\leq n$, $C = \{p(x) : g(x) \mid p(x) \text{ y } \deg(p) \leq n\}$
- $C = \{v(x) \odot g(x) : v \text{ polin. cuadrática}\}$
- $\deg(g(x)) = n - k$
- $g(x) \mid 1 + x^n$

probar 1), 2)

Sean $C_1 = \{p(x) : \deg(p) \leq n \wedge g(x) \mid p(x)\} \supset C_2 = \{v(x) \odot g(x) : v \in \text{polin. cuadrática}\}$
por prop " $w \in C \Rightarrow w \otimes v \in C \Rightarrow C_2 \subseteq C$.

- Sea $p(x) \in C$, es dividir $p(x)$ por $g(x)$, obtenemos $p(x) = q(x)g(x) + r$ con $\deg(r) < \deg(g)$

Como $\deg(r(x)) < \deg(g(x)) < n \Rightarrow r(x) = r(x) \bmod (1+x^n)$

y como $p(x) \in C \Rightarrow \deg(p) < n \Rightarrow p(x) = p(x) \bmod (1+x^n)$, entonces:

$$\begin{aligned} r(x) &= r(x) \bmod (1+x^n) \\ &= (p(x) + q(x)g(x)) \bmod (1+x^n) \\ &= p(x) \bmod (1+x^n) + (q(x)g(x)) \bmod (1+x^n) \\ &= p(x) + q \otimes g \end{aligned}$$

- Como $p(x) \in C$, $q \otimes g \in C$ y C es lineal, concluimos que $r \in C$. pero $\deg(r) < \deg(g)$, pero g es el polin. de menor grado no nulo, $\Rightarrow r = 0$ por lo tanto $p(x) = q(x)g(x) \in C_1$.

Concluimos $C \subseteq C_1$

Veamos que $C \subseteq C_2$, si $\deg(p) < n \wedge p(x) = q(x)g(x)$ entonces:

$$p(x) = p(x) \bmod (1+x^n) \quad (\text{pues } \deg(p(x)) < n)$$

$$p(x) = q(x)g(x) \bmod (1+x^n) = q(x) \odot g(x) \in C_2, \text{ entonces } C_1 \subseteq C_2.$$

y como vimos que $C_2 \subseteq C \subseteq C_1 \subseteq C_2$, podemos probar las partes 1, 2.

3) Sea t el grado de $g(x)$, pues i) vimos que $p(x) \in C \Leftrightarrow q(x)g(x)$ para algún $q(x)$ pero como el gr de los elementos de C es $\leq n \Rightarrow \deg(q(x)g(x)) \leq n$, por lo tanto el grado de $g(x)$ es $< n-t$. Así para cada polinomio de grado $< n-t$, existe un polin en C , viceversa.

Entonces la cardinalidad de C es igual a la cardinalidad del conjunto de polinomios con $\deg < n-t$.

la cardinalidad de los polinomios que cumplen esto es 2^{n-t} , porque son polinomios que tienen $n-t$ coeficientes, y para cada coeficiente 2 opciones (0 o 1).

Tenemos que la cardinalidad de C es 2^{n-t} , pero como es lineal tiene que 2^t en totales tenemos $2^t = 2^{n-t} \Rightarrow t = n - t$ y ese es el grado de $g(x)$.

4) dividimos $1+x^n$ por $g(x)$ y obtenemos $1+x^n = q(x)g(x) + r(x)$ con $g(x) \mid g(x)$ $\Leftrightarrow g(x) \mid r(x)$

tenemos que $r(x) = 1+x^n + g(x)g(x) \Rightarrow$ como $g(x) \mid g(x)$ $\Rightarrow r(x) \equiv 1(x) \pmod{1+x^n}$

entonces:

$$r(x) = (1+x^n + g(x)g(x)) \pmod{1+x^n} = (1+x^n) \pmod{1+x^n} + (g(x)g(x)) \pmod{1+x^n}$$

$$= g(x)g(x) \in \mathbb{C} \Rightarrow r \in \mathbb{C}, \text{ pero como } g(x) \mid g(x) \text{ concluimos}$$

que $r(x) = 0 \Rightarrow$ entonces $1+x^n \equiv g(x)g(x) \Rightarrow g(x)$ divide a $1+x^n$.

Fin 

15) Demuestra que 3-SAT es NP-Completo.

basta con reducir polynomialmente SAT a 3-SAT ($SAT \leq_p 3-SAT$) o decir que dado una instancia de SAT $B = D_1 \wedge \dots \wedge D_n$ tenemos que construir una expresión booleana $\tilde{B} = E_1 \wedge \dots \wedge E_m$ donde E son expresiones 3-CNF.

para cada D_i construimos E_i . Sea $D_i = l_{i,1} \vee l_{i,2} \vee l_{i,k_i}$ (k_i : cantidad de literales de D_i)

Construcción: en E_i para cada D_i , con $y_{i,j}$ las variables que agregamos para \tilde{B}

Si $k_i = 1 \Rightarrow E_i = (l_{i,1} \vee y_{i,1} \vee \bar{y}_{i,2}) \wedge (\bar{l}_{i,1} \vee y_{i,1} \vee \bar{y}_{i,2}) \wedge (\bar{l}_{i,1} \vee \bar{y}_{i,1} \vee y_{i,2}) \wedge (\bar{l}_{i,1} \vee \bar{y}_{i,1} \vee \bar{y}_{i,2})$

Veamos que es equivalente a D_i (entonces si $B = 1 \Rightarrow D_i = 1 \Rightarrow E_i = 1 \wedge k_i = 1$)

$$\begin{aligned} * D_i = l_{i,1} & \text{ distributividad} = DV((y_{i,1} \vee y_{i,2}) \wedge (\bar{y}_{i,1} \vee \bar{y}_{i,2}) \wedge (\bar{y}_{i,1} \vee y_{i,2}) \wedge (\bar{y}_{i,1} \vee \bar{y}_{i,2})) \\ & \text{ asociatividad y} \\ & \text{ distributividad} = DV(y_{i,1} \vee (y_{i,2} \wedge \bar{y}_{i,2})) \wedge \bar{y}_{i,1} \vee (\bar{y}_{i,2} \wedge \bar{y}_{i,2}) \\ & \text{ no contradiccion y} \\ & \text{ neutro disyuncion} = DV(y_{i,1} \wedge \bar{y}_{i,1}) \\ & \text{idem} = D_i. \end{aligned}$$

Si $k_i = 2 \Rightarrow E_i = (l_{i,1} \vee l_{i,2} \vee y_{i,1}) \wedge (\bar{l}_{i,1} \vee \bar{l}_{i,2} \vee \bar{y}_{i,1})$

Veamos que es igual a D_i .

$$\text{distributividad} = (l_{i,1} \vee l_{i,2}) \vee (y_{i,1} \wedge \bar{y}_{i,1})$$

$$\text{no contradiccion y neutro disyuncion} = l_{i,1} \vee l_{i,2} = D_i.$$

Si $k_i = 3 \Rightarrow D_i = E_i$ y no agregó variables.

Si $k_i \geq 4 \Rightarrow$ agregó $k_i - 3$ variables extra \Rightarrow defino

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

d igual que antes queremos ver que son equivalentes. lo haremos viendo que

$$\exists \vec{I}: D_i(\vec{I}) = 1 \iff \exists \vec{I}, \vec{Y}: E_i(\vec{I}, \vec{Y}) = 1.$$

Veamos la vuelta \Leftarrow y asumimos que $D_i(\vec{I}) = 0 \Rightarrow \forall l_{i,j} = 0 \Rightarrow$ tenemos

$$\vec{I} = E_i = y_{i,1} \wedge (\bar{y}_{i,1} \vee y_{i,2}) \wedge \dots \wedge (\bar{y}_{i,k_i-4} \vee y_{i,k_i-3}) \wedge \bar{y}_{i,k_i-3}$$

esto me dice que $y_{i,1} = 1$, entonces $y_{i,2} = 1 \Rightarrow$ an sucesivamente obtengo $y_{i,3} = y_{i,4} = \dots = y_{i,k_i-3} = 1$, pero entonces $\bar{y}_{i,k_i-3} = 0 \Rightarrow$ eso significa que $E_i = 0$. Abusando que parte de suponer que $D_i(\vec{I}) = 0$.

y la vuelta queda probado.

ida \Rightarrow como $D_i(\vec{r}) = 1 \Rightarrow \exists j : l_{i,r} = 1$. Y generar una asignación

$$Y_{i,j} = \begin{cases} 1 & \text{Si } j \leq r-2 \\ 0 & \text{Si } j \geq r-1 \end{cases}$$

entonces tengo que

$$\begin{aligned} E_i(\vec{r}, \vec{s}) &= (l_{i,1} \vee l_{i,2} \vee \underbrace{Y_{i,1}}_{=1}) \wedge (l_{i,3} \vee \overline{Y_{i,1}} \vee \underbrace{Y_{i,2}}_{=1}) \wedge \dots (l_{i,r-1} \vee \overline{Y_{i,r-3}} \vee \underbrace{Y_{i,r-2}}_{=1}) \\ &\dots \wedge (\underbrace{l_{i,r}}_{=1} \vee \overline{Y_{i,r-2}} \vee Y_{i,r-1}) \wedge (l_{i,r+1} \vee \underbrace{\overline{Y_{i,r-1}}}_{=1} \vee Y_{i,r}) \\ &\dots \wedge (l_{i,k_1} \vee \underbrace{\overline{Y_{i,k_1}}}_{=1} \vee Y_{i,k_2}) \wedge (l_{i,k_2} \vee l_{i,k_3} \vee \underbrace{\overline{Y_{i,k_3}}}_{=1}) \\ &= 1. \end{aligned}$$

y queda demostrado que $\exists \vec{r} : D_i(\vec{r}) = 1 \Rightarrow \exists \vec{r}, \vec{s} : E_i(\vec{r}, \vec{s}) = 1$

y podemos ver que por cada disyunción D_i de B podemos dar un E_i equivalente, entonces si $E = D_1 \wedge \dots \wedge D_n \Rightarrow B = E_1 \wedge \dots \wedge E_n$ entonces $B \equiv B'$ y B' es una instancia de SAT en la que si la respuesta es "si" en 3-SAT también lo es en reverso.

la complejidad de la transformación dada en $O(nk)$ donde m es la cantidad de disyunciones y k la máxima cantidad de literales por disyunción.

16) Probar que 3-COLOR es NP-Completo, Reducir polynomialmente 3SAT.
3COLOR \leq_p 3SAT