Pruebas para el final de Discreta II

Emilio Pereyra

July 4, 2024

Nota: En general las notas estan (muy) verbosas, pero se pueden condensar mucho más. No es necesario en el final notar absolutamente todo lo que yo noto.

1 Flujos

1.1 Complejidad de Edmonds-Karp

1.1.1 Definiciones preeliminares

Notación:

1. Llamaremos N al network en cuestión.
2. n = Cantidad de vertices del Network.
3. m = Cantidad de lados de la network.
4. CA = 'Camino aumentante'

Primero debemos definir que son las distancias relativas al un f camino aumentante.

Definition 1.1 ($d_f(x,z)$) *Dados dos vertices x, z y un flujo f, definimos la distancia relativa a f entre x y z como la longitud del menor f-camino aumentante entre ambos vertices, si dicho camino existe. En caso de que no exista decimos $d_f(x,z) = \infty$, ó en caso de que $x = z$ decimos $d_f(x,z) = 0$.*

De la definición anterior definimos $d_k(x) = d_{f_k}(s,x)$ donde f_k es el k-ésimo flujo en una corrida de EK. Análogamente tambien definimos $b_k(x) = d_{f_k}(x,t)$.

Lo probaremos en la siguiente sección pero por ahora debemos asumir que $d_k(x) \leq d_{k+1}(x)$ i.e. las distancias en EK no disminuyen.

1.1.2 Demostración

Recordemos que el algoritmo construye en pasos 1,2,3...,z flujos intermedios, a partir de f_i caminos aumentantes para cada flujo $f_1, f_2, f_3, ...f_z$. Además hay que recordar para mas adelante en la prueba que las distancias en EK no disminuyen de un network auxiliar al siguiente. Para calcular la complejidad del algoritmo lo que haremos es decir:

$$\text{Complejidad de EK} \quad = (\text{Complejidad de hallar un CA}) \cdot (\text{Cantidad de CA posibles.}) \tag{1}$$

1. Es fácil ver que la complejidad de hallar un solo camino aumentante es $O(m)$, ya que a lo sumo es recorrer todos los lados del network.

2. Vamos a acotar la cantidad de caminos aumentantes posibles pero antes necesitamos una definición auxiliar que será fundamental para hacerlo.

Definition 1.2 (Lados críticos) *Un lado $\overrightarrow{xy}$ se vuelve crítico si este se vacía, ó si se satura i.e.:*

$$\begin{aligned} &f(\overrightarrow{xy}) = c(xy) \\ &\textit{ó bien} \\ &f(\overrightarrow{xy}) = 0 \end{aligned} \tag{2}$$

Notemos que en cada paso del algoritmo al menos un lado se vuelve crítico. ¿Cuantas veces se vuelve crítico un lado?

Supongamos que un lado $\overrightarrow{xy}$ se vuelve crítico en el paso k y luego otra vez en el paso r $(k < r)$.

I) Si $\overrightarrow{xy}$ se usa forward para construir el flujo f_k: Osea el CA aumentante es de la forma $s...\overrightarrow{xy}...t$, es obvio que:

$$d_k(x) + 1 = d_k(y). \tag{3}$$

Ahora para que el lado pueda ser crítico de nuevo en el paso r debe ser que:
1. El lado se satura nuevamente.
2. El lado se vacía.

Para que suceda (1) es necesario que el lado haya perdido algo de flujo antes de volver a saturarse. Para que suceda (2) puede suceder que r es el paso $k+1$ y vacía el lado completamente, o que antes de vaciarse el lado pierde flujo o se mantiene igual luego de algunos otros pasos.
En cualquier caso concluimos que existe un paso l tal que $k < l \leq r$, y el CA en el paso l es de la forma $s...\overrightarrow{yx}...t$. De esto último y sabiendo que los caminos en EK son de longitud minima podemos ver que:

$$d_l(x) = d_l(y) + 1. \tag{4}$$

Ahora miremos lo siguiente.

$$\begin{aligned} d_k(t) &= d_k(y) + b_k(y) \\ &= d_k(x) + 1 + b_k(y) \\ &\leq d_l(x) + 1 + b_l(y) \\ &\leq d_l(y) + 1 + 1 + b_l(y) \\ &\leq 2 + d_l(t) \end{aligned} \tag{5}$$

NOTA: La desigualdad se puede plantear porque las distancias no disminuyen.
$\therefore$ Descubrimos que para que un lado vuelva a ser crítico, toma al menos 2 pasos más, si el lado primero se usa forward.

II) Si $\overrightarrow{xy}$ se usa backwards para construir el flujo f_k: Osea que el CA es de la forma $s...\overrightarrow{yx}...t$ en el paso k. El análisis es casi identico al anterior.

$$d_k(y) + 1 = d_k(x). \tag{6}$$

Luego pasa 1 de dos cosas. 1. En el paso r el lado vuelve a vaciarse. 2. En el paso r el lado se satura.

En cualquier caso concluimos que existe l un paso intermedio (o el mismo r) de forma que: $k < l \leq r$. Y el CA en este paso es de la forma $s...\overrightarrow{xy}...t$ como

los caminos son de de distancia mínima concluimos:

$$d_l(x) + 1 = d_l(y). \tag{7}$$

Igual que antes veamos que:

$$\begin{aligned} d_k(t) &= d_k(x) + b_k(x) \\ &= d_k(y) + 1b_k(x) \\ &\leq d_l(y) + 1b_l(x) \\ &\leq d_l(x) + 1 + 1 + b_l(x) \\ &\leq 2 + d_l(t) \end{aligned} \tag{8}$$

$\therefore$ Descubrimos que para que un lado vuelva a ser crítico, toma al menos 2 pasos más, si el lado primero se usa backward.

Conclusiones: Como vimos, para que un lado que ya es crítico vuelva a serlo, ocurren al menos 2 pasos. Pero la distancia esta acotada, en el peor caso podemos tener que recorrer todos los n vertices del Network y esa es la distancia máxima (n-1 en realidad) ya que usamos BFS. Por lo que un lado es crítico $O(\frac{n}{2}) = O(n)$ veces. Por lo que si consideramos que hay m lados y pensamos que en el peor caso todos se vuelven críticos tantas veces como podrían tenemos una cota para la cantidad de CA posibles. Esto es $O(n) \cdot m = O(nm)$.

Finalmente volvemos al principio y ahora simplemente usamos la cota que estuvimos buscando:

$$\begin{aligned} \text{Complejidad de EK} &= (\text{Complejidad de hallar un CA}) \cdot (\text{Cantidad de CA posibles.}) \\ &= O(m) \cdot O(mn) \\ &= O(nm^2). \end{aligned} \tag{9}$$

2 Las distancias no disminuyen en EK

Anteriormente definimos $d_k(x) = d_{f_k}(s, x)$ donde f_k es el k-ésimo flujo, en este caso, en una corrida de EK. Ahora probaremos: $d_k(x) \geq d_{k+1}(x)\ \forall k$.
Demostración:
Supondremos lo contrario y llegaremos al absurdo. Entonces, supongamos que el conjunto A definido como:

$$A = \{y \in V\ :\ d_{k+1}(y) < d_k(y)\} \tag{10}$$

(V es el conjunto de vertices del Network) Y diremos que A tiene al menos un elemento.
Sea $x = min\{d_{k+1}(y) : y \in A\ \}$. Osea el vertice cuya distancia relativa es además la menor de entre todos los vertices de A. Notemos que $x \neq s$, ya que $s \notin A$ debido a que $d_k(s) = d_{k+1}(s) = 0\ \forall k$.

Podemos deducir que $d_{k+1} < \infty$, ya que $d_{k+1} < d_k \leq \infty$. Y entonces podemos afirmar que de hecho existe un f_{k+1} camino aumentante tal que 's...x'.

Ahora diremos que existe z, el vertice previo a x en el f_{k+1} camino aumentante 's...zx' que llamaremos p_{k+1}. Podemos decir que existe z ya que $s \neq x$ y además notemos que puede suceder que z sea s. Tendremos que hacer análisis sobre si es $\overleftarrow{zx}$, un lado backwards; ó es $\overrightarrow{zx}$, un lado forward, pero antes observemos los siguiente puntos.

Observación 1: $d_{k+1}(z) < d_{k+1}(x)$, obvio porque $d_{k+1}(z) + 1 = d_{k+1}(x)$. Basta con mirar el camino.

Observación 2: $z \notin A$ ya que si asi fuera, entonces x no es el de menor distancia en A. Por lo que $d_k(z) \leq d_{k+1}(z)$.

Observación 3: $d_k(z) < \infty$ y $d_{k+1}(z) < \infty$. Obvio porque $d_k(z) \leq d_{k+1}(z) < d_{k+1}(x) < \infty$.

Ahora si consideremos ambos casos $\overrightarrow{zx}, \overleftarrow{zx}$ por separado.

Si el camino es $p_{k+1} : s...\overrightarrow{zx}$:

Entonces sabiendo que $d_k(z) < \infty$(Observación 3), podemos afirmar que existe un f_k camino aumentante de s a z, que además como se construye con BFS es el de menor distancia, llamaremos a este camino $p_k : s...z$. Ahora como x y z son vecinos podemos añadir x al final de p_k resultando en: 's...zx', y pensar que este es un camino aumentante válido. Pero si hacemos eso notariamos:

$$\begin{aligned} d_k(x) \leq d_k(z) + 1 \leq d_{k+1}(z) + 1 \leq d_{k+1}(x) &\implies d_k(x) \leq d_{k+1}(x). \\ &\implies x \notin A. \text{ Absurdo.} \end{aligned} \tag{11}$$

Este absurdo viene de suponer que es válido poner x al final de p_k. La única razón que no nos permitiría poner x al final de p_k es que el lado zx fue saturado en un paso anterior. Otra cosa que tenemos que notar es que zx si usa en el p_{k+1} y de forma forward. Pero entonces el lado se vacia al menos un poco antes del paso $k+1$, por lo que la única opción es que en el paso k se use backwards, osea que el camino aumentante construido en el paso k es: $s...\overleftarrow{xz}...t$ de forma que se des-satura el lado para el paso siguiente.

A causa de esto $d_k(x) + 1 = d_k(z)$, y planteamos:

$$\begin{aligned} d_k(t) &= d_k(z) + b_k(z) \\ &= d_k(x) + 1 + b_k(z) \\ &> d_{k+1}(x) + 1 + b_k(z) \\ &= d_{k+1}(z) + 1 + 1 + b_k(z) \\ &\geq d_k(z) + 2 + b_k(z) \\ &= d_k(t) + 2. \\ \implies d_k(t) &> d_k(t) + 2. \end{aligned} \tag{12}$$

Absurdo.
Si el camino es $p_{k+1} : s...\overleftarrow{zx}$:

Es analogo al caso anterior, y de nuevo podemos hablar de $p_k : s...z$, al cual intentaremos agregar x al final. Resultando 's...zx' como un posible camino aumentante válido, pero notariamos:

$$\begin{aligned} d_k(x) \leq d_k(z) + 1 \leq d_{k+1}(z) + 1 \leq d_{k+1}(x) &\implies d_k(x) \leq d_{k+1}(x). \\ &\implies x \notin A. \text{ Absurdo.} \end{aligned} \tag{13}$$

Y en este caso concluimos que agregar x a final de p_k no es posible, con única causa posible que el lado este vacío por un paso anterior. Pero cómo en el paso $k+1$ si se puede usar, debemos llenarlo al menos un poco en este caso. Por lo que el f_k camino aumentante que se construye en el paso k es $s...\overrightarrow{xz}...t$ para que haya algo de flujo en el el paso siguiente donde se devuelve. Así que $d_k(x) = d_k(z) + 1$. Planteamos:

$$\begin{aligned} d_k(t) &= d_k(z) + b_k(z) \\ &= d_k(x) + 1 + b_k(z) \\ &> d_{k+1}(x) + 1 + b_k(z) \\ &= d_{k+1}(z) + 1 + 1 + b_k(z) \\ &\geq d_k(z) + 2 + b_k(z) \\ &= d_k(t) + 2. \\ \implies d_k(t) &> d_k(t) + 2. \end{aligned} \tag{14}$$

Llegando nuevamente al absurdo, así que en ningún caso valen nuestras suposiciones iniciales, asi que A es de hecho un conjunto vacío, y determinando que la propiedad no se cumple para ningún vertice, y que es falsa para todo k.
$\implies \quad d_k(x) \leq d_{k+1}(x) \ \forall k.$

Fin.

3 Complejidad de Dinic original

TODO Fin.

4 Complejidad de Dinic-even (versión occidental)

TODO Fin.

5 Maxflow - Mincut theorem

En esta sección asumimos que vale la propiedad que dice que dado f un flujo sobre un network N, y S un corte sobre el mismo network, entonces:

$$v(f) = f(S, \overline{S}) - f(\overline{S}, S). \tag{15}$$

Nota: $\overline{S} = V - S$.
El teorema enuncia que se cumplen tres cosas:

1. $v(f) \leq CAP(S)$.

2. Si $v(f) = CAP(S)$, entonces f es maximal, y S es minimal.

3. f es un flujo maximal $\iff$ existe S un corte minimal.
   Demostración de 1:

   Recordemos que
$$v(f) = f(S, \overline{S}) - f(\overline{S}, S) \tag{16}$$

Cómo el valor de flujo debe ser ≥ 0, es obvio que $f(S, \overline{S}) \geq f(\overline{S}, S)$.

$$v(f) = f(S, \overline{S}) - f(\overline{S}, S) \leq f(S, \overline{S}) = \sum_{\substack{y \in \overline{S} \\ x \in S \\ xy \in E}} f(\overrightarrow{xy}) \tag{17}$$

Recordemos que la función de flujo no puede superar la capacidad del lado asi que:

$$\begin{aligned} v(f) = f(S, \overline{S}) - f(\overline{S}, S) \leq f(S, \overline{S}) &= \sum_{\substack{y \in \overline{S} \\ x \in S \\ xy \in E}} f(\overrightarrow{xy}) \\ &\leq \sum_{\substack{y \in \overline{S} \\ x \in S \\ xy \in E}} c(\overrightarrow{xy}) \\ &= CAP(S). \end{aligned} \tag{18}$$

En sintesis:

$$v(f) \leq CAP(S). \tag{19}$$

Demostración de 2:

Sea g un flujo en el network.

$$v(g) \leq CAP(S) = v(f) \tag{20}$$

Ahora sea T un corte en el network.

$$CAP(S) = V(f) \leq CAP(T) \tag{21}$$

Demostración de 3:
Propondremos $S = \{x \in V : \text{Existe un } f\text{-camino aumentante hasta } x\} \cup \{s\}$.

Veamos que es un corte, osea que $t \notin S$. Supongamos por el absurdo que $t \in S$. Entonces existe un camino aumentante que permite definir un nuevo flujo $f*$, tal que auementa el valor de flujo en ε. Osea

$$v(f) + \varepsilon = v(f*). \tag{22}$$

Absurdo porque f es maximal. $\implies t \notin S$, S es corte.

Y ahora recordemos la propiedad

$$v(f) = f(S, \overline{S}) - f(\overline{S}, S). \tag{23}$$

Veamos ambos terminos de la resta por separado.

1.
$$f(S, \overline{S}) = \sum_{\substack{y \in \overline{S} \\ x \in S \\ xy \in E}} f(\overrightarrow{xy}). \tag{24}$$

Como sabemos que $y \notin S$, sabemos que el lado $\overrightarrow{xy}$ esta saturado i.e. $f(\overrightarrow{xy}) = c(\overrightarrow{xy})$, esto para todos los terminos de la suma por lo que:

$$f(S, \overline{S}) = \sum_{\substack{y \in \overline{S} \\ x \in S \\ xy \in E}} f(\overrightarrow{xy}) = \sum_{\substack{y \in \overline{S} \\ x \in S \\ xy \in E}} c(\overrightarrow{xy}) = CAP(S). \tag{25}$$

2.
$$f(\overline{S}, S) = \sum_{\substack{w \in \overline{S} \\ z \in S \\ wz \in E}} f(\overrightarrow{wz}) \tag{26}$$

a) Por un lado sabemos que no existe un f-camino aumentante hasta un w.
b) Pero existe uno para hasta cada uno de los z.

Ahora, por *b*) sabemos que existe el camino '$sz_1z_2...z$' y si agregaramos w al final del camino, tal que '$sz_1z_2...\overleftarrow{zw}$', sabemos por *a*) que este no es un f camino aumentante válido, osea que no se puede usar $\overleftarrow{zw}$ backwards y la razón debe ser que $f(\overrightarrow{zw}) = 0$ (No puede devolver flujo).

Esto vale para todos los términos de la suma por lo que:

$$f(\overline{S}, S) = \sum_{\substack{w \in \overline{S} \\ z \in S \\ wz \in E}} f(\overrightarrow{wz}) = \sum_{\substack{w \in \overline{S} \\ z \in S \\ wz \in E}} 0 = 0. \tag{27}$$

Volviendo a la expresión original:

$$\begin{aligned} v(f) &= f(S,\overline{S}) - f(\overline{S},S) \\ &= CAP(S) - 0 \\ &= CAP(S). \end{aligned} \tag{28}$$

Fin.

6 P-NP

6.1 3SAT es NP completo.

Tenemos que probar que SAT se reduce polinomialmente a 3-SAT ($SAT \leq_p 3-SAT$).

Sea B una instancia de SAT, una conjunción de disyunciones con varibles $x_1, x_2, x_3, ..., x_n$, donde llamamos a cada disyunción D_j. Osea que: $B = D_1 \wedge D_2 \wedge ... \wedge D_m$.

Estructura de la prueba:
En primer lugar vamos definir A, un algoritmo de orden polinomial que transforma instancias B de SAT, en instancias E de 3-SAT.
Lo siguiente es demostrar que 'B satisfacible $\iff$ E satisfacible' i.e.:
Dado $\vec{b}$ un vector de bits que representa la asignación de n variables $x_1, x_2, ..., x_n$ tal que $\vec{b} = (b_1, b_2, b_3, ..., b_n)$. Entonces se cumple que:

$$B(\vec{b}\,) = 1 \iff \exists \vec{d}\ : E(\vec{b}\,, \vec{d}\,) = 1. \tag{29}$$

Lo que se quiere decir con esto es: 'B es satisfacible con la asignación $\vec{b}$ si y solo si la misma asignación satisface E, que además poseerá variables extra que asignaremos con $\vec{d}$.

Definiendo A:

A será una transformación $A : B \to E$, osea que dada nuestra instancia con disyunciones $D_j = l_{j,1} \vee l_{j,2} \vee ... \vee l_{j,r_j}$ (Donde l_{j,r_j} es el último termino de una disyuncion D_j) con una cantidad arbitraria de terminos, que pase a ser una con exactamente 3 terminos. Lo haremos caso por caso para los D_j posibles.

Para que quede mas claro, veremos 'uno por uno' los D_j de B y si es de una forma u otra determina como es al final la transformación.

1. Si $D_j = (l_{j,1} \vee l_{j,2} \vee l_{j,3})$, i.e disyunción de tres literales: Es inmediato, no hay que hacer nada.

$$D_j \to E_j = D_j \tag{30}$$

2. Si $D_j = (l_{j,1} \vee l_{j,2})$, i.e disyunción de dos literales:
La transformación es agregar una variable y_j, que sea intrascendente:

$$D_j \to E_j = (l_{j,1} \vee l_{j,2} \vee y_j) \wedge (l_{j,1} \vee l_{j,2} \vee \overline{y_j}) \tag{31}$$

3. Si $D_j = (l_{j,1})$, i.e disyunción de un literal:
Similarmente a 2. agregamos dos variables intrascendentes:

$$D_j \to E_j = (l_{j,1} \vee y_{j,1} \vee y_{j.2}) \wedge (l_{j,1} \vee \overline{y_{j,1}} \vee \overline{y_{j.2}}) \tag{32}$$

De esta forma solo depende del literal original.

Con cuatro aun no podremos dar una forma general, pero a partir de 5 términos es totalmente general.

4. Si $D_j = (l_{j,1} \vee l_{j,2} \vee l_{j,3} \vee l_{j,4})$, i.e disyunción de cuatro literales:

$$D_j \to E_j = (l_{j,1} \vee l_{j,2} \vee y_j) \wedge (l_{j,3} \vee l_{j,4} \vee \overline{y_j}) \tag{33}$$

5. Si $D_j = (l_{j,1} \vee l_{j,2} \vee ... \vee l_{j,r_j})$ con $r_j \geq 5$. La idea para transformar estos casos es agregar variables intrascendentes 'en forma de cadena'.

$$\begin{aligned} D_j \to E_j =& (l_{j,1} \vee l_{j,2} \vee y_{j,1}) \wedge (\overline{y_{j,1}} \vee l_{j,3} \vee y_{j,2}) \wedge (\overline{y_{j,2}} \vee l_{j,4} \vee y_{j,3}) \wedge ... \\ & ... \wedge (\overline{y_{j,r_j-4}} \vee l_{j,r_j-2} \vee y_{j,r_j-3}) \wedge (\overline{y_{j,r_j-3}} \vee l_{j,r_j-1} \vee l_{j,r_j}). \end{aligned} \tag{34}$$

El algoritmo A es obviamente polinomial, ya que simplemente recorriendo la instancia B y mirando las disyunciones podemos ir escribiendo la instancia 3-SAT que llamamos $E = E_1 \wedge E_2 \wedge ... \wedge E_s$.

Ahora queremos ver que una es satisfacible si y solo si la otra lo es. Recordemos que en la transformación aparecen variables y que también deben ser asignadas, pero por la forma en la que las elegimos resultara que solo depende de que exista$\vec{b}$.

$$B \text{ satisfacible } \iff E \text{ satisfacible.}$$

i.e dado $\vec{b}$ un vector de bits tal que:

$$B(\vec{b}\,) = 1 \iff \exists \vec{d}\ : E(\vec{b}\,, \vec{d}\,) = 1 \tag{35}$$

Nota: Vamos a probar solamente para los casos dificiles, donde las D_j son las del caso 5, las otras al ser casos particulares son triviales.

1. $B(\vec{b}\,) = 1 \implies$

Primero recordemos que B es de la forma SAT, conjunciones de disyunciones, por lo que para que B sea satisfacible en todos los D_j existe al menos un literal

$l_{j,k}$ tal que el literal es uno. En base a esto vamos a proponer $\vec{d}$ la asignación para las variables agregadas y como:
Para cada disyunción D_j a la variable $y_{j,i}$ le hacemos la siguiente asignacion

$$d_{j,i} = \begin{cases} 1 & \text{Si} \quad i \leq k-2 \\ 0 & \text{En caso contrario.} \end{cases} \tag{36}$$

i.e: $1 = d_{j,1} = d_{j,2} = ... = d_{j,k-2}$; y $0 = d_{j,k-1} = d_{j,k} = ... = d_{j,r_j}$.

De esta forma sucede que en la disyunción donde aparece el literal que **sabemos** es asignado con 1, es el único uno que aparece, pues $\overline{y_{j,k-2}}$ sera 0, y $y_{j,k-1}$ también. Entonces en todas las disyunciones anteriores sabemos que aparece el uno de todos las $k-2$ y's que asignamos con 1. En todas las disyunciones posteriores aparece el uno porque ahora las negadas propagan el 1 hasta el final. Es mas fácil ver esto escrito:

$$\begin{aligned} E_j(\vec{b}, \vec{d}\,) = {} & (l_{j,1} \vee l_{j,2} \vee y_{j,1}) \wedge (\overline{y_{j,1}} \vee l_{j,3} \vee y_{j,2}) \wedge \dots \\ & \dots \wedge (\overline{y_{j,k-2}} \vee l_{j,k} \vee y_{j,k-1}) \wedge \overline{y_{j,1}} \vee l_{j,3} \vee y_{j,2}) \wedge \dots \\ & \dots \wedge (\overline{y_{j,r_j-3}} \vee l_{j,r_j-1} \vee l_{j,r_j}) \\ \\ = {} & (l_{j,1} \vee l_{j,2} \vee 1) \wedge (0 \vee l_{j,3} \vee 1) \wedge \dots \\ & \dots \wedge (0 \vee l_{j,k} \vee 0) \wedge (1 \vee l_{j,3} \vee 0) \wedge \dots \\ & \dots \wedge (1 \vee l_{j,r_j-1} \vee l_{j,r_j}) \end{aligned} \tag{37}$$

Entonces es obvio que como aparece un uno en cada una de las conjunciones, la expresión se satiface.

$2. \exists \vec{d}\ : E(\vec{b}\,, \vec{d}\,) = 1 \implies$

Para este caso vamos a suponer por el absurdo que $\vec{b}$ no satisface a B, osea que $B(\vec{b}\,) \neq 1$. Para que B no sea satisfacible es necesario que alguna de las disyunciones D_j tenga literales todos 0. Veamos entonces lo que pasa en 'la versión transformada' de D_j. Si D_j fue asignada insatisfacible:

$$\begin{aligned} D_j &= (l_{j,1} \vee l_{j,2} \vee \dots \vee l_{j,r_j}), \\ D_j(\vec{b}\,) &= (0 \vee 0 \vee \dots \vee 0) = 0. \end{aligned} \tag{38}$$

En la versión transformada:

$$\begin{aligned} D_j \to E_j &= (l_{j,1} \vee l_{j,2} \vee y_{j,1}) \wedge (\overline{y_{j,1}} \vee l_{j,3} \vee y_{j,2}) \wedge (\overline{y_{j,2}} \vee l_{j,4} \vee y_{j,3}) \wedge ... \\ &... \wedge (\overline{y_{j,r_j-4}} \vee l_{j,r_j-2} \vee y_{j,r_j-3}) \wedge (\overline{y_{j,r_j-3}} \vee l_{j,r_j-1} \vee l_{j,r_j}). \\ &= (0 \vee 0 \vee y_{j,1}) \wedge (\overline{y_{j,1}} \vee 0 \vee y_{j,2}) \wedge (\overline{y_{j,2}} \vee 0 \vee y_{j,3}) \wedge ... \\ &... \wedge (\overline{y_{j,r_j-4}} \vee 0 \vee y_{j,r_j-3}) \wedge (\overline{y_{j,r_j-3}} \vee 0 \vee 0). \\ &= (y_{j,1}) \wedge (\overline{y_{j,1}} \vee y_{j,2}) \wedge (\overline{y_{j,2}} \vee y_{j,3}) \wedge ... \\ &... \wedge (\overline{y_{j,r_j-4}} \vee y_{j,r_j-3}) \wedge (\overline{y_{j,r_j-3}}). \end{aligned} \tag{39}$$

Ahora recordemos la caracterización del implica ($\neg p \vee q = p \to q$) y utilicemos la propiedad reemplazando enla expresión.

$$= y_{j,1} \wedge (y_{j,1} \to y_{j,2}) \wedge (y_{j,2} \to y_{j,3}) \wedge ..... \wedge (y_{j,r_j-4} \to y_{j,r_j-3}) \wedge (\overline{y_{j,r_j-3}}). \tag{40}$$

Es simple ver entonces que evaluando primero que $y_{j,1}$ se asigna necesariamente 1, y además y_{j,r_j-3} se asigna necesariamente en 0. Entonces de la primera implicación (de izquierda a derecha) solo puede resultar verdadera en caso de que $y_{j,2}$ también sea asignada 1. En la siguiente es necesario que $y_{j,3}$ sea verdadera. Asi sucesivamente hasta que es necesario que y_{j,r_j-3} sea verdadera, cosa que no permite que se satisfaga la expresión. Por lo que concluimos que E no puede ser satisfecha con ninguna asignación posible de los y. Pero habiamos asumido que existe alguna asignación $\vec{d}$ que satisfacia a E si asignabamoss el resto de variables con $\vec{b}$. Esto es un **absurdo** que vino de suponer que hay un D_j no satisfacible. Pero entonces todos los D_j son satisfacibles por lo que la expresión B debe ser satisfacible. Fin.

7 3-color es NP completo