# P vs NP 2da parte

Daniel Penazzi

9 de mayo de 2021

PNP2

Daniel Penazzi



# Tabla de Contenidos

# *NP*-completo



- La clase pasada dimos la definición y ejemplos de problemas en *NP* (y co-*NP*).
- En esta clase introduciremos las nociones de *NP*-hard y *NP*-completo, y similar para co-*NP*.
- Estos saldrán de una herramienta que se usa para "comparar" problemas de decisión.
- Es decir, dados dos problemas de decisión queremos poder decir cual de los dos es "mas facil" de resolver.

# Reducción Polinomial



- Obviamente si supieramos cuál es la mejor complejidad de algoritmos que resuelven cada uno de ellos, ya estaría, pero como vimos, no se conoce la complejidad de muchos problemas en *NP*, ni siquiera si son o no polinomiales.
- El concepto clave para comparar problemas es el de "Reducción Polinomial"
- La idea básica es definir "Reducción Polinomial" de forma tal que si "reducimos polinomialmente" un problema de decisión $\tau$ a un problema de decisión $\rho$, entonces si podemos resolver $\rho$, podamos resolver $\tau$ con sólo un esfuerzo extra polinomial.

# Reducción Polinomial



- De esa forma si podemos "reducir polinomialmente" un problema de decisión $\tau$ a un problema de decisión $\rho$, entonces $\tau$ es "mas fácil" de resolver que $\rho$, porque si pudieramos resolver $\rho$, entonces automáticamente podriamos resolver $\tau$ usando la forma de resolver $\rho$ mas un esfuerzo extra polinomial para "transformar" $\tau$ en $\rho$.
- Ahora bien, hay varios conceptos distintos de "Reducción Polinomial" en la literatura.

# Reducción Polinomial de Cook y de Karp



- Las dos mas conocidas son:
  1. La reducción polinomial de Cook (tambien llamada reducción de Cook-Turing o a veces reducción polinomial de Turing)
  2. La reducción polinomial de Karp. (el de Edmonds-Karp)
- Si bien la primera fue la de Cook, la que casi todo el mundo usa es la de Karp.

# Razones para usar la de Karp



- Hay varios motivos para eso.
- Si un problema se reduce a otro por medio de una reducción de Karp, entonces tambien se reduce por medio de una reducción de Cook.
- Al revés no se sabe si siempre es cierto. (se sospecha que no, pero no se sabe. De hecho si fuera cierto entonce *NP* $=$co-*NP*).
- Ambas nociones dan origen a distintos conjuntos *NP*-hard y co-*NP*-hard, y en realidad bajo reducción de Karp esos dos conjuntos son distintos pero bajo Cook son iguales.

# Razones para usar la de Karp



- Como la reducción de Karp es mas “granular” y permite diferenciar mas clases de complejidad, es la que mas se usa.
- Aún si les gusta mas la de Cook, si les enseñara todo usando reducciones de Cook despues no entenderían mucha parte de la literatura de esta area, que usa Karp.
- Asi que daremos una idea de ambas, pero luego daremos la definición formal y todas las pruebas usando la de Karp.

# Reducción Polinomial de Cook



- En ambas, como dije, la idea es que si reducimos polinomialmente $\tau$ a $\rho$, deberiamos poder resolver $\tau$ con sólo esfuerzo polinomial extra sabiendo resolver $\rho$.
- Cook (basandose en una noción similar de Turing, pero donde no requeria la parte "polinomial" ) define que $\tau$ se reduce polinomialmente a $\rho$ si:
  - dado un oraculo que resuelve instancias de $\rho$ en tiempo $O(1)$, podemos construir un programa que tome una instancia de $\tau$ y haciendo una cantidad polinomial de llamadas al oraculo de $\rho$, resolver $\tau$ en tiempo polinomial.

# Reducción Polinomial de Cook vs Karp



- Entonces en la reducción de Cook, podemos resolver muchas veces (mientras sea una cantidad polinomial de veces) distintas instancias de $\rho$ para resolver una instancia de $\tau$.
- Tambien tiene el “poder” de que aunque se llame al oraculo una sola vez, luego se pueden hacer “cosas” con el resultado del oraculo.
- Karp restringe mucho mas lo que se puede hacer.
- Sólo permite una llamada al oraculo y ademas hay que usar directamente el resultado.

# Reducción Polinomial de Karp



## Reducción Polinomial (de Karp)

Dados dos problemas de decisión $\rho$ y $\tau$, se dice que $\tau$ es reducible polinomialmente a $\rho$ ($\tau \leq_P \rho$) si existe un algoritmo $A$ polinomial que dadas instancias $x$ de $\tau$, produce instancias $A(x)$ de $\rho$, de forma tal que para toda instancia $x$ de $\tau$ se cumple que $\tau(x) = \rho(A(x))$.

# Reducción Polinomial de Karp



- Como estamos hablando de problemas de decisión las respuestas posibles son sólo SI o NO, asi que la forma general de los teoremas que prueban reducción de Karp es:
  1. Se da el algoritmo $A$ y se muestra que es polinomial.
  2. Se prueba que $\tau(x) = \text{“SI”} \Rightarrow \rho(A(x)) = \text{“SI”}$
  3. Se prueba que $\rho(A(x)) = \text{“SI”} \Rightarrow \tau(x) = \text{“SI”}$
- (o bien sus equivalentes lógicos, pej [3] es lo mismo que probar $\tau(x) = \text{“NO”} \Rightarrow \rho(A(x)) = \text{“NO”}$

# Reducción Polinomial de Karp



- Muchos alumnos cometen serios errores lógicos, pej, intentar probar [2] sin dar antes el algoritmo $A$.
- O peor, intentan probar [$\rho(A(x)) =$ "SI" $\Rightarrow \tau(x) =$ "SI"] usando en el medio de la prueba....que $\tau(x) =$ "SI".
- Esto suele pasar porque memorizan sin entender las partes [2] y [3] y en el medio de la prueba de pej [3] ponen un fragmento de la prueba de [2].
- Veremos algunas pruebas de reducción y en todo examen final se requerirá una de estas pruebas.
- Por mas que tengan el 90% de la prueba "bien" si cometene un error lógico grave como los indicados arriba el descuento de puntos será masivo.

# Completitud NP



## Definición

1. Un problema de decisión $\rho$ se dice *NP*-hard si $\tau \in \text{NP}$ implica $\tau \leq_P \rho$.
2. $\rho$ se dice *NP*-completo si es NP-hard y está en *NP*.

- Es decir, los problemas *NP*-hard son problemas que son “mas dificiles" que cualquier problema en *NP*.
- No es extraño pensar que existan problemas asi, pero para el caso de *NP* completo, pedimos que ademas el problema tambien este en *NP*.

# Completitud NP



- En principio podria suceder que no exista ningún problema *NP* completo sino que tuvieramos dentro de *NP* problemas cada vez mas dificiles, o problemas incomparables entre si, pero de hecho se conocen miles de problemas *NP* completos.
- Son todos “igualmente" dificiles entre si, y todos ellos son los mas dificiles de los problemas NP.

# Completitud NP



- Para probar $P = NP$ no hace falta dar un algoritmo polinomial PARA CADA problema NP.
- Basta darlo para UNO cualquiera de los problemas *NP* completos.
- Pues si $\rho$ es NP completo y se prueba que $\rho$ está en $P$, entonces dado cualquier otro problema $\tau \in$NP, podemos probar que $\tau \in$P reduciendolo a $\pi$, como se explicó antes.

# Completitud NP



- Por otro lado, si se quiere probar $P \neq NP$, entonces en vez de buscar algún problema cualquiera, lo lógico es buscar uno de entre la lista de problemas *NP* completos, ya que siendo los mas dificiles, si algun problema *NP* no está en *P*, ellos tampoco lo estarán.
- Veamos cual fue el primer problema que se demostró que era *NP* completo.

# Cook



**Teorema (de Cook, 1971)**

(CNF)SAT es *NP*-completo.

- Cook probó este teorema usando su reducción y luego Karp probó que tambien valia para la suya.
- El ruso Levin probó en forma independiente un teorema similar pero no era sobre problemas de decisión, asi que a veces es llamado el teorema de Cook-Levin.

# Demostración del teorema de Cook



- Prueba: no la daremos.
- En gral requiere estar familiarizado con maquinas de Turing, aunque sospecho que debe haber alguna prueba mas informal dando vuelta.
- Esencialmente hay que ver que se puede codificar cualquier máquina de Turing no determinística como una expresión booleana en CNF.
- Algunas veces la han dado en 4to año, pero por lo que me comentaron la prueba les ha llevado de 2 a 4 clases, y entiendo que no la dan mas.

# Karp



- Karp probó no sólo que el teorema de Cook valia para su noción de reducción polinomial, sino que probó que otros 21 problemas eran *NP* completos. (según SU noción de reducibilidad, y por lo tanto, tambien para la de Cook).
- Para probar esto lo que usaba Karp era reducción DE SAT(o de algún otro de los problemas que ya habian sido probados NP completos) a ese problema.
- Esto es asi pues si $\sigma$ es *NP* completo y $\sigma \leq_P \rho$ entonces $\rho$ también es *NP* completo, pues dado $\tau \in NP$ tenemos:
  - $\tau \leq_P \sigma \leq_P \rho$
  - y es fácil ver que $\leq_P$ es transitiva
  - asi que $\tau \leq_P \rho$.

# Karp



- Pej, Karp tomó un problema $\rho_1$ y probó que SAT$\leq_P \rho_1$
- y luego vio que $\rho_1 \leq_P \rho_2$ y luego que $\rho_2 \leq_P \rho_3$,etc,
- lo cual prueba que $\rho_1, \rho_2, \rho_3$, etc, son todos *NP*-completos.
- A la lista inicial de Karp se le han agregado miles de problemas, de todo tipo, y pocas personas creen que pueda ser que todos ellos sean polinomiales, asi que esta es otra razón por la que se cree que $P \neq NP$ aunque nadie lo ha podido probar.

# Karp



- Muchos de esos problemas tienen que ver con problemas sobre grafos, otros sobre otras partes de la matemática, varios sobre areas de programación, incluso problemas de biologia computacional como algunos casos de protein folding.
- Algunos juegos, pej, Sudoku, Buscaminas, Tetris y MagicTheGathering son *NP*-completos.
- (hay que buscar la definición técnica de qué significa que esos juegos sean NP completos, es decir, cual es el problema de decisión asociados a cada uno de esos juegos)
- Veamos algunos de los 21 problemas originales de Karp.

# 3SAT



## Definición

El problema 3SAT es como CNF-SAT excepto que se requiere ademas que cada disjunción sea disjunción de exactamente tres literales.

## Teorema (Karp)

3SAT es NP completo.

Prueba: La idea, como dije, es probar que (CNF-)SAT$\leq_P 3SAT$

# 3SAT

- Sea $B = D_1 \wedge ... \wedge D_m$ una booleana instancia de (CNF)SAT, es decir una expresión booleana en CNF, con variables $x_1, ..., x_n$ y cada $D_j$ una disjunción de literales.
- Tenemos que construir polinomialmente una $\tilde{B}$ que sea instancia de 3SAT y tal que $B$ sea satisfacible si y solo si $\tilde{B}$ lo es.
- Lo de polinomial es clave, porque si no podriamos simplemente definir $\tilde{B}$ como 1 si $B$ es satisfacible y 0 si no,
- pero para HACER la construcción, deberiamos primero resolver si $B$ es o no satisfacible, y eso no se sabe como hacerlo en tiempo polinomial.

# 3SAT



- La idea de Karp es construir $A(B) = \tilde{B} = E_1 \wedge \ldots \wedge E_m$ donde cada $E_j$ se construye a partir de cada $D_j$ y es tal que es conjunción de disjunciones de 3 literales.
- Si $D_j$ tiene exactamente 3 literales, se toma $E_j = D_j$.
- Si tiene menos de 3 literales, se agregan variables extras que basicamente no hacen nada mas que "rellenar" espacio para completar los 3 literales.
- Para distinguir las variables nuevas de las originales $x_i$, usaremos las letras "$y$".

# 3SAT

- Si $D_j$ tiene dos literales: $D_j = \ell_{1,j} \vee \ell_{2,j}$
  - tomamos $E_j = (\ell_{1,j} \vee \ell_{2,j} \vee y_j) \wedge (\ell_{1,j} \vee \ell_{2,j} \vee \overline{y}_j)$
  - donde $y_j$ es una variable nueva.
- Si $D_j$ tiene un sólo literal: $D_j = \ell_j$, se toma:
  - $E_j = (\ell_j \vee y_{1,j} \vee y_{2,j}) \wedge (\ell_j \vee \overline{y}_{1,j} \vee y_{2,j}) \wedge (\ell \vee y_{1,j} \vee \overline{y}_{2,j}) \wedge (\ell_j \vee \overline{y}_{1,j} \vee \overline{y}_{2,j})$
  - con $y_{1,j}, y_{2,j}$ las variables nuevas.

# 3SAT





- La dificultad es cuando $D_j$ tiene mas de 3 literales.
- Supongamos $D_j = \ell_{1,j} \vee ... \vee \ell_{k,j}$ con $k \geq 4$.
- Definimos $E_j$ de la siguiente forma, introduciendo nuevas variables $y_{1,j}, ..., y_{k-3,j}$

$$E_j = F_{0,j} \wedge F_{1,j} \wedge .... \wedge F_{k-3,j}$$

donde los $F_{i,j}$ son disjunciones de 3 literales, definidos en la página siguiente.

# 3SAT



$$
\begin{aligned}
F_{0,j} &= \quad \ell_{1,j} \vee \ell_{2,j} \vee y_{1,j} \\
F_{i,j} &= \quad \overline{y}_{i,j} \vee y_{i+1,j} \vee \ell_{i+2,j} \qquad \text{si } 1 \leq i < k-3 \\
F_{k-3,j} &= \quad \overline{y}_{k-3,j} \vee \ell_{k-1,j} \vee \ell_{k,j}
\end{aligned}
$$

Observar que si $k = 4$ entonces la condición $1 \leq i < k - 3$ nunca se cumple, es decir, para $k = 4$ solo tenemos $F_{0,j}$ y $F_{1,j} = \overline{y}_{1,j} \vee \ell_{3,j} \vee \ell_{4,j}$.

# 3SAT



- La construcción $B \mapsto A(B) = \tilde{B}$ es claramente polinomial, de hecho lineal, pues sólo agregamos un número lineal de nuevas variables y nuevas disjunciones.
- Tenemos que probar ahora que $SAT(B) = 3SAT(\tilde{B})$ es decir que $B$ es satisfacible si y solo si $\tilde{B}$ lo es.
- Recalquemos que $\tilde{B}$ tiene las mismas variables de $B$ mas otras variables extras asi que hay que ser cuidadoso.
- Hagamos cada implicación por separado empezando por $B$ satisfacible $\Rightarrow$ $\tilde{B}$ satisfacible.

# $B$ satisfacible $\Rightarrow$ $\tilde{B}$ satisfacible.



- Si $B$ es satisfacible, entonces existe un asignamiento de valores a las variables de $B$ (es decir a $x_1, x_2, ..., x_n$) que la vuelve verdadera.
- Entre las variables de $\tilde{B}$ tambien estan las $x_i$ asi que les damos este mismo asignamiento a ellas.
- Pero $\tilde{B}$ tiene ademas varias variables extras "$y$"a las cuales tenemos que darle un valor.
- Estas variables extras aparecen asociadas a cada $D_j$ dependiendo de cuantos literales tenga, asi que analicemos cada caso por separado.
- Como el asignamiento hace verdadera a $B$, entonces hace verdadero a todos los $D_j$ pues $B$ es conjunción de los $D_j$.

# $B$ satisfacible $\Rightarrow \tilde{B}$ satisfacible.



- Lo que queremos ver es cada $E_j$ se puede volver verdadero, con el asignamiento que vuelve verdadero a $D_j$, mas valores adecuadamente elegidos para las variables extras que aparecen en $E_j$ pero no en $D_j$.
- Si $D_j$ tiene exactamente 3 literales, no tiene asociado ninguna variable extra asi que no hay que hacer nada
- Para los $D_j$ con 1 o 2 literales, le podemos dar los valores que queramos a las variables extras "$y$" : con cualquiera de ellos $E_j$ se volverá verdadero. (ejercicio).

# $B$ satisfacible $\Rightarrow \tilde{B}$ satisfacible.



- Para el caso de $D_j$ con 4 o mas literales, como el asignamiento de valores a las $x$ lo vuelve verdadero y es una disjunción de literales, entonces al menos uno de sus literales debe haberse vuelto verdadero.
- Sea $r$ tal que $\ell_{r,j}$ es verdadero con el asignamiento de valores dado. (si hay mas de uno, tomo alguno).
- Demos valores a las $y$ que haga que $y_{i,j}$ sea verdadera para $i \leq r-2$ y falsa para $i \geq r-1$.
- Analicemos los distintos $F_{i,j}$.

# $B$ satisfacible $\Rightarrow$ $\tilde{B}$ satisfacible.



- Primero supongamos que $3 \leq r \leq k - 2$.
- En particular $1 \leq r - 2$ y como estamos especificando que $y_{i,j}$ es verdadera para $i \leq r - 2$, tenemos que $y_{1,j}$ es verdadera.
- Asi que $F_{0,j} = \ell_{1,j} \vee \ell_{2,j} \vee y_{1,j}$ es verdadero.
- Y para $1 \leq i < k - 3$, como $F_{i,j} = \overline{y}_{i,j} \vee y_{i+1,j} \vee \ell_{i+2,j}$ tiene la variable $y_{i+1,j}$ entonces $F_{i,j}$ es verdadero mientras $i + 1$ sea $\leq r - 2$, es decir, $i \leq r - 3$.

# $B$ satisfacible $\Rightarrow \tilde{B}$ satisfacible.



- Por otro lado, como $F_{i,j}$ tiene el literal $\overline{y}_{i,j}$, y la variable $y_{i,j}$ es FALSA si $i \geq r - 1$, con lo cual el literal $\overline{y}_{i,j}$ es verdadero en ese caso, tenemos que $F_{i,j}$ tambien es verdadero si $i \geq r - 1$.
- Esto tambien vale para $i = k - 3$ pues $F_{k-3,j} = \overline{y}_{k-3,j} \vee \ell_{k-1,j} \vee \ell_{k,j}$ tiene el literal $\overline{y}_{k-3,j}$ que esta evaluado a verdadero, pues $k - 3 \geq r - 1$ dado que estamos suponiendo que $r \leq k - 2$.
- Concluimos que $F_{i,j}$ es verdadero para $i \leq r - 3$ y para $i \geq r - 1$, asi que queda por ver el caso $i = r - 2$.

# $B$ satisfacible $\Rightarrow \tilde{B}$ satisfacible.



- Pero $F_{r-2,j} = \overline{y}_{r-2,j} \vee y_{r-1,j} \vee \ell_{r,j}$
- y $\ell_{r,j}$ es verdadero asi que $F_{r-2,j}$ es verdadero.
- Como todos los $F_{i,j}$ son verdaderos y $E_j = F_{0,j} \wedge F_{1,j} \wedge .... \wedge F_{k-3,j}$ concluimos que $E_j$ es verdadero en el caso $3 \leq r \leq k-2$.
- Quedan entonces los casos $r = 1, 2, k-3, k-4$ que dejamos como ejercicio.
- Veamos ahora la vuelta, es decir, $\tilde{B}$ satisfacible $\Rightarrow B$ satisfacible.

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



- Supongamos que $\tilde{B}$ es satisfacible. Entonces hay un asignamiento de valores a las variables de $\tilde{B}$ que hace que $\tilde{B}$ sea verdadera.
- Las variables de $\tilde{B}$ incluyen a las $x$ que son las variables de $B$, asi que les asignamos a esas variables los valores que se les asignan para que $\tilde{B}$ sea verdadera.
- Queremos ver que con ese asignamiento, $B$ es verdadera, asi que supongamos que no.
- Entonces algun $D_j$ no es verdadero, pues $B = D_1 \wedge .... \wedge D_m$.
- Es fácil ver que no puede ser uno de los que tiene 3 o menos literales. (ejercicio)

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



- Sea entonces $j$ tal que $D_j$ tiene 4 o mas literales y $D_j$ no evalua a verdadero con ese asignamiento.
- Esto significa que el asignamiento de valores a las variables vuelve a todos los $\ell_{r,j}, r = 1, 2, ..., k$ igual a 0 pues $D_j$ es la disjunción de ellos.
- Pero estamos suponiendo que $\tilde{B}$ evalua a verdadero, con lo cual $E_j$ debe evaluar a verdadero.
- Como $E_j$ es la conjunción de los $F_{i,j}$ concluimos que todos los $F_{i,j}$ evaluan a verdadero.
- Pero si evaluamos $F_{i,j}$ teniendo en cuenta que todos los $\ell_{r,j}$ evaluan a 0, obtenemos:

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$F_{0,j} = \ell_{1,j} \vee \ell_{2,j} \vee y_{1,j}$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$F_{0,j} = 0 \vee 0 \vee y_{1,j}$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.

$$F_{0,j} = y_{1,j}$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$
\begin{array}{rrcl}
 & F_{0,j} & = & y_{1,j} \\
(1 \leq i < k-3) \to & F_{i,j} & = & \overline{y}_{i,j} \vee y_{i+1,j} \vee \ell_{i+2,j}
\end{array}
$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$\begin{array}{rcl} F_{0,j} & = & y_{1,j} \\ (1 \leq i < k-3) \rightarrow \quad F_{i,j} & = & \overline{y}_{i,j} \vee y_{i+1,j} \vee 0 \end{array}$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$
\begin{array}{rrcl}
 & F_{0,j} & = & y_{1,j} \\
(1 \leq i < k-3) \rightarrow & F_{i,j} & = & \overline{y}_{i,j} \vee y_{i+1,j} \\
 & & = & (y_{i,j} \Rightarrow y_{i+1,j}) \\
 & F_{k-3,j} & = & \overline{y}_{k-3,j} \vee \ell_{k-1,j} \vee \ell_{k,j}
\end{array}
$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$
\begin{array}{rrcl}
 & F_{0,j} & = & y_{1,j} \\
(1 \leq i < k-3) \rightarrow & F_{i,j} & = & \overline{y}_{i,j} \vee y_{i+1,j} \\
 & & = & (y_{i,j} \Rightarrow y_{i+1,j}) \\
 & F_{k-3,j} & = & \overline{y}_{k-3,j} \vee 0 \vee 0
\end{array}
$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



$$
\begin{array}{rrcl}
 & F_{0,j} & = & y_{1,j} \\
(1 \leq i < k-3) \rightarrow & F_{i,j} & = & \overline{y}_{i,j} \vee y_{i+1,j} \\
 & & = & (y_{i,j} \Rightarrow y_{i+1,j}) \\
 & F_{k-3,j} & = & \overline{y}_{k-3,j}
\end{array}
$$

# $\tilde{B}$ satisfacible $\Rightarrow$ $B$ satisfacible.



- En definitiva, como $y_{i,j} \Rightarrow y_{i+1,j}$ es verdad para $1 \leq i < k - 3$ y la primera $y_{1,j} = F_{0,j}$ es verdadera, entonces todas las $y_{i,j}$ evaluan a verdadera.
- Esto es un absurdo pues $\overline{y}_{k-3,j} = F_{k-3,j}$ tambien evalua como verdadera.
- Fin.

# 3COLOR



**Teorema (Karp)**

3COLOR es NP completo.

- Prueba:
- Veremos que 3-SAT$\leq_P$3-COLOR.
- Para ello, dada una instancia $B$ de 3-SAT,
  - i.e., una expresion booleana en CNF con exactamente 3 literales en cada disjunción
- crearemos polinomialmente una instancia $A(B) = G$ de 3-COLOR.
  - i.e., un grafo $G$
- tal que $B$ sea satisfacible si y solo si $\chi(G) \leq 3$.

# Prueba de 3COLOR es NP completo



- Lo de polinomial es clave, pues si no seria trivial: definiriamos $G$ como un triangulo si $B$ es satisfacible y como $K_4$ si no.
- Pero eso requiere primero DECIDIR si $B$ es o no satisfacible, que es justo lo que no sabemos hacer en tiempo polinomial.
- Supongamos que las variables de $B$ son $x_1, ..., x_n$
- y que $B = D_1 \wedge ... \wedge D_m$,
- con las disjunciones $D_j = \ell_{1,j} \vee \ell_{2,j} \vee \ell_{3,j}$
  - $\ell_{k,j}$ literales
- Construiremos el grafo $G$ dando sus vértices y sus lados.

# Vértices de $G$





- Los vértices son:
  1. $2n$ vértices $v_\ell$, uno por cada literal $\ell$
     - (es decir, por cada variable $x_i$ hay dos vértices $v_{x_i}$ y $v_{\overline{x}_i}$)
  2. $6m$ vértices $\{e_{k,j}\}_{\substack{k=1,2,3\\ j=1,\dots,m}} \cup \{a_{k,j}\}_{\substack{k=1,2,3\\ j=1,\dots,m}}$
     - Es decir, para cada $j = 1, 2, ..., m$, seis vértices $e_{1,j}, e_{2,j}, e_{3,j}, a_{1,j}, a_{2,j}, a_{3,j}$
  3. Dos vértices especiales, $s$ y $t$
- Observar que los vértices se construyen en tiempo polinomial, porque simplemente los listamos siguiendo los nombres de las variables, el número de las disjunciones y son $2 + 2n + 6m$.

# Lados de $G$



- Los lados son tales de que construyen dos estructuras disjuntas, mas algunos lados extras que las unen.
- Una estructura es un abanico de triangulos todos con un vértice en común.
- La otra es una serie de “garras” disjuntas.
- Estas estructuras se unirán por medio del vértice $s$ y con lados directos entre los extremos de las garras y algunos extremos del abanico.

# Lados de $G$



- Los lados son:
  1. $3m$ lados $a_{1,j}a_{2,j}$, $a_{1,j}a_{3,j}$, $a_{3,j}a_{2,j}$. $(j = 1, 2, ...., m)$
     - Es decir, para cada $j$ tenemos un triangulo con los vértices $a_{1,j}a_{2,j}, a_{3,j}$.
  2. $3m$ lados $e_{k,j}a_{k,j}$, $(k = 1, 2, 3, j = 1, 2, ...., m.)$
     - Por lo tanto los $e$ son como “uñas” que salen de los $a$. Para cada $j$ los $a$ y los $e$ forman una especie de “garra”.
  3. $2n$ lados $tv_\ell$, uno por cada literal $\ell$.
  4. $n$ lados $v_x v_{\overline{x}}$, uno por cada variable $x$.
     - Es decir, para cada variable $x$ tenemos un triangulo con los vértices $t, v_x, v_{\overline{x}}$.
  5. El lado $st$
  6. $3m$ lados $se_{k,j}$ $(k = 1, 2, 3, j = 1, 2, ...., m.)$
  7. $3m$ lados $e_{k,j}v_{\ell_{k,j}}$ $(k = 1, 2, 3, j = 1, 2, ...., m.)$

- Los hemos construido simplemente leyendo las variables y los literales de cada disjunción, y son $1 + 3n + 12m$ asi que la construcción de $G$ es polinomial.
- (aunque el número fuese polinomial en $n$, $m$, si para decidir donde va cada lado tuviera que hacer un calculo no polinomial, la construcción no seria polinomial, pero aca esta especificado en forma simple donde va cada uno).
- Para que se entienda un poco mas como es la construcción, damos un ejemplo.
- Supongamos que

$$B = (x_1 \vee x_2 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_3 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4)$$

# Ejemplo

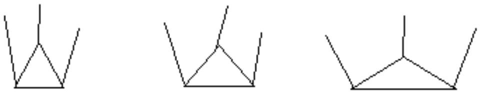


Entonces tenemos $n = 4$ variables y $m = 3$ disjunciones, cada una, como corresponde, con 3 literales.

# Ejemplo

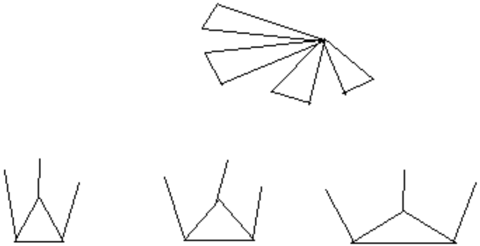


Las $m = 3$ garras:



# Ejemplo

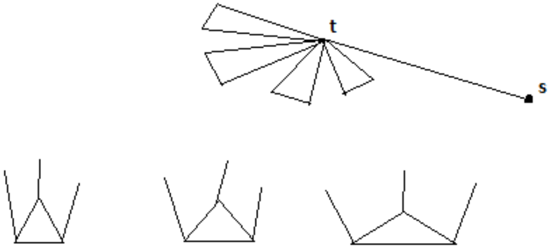
Y el abanico correspondiente a las $n = 4$ variables:



# Ejemplo



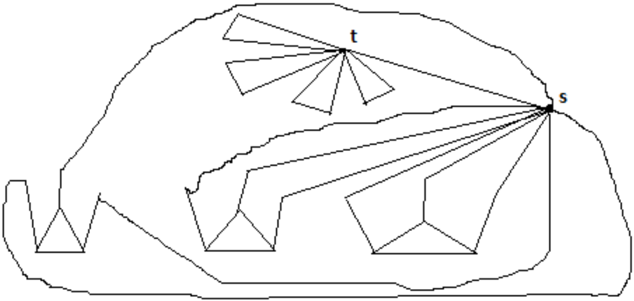
Unimos $t$ con $s$



# Ejemplo



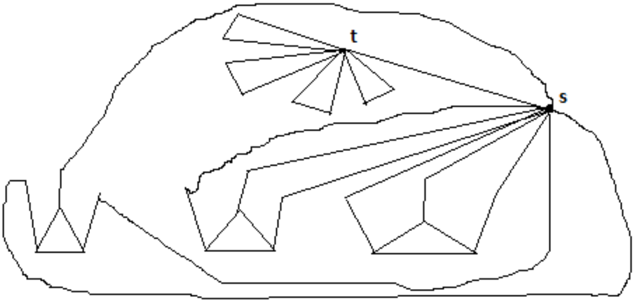
Y $s$ con todos los extremos de las garras:



# Ejemplo

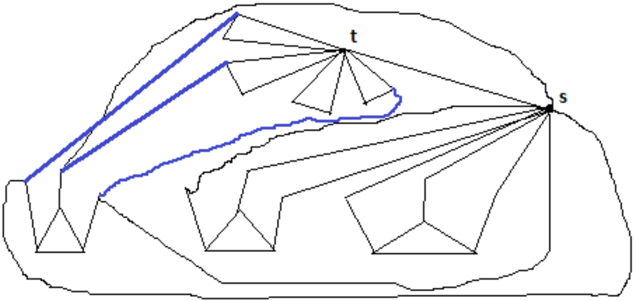


Ahora debemos dibujar todos los lados $e_{k,j}v_{\ell_{k,j}}$



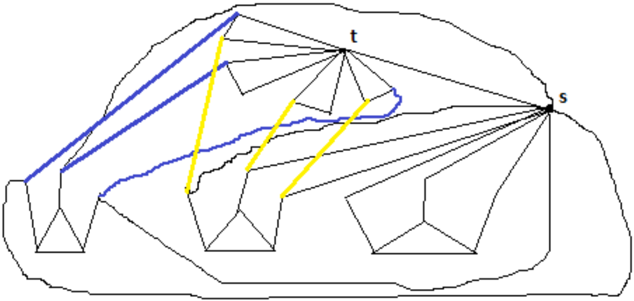
# Ejemplo

Veamos primero $D_1 = x_1 \vee x_2 \vee \overline{x}_4$.



# Ejemplo

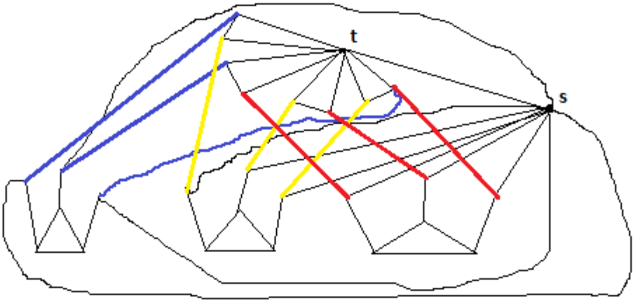
$$D_2 = \overline{x}_1 \vee x_3 \vee x_4.$$



# Ejemplo



$$D_3 = \overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4$$



# Prueba de 3COLOR es NP completo



- Volviendo a la prueba general, como $G$ tiene triangulos, entonces $\chi(G) \geq 3$.
- Asi que $\chi(G) \leq 3$ si y solo si $\chi(G) = 3$.
- Entonces lo que hay que probar que: $B$ es satisfacible si y solo si $\chi(G) = 3$.

# Prueba de 3COLOR es NP completo



- Vamos a necesitar distinguir entre las variables y los asignamientos de valores a las variables.
- Un asignamiento de valores, dado que las variables son booleanas, es darle valor 1 o 0 a cada variable.
- Por lo tanto un asignamiento de valores es un vector de bits en $\{0, 1\}^n$.
- Asi que lo denotaremos por $\vec{b}$.
- Y $x(\vec{b})$ es el valor que asume la variable $x$ en ese asignamiento.
- Por ejemplo si $n = 4$, $x_2(1, 0, 1, 1) = 0$, pero $x_4(1, 0, 1, 1) = 1$.

# Prueba de 3COLOR es NP completo



- Similarmente denotaremos por $\ell(\vec{b})$ el valor que asume el literal $\ell$
- Asi por ejemplo $\overline{x}_3(1,0,1,1) = 1 - x_3(1,0,1,1) = 1 - 1 = 0$.
- Y $D_j(\vec{b})$ el valor que asume toda la disjunción y $B(\vec{b})$ el valor que asume toda la expresión booleana.
- Veamos primero la implicación $B$ satisfacible $\Rightarrow \chi(G) = 3$.

# $B$ satisfacible $\Rightarrow \chi(G) = 3$



- Esta es fácil pero engorrosa: debemos dar un coloreo propio con 3 colores de $G$, asi que debemos colorear todos los vértices de $G$, uno por uno.
- Tenemos que usar que hay un asignamiento de valores a las variables de $B$ que la vuelve verdadera.
- Es decir, existe un $\vec{b} \in \{0, 1\}^n$ tal que $B(\vec{b}) = 1$.
- IMPORTANTE por algún motivo, posiblemente por memorizar sin entender, muchos en el final hacen todo el coloreo sin usar nunca que existe un $\vec{b} \in \{0, 1\}^n$ tal que $B(\vec{b}) = 1$. Una prueba de ese tipo, aunque ocupe muchas lineas, está mal y no suma puntos.

# $B$ satisfacible $\Rightarrow \chi(G) = 3$





- Coloreamos los vértices $v_\ell$ por medio de $c(v_\ell) = \ell(\vec{b})$
- Al ir coloreando los vértices, iremos chequeando que el coloreo sea propio.
- Para ello, chequearemos que en cada lado donde los dos vértices ya hayan sido coloreados, los extremos tengan colores distintos.
- Al chequearlo, diremos que el lado “no crea problemas".

# $B$ sat $\Rightarrow \chi(G) = 3$



- Ya tenemos coloreados los vértices $v_\ell$ y $v_x v_{\overline{x}}$ forman lado, asi que tenemos que chequearlos.
- Pero $c(v_{\overline{x}}) = \overline{x}(\vec{b}) = 1 - x(\vec{b}) \neq x(\vec{b}) = c(v_x)$ asi que $v_x v_{\overline{x}}$ no crea problemas
- Coloreemos ahora $c(s) = 1$ y $c(t) = 2$.
- Como $c(s) \neq c(t)$ entonces $st$ no crea problemas
- Como $c(t) = 2$ y $c(v_\ell) \in \{0, 1\}$, entonces $tv_\ell$ no crea problemas
- Ahora debemos colorear las garras, es decir los $e$ y $a$.
- Aca tenemos que usar que $B(\vec{b}) = 1$ (hasta ahora no lo hicimos)

# $B$ sat $\Rightarrow \chi(G) = 3$



- Como $B(\vec{b}) = 1$ y $B = D_1 \wedge ... \wedge D_m$, entonces $D_j(\vec{b}) = 1\ \forall j$.
- Como $D_j = \ell_{1,j} \vee \ell_{2,j} \vee \ell_{3,j}$ entonces por lo de arriba para todo $j$ existe al menos un $k_j$ tal que $\ell_{k_j,j}(\vec{b}) = 1$. $(\ddagger)$
- Si hay mas de un tal $k_j$, elegimos uno sólo, pej el primero.
- Coloreamos, para cada $j$:
  - $c(a_{k_j,j}) = 2$
  - Para los $a_{r,j}$ con $r \neq k_j$, que son los dos que quedan, coloreamos uno de ellos con 1, y el otro con 0.
- Esto significa que el triangulo formado por los $a_{k,j}$ tiene los tres vértices de distinto color, por lo tanto no crea problemas

# $B$ sat $\Rightarrow \chi(G) = 3$



- Quedan los $e$, que estan unidos con $s$, algunos $v_\ell$ y los $a$.
- Coloreamos, para cada $j$:
  - $c(e_{r,j}) = 2$ para los dos $r \neq k_j$.
  - y: $c(e_{k_j,j}) = 0$.
- Como para $r \neq k_j$ tenemos $c(e_{r,j}) = 2$ y $c(a_{r,j}) \in \{0,1\}$ entonces $e_{r,j}a_{r,j}$ no crea problemas para esos $r$.
- Como $c(e_{k_j,j}) = 0$ y $c(a_{k_j,j}) = 2$ entonces $e_{k_j,j}a_{k_j,j}$ no crea problemas
- Por lo tanto las "garras" estan coloreadas de forma tal de no tener problemas internos.
- Hay que ver que pasa con los lados que las conectan con el resto del grafo.

# $B$ sat $\Rightarrow \chi(G) = 3$



- Primero, veamos $s$, que esta conectado con todos los $e$:
- Pero $se_{i,j}$ no crea problemas pues $c(s) = 1$ y $c(e_{i,j}) \in \{2, 0\}$.
- Sólo nos queda ver los lados $e_{r,j}v_{\ell_{r,j}}$.
- El caso $r \neq k_j$ es facil: tenemos $c(e_{r,j}) = 2$ y $c(v_{\ell_{r,j}}) \in \{0, 1\}$ asi que $e_{r,j}v_{\ell_{r,j}}$ no crea problemas para esos $r$.

## $B$ sat $\Rightarrow \chi(G) = 3$



- Queda el caso $k_j$.
- Tenemos $c(v_{\ell_{k_j,j}}) = \ell_{k_j,j}(\vec{b}) = 1$ (por ‡)
- Por lo tanto, como $c(e_{k_j,j}) = 0$ y $c(v_{\ell_{k_j,j}}) = 1$, entonces $e_{k_j,j}v_{\ell_{k_j,j}}$no crea problemas
- Hemos terminado de colorear todos los vértices con 3 colores y chequeado que el coloreo es propio.
- Por lo tanto, hemos completado la implicación $B$ satisfacible $\Rightarrow \chi(G) = 3$.
- Veamos la vuelta.

# $\chi(G) = 3 \Rightarrow B$ satisfacible



- Suponemos entonces que existe un coloreo propio $c$ de $G$ con tres colores.
- A partir de $c$ debemos definir un $\vec{b} \in \{0,1\}^n$ tal que $B(\vec{b}) = 1$.
- Otra vez, si nunca lo definen, o no usan el coloreo $c$, la prueba de esta parte no vale nada.
- Definimos $\vec{b}$ usando el coloreo $c$ de la siguiente forma, con la notación [] que vimos en la parte de flujos:

$$b_i = [c(v_{x_i}) = c(s)]$$

- Es decir $b_i = 1$ si $c(v_{x_i}) = c(s)$ y $b_i = 0$ si $c(v_{x_i}) \neq c(s)$

# $\chi(G) = 3 \Rightarrow B$ satisfacible



- Sea $\ell$ un literal cualquiera tal que $c(v_\ell) = c(s)$.
- Si $\ell$ es una variable: $\exists i$ con $\ell = x_i$
  - Entonces decir $c(v_\ell) = c(s)$ es lo mismo que decir $c(v_{x_i}) = c(s)$.
  - Por la definición de $\vec{b}$ eso implica que $b_i = 1$.
  - Con lo que $\ell(\vec{b}) = x_i(\vec{b}) = b_i = 1$.
- Si $\ell$ es negación de una variable:$\exists i$ con $\ell = \overline{x}_i$
  1. Entonces decir $c(v_\ell) = c(s)$ es lo mismo que $c(v_{\overline{x}_i}) = c(s)$.
  2. Como $v_{x_i} v_{\overline{x_i}} \in E$ entonces $c(v_{x_i}) \neq c(v_{\overline{x}_i})$.
  3. Juntando [1] y [2] concluimos que $c(v_{x_i}) \neq c(s)$.
  4. El punto [3] implica que $b_i = 0$.
  5. Entonces $\ell(\vec{b}) = \overline{x}_i(\vec{b}) = 1 - x_i(\vec{b}) = 1 - b_i = 1 - 0 = 1$.
- En cualquiera de los casos hemos probado que $c(v_\ell) = c(s) \Rightarrow \ell(\vec{b}) = 1$.

# $\chi(G) = 3 \Rightarrow B$ satisfacible.



- Sea ahora $j \in \{1, 2, ..., m\}$.
- Como los $a_{k,j}$ forman un triangulo, y $c$ colorea $G$ con tres colores, entonces los tres colores deben estar representados en ese triangulo.
- En particular, existe un $q$ tal que $c(a_{q,j}) = c(t)$.
- Analicemos el color posible del $e_{q,j}$ asociado a $a_{q,j}$.
  1. $e_{q,j}a_{q,j} \in E \Rightarrow c(e_{q,j}) \neq c(a_{q,j}) = c(t)$.
  2. $e_{q,j}s \in E \Rightarrow c(e_{q,j}) \neq c(s)$.
- Por lo tanto $e_{q,j}$ debe tener el “tercer" color, es decir, el color que no es el color de $s$ ni el color de $t$.

# $\chi(G) = 3 \Rightarrow B$ satisfacible.



- (Como $st \in E$, entonces necesariamente $c(s) \neq c(t)$ asi que efectivamente sólo queda un color libre para $e_{q,j}$)
  1. $v_{\ell_{q,j}} e_{q,j} \in E \Rightarrow c(v_{\ell_{q,j}}) \neq c(e_{q,j})$.
  2. $v_{\ell_{q,j}} t \in E \Rightarrow c(v_{\ell_{q,j}}) \neq c(t)$.
- Como $e_{q,j}$ tiene el tercer color, los dos puntos anteriores implican que $v_{\ell_{q,j}}$ no puede tener ni el tercer color ni el color de $t$.

# $\chi(G) = 3 \Rightarrow B$ satisfacible.



- Pero si $v_{\ell_{q,j}}$ no tiene ni el color de $t$ ni el tercer color, sólo le queda tener el color de $s$: $c(v_{\ell_{q,j}}) = c(s)$, pues sólo hay 3 colores disponibles.
- Probamos antes que eso implica que $\ell_{q,j}(\vec{b}) = 1$.
- Como $D_j = \ell_{1,j} \vee \ell_{2,j} \vee \ell_{3,j}$ entonces $\ell_{q,j}(\vec{b}) = 1$ implica que $D_j(\vec{b}) = 1$.
- El $j$ era cualquiera en $\{1, 2, ..., m\}$, es decir, hemos probado que $D_j(\vec{b}) = 1$ para todo $j$.
- Como $B = D_1 \wedge ... \wedge D_m$, lo anterior implica que $B(\vec{b}) = 1$ asi que $B$ es satisfacible y hemos concluido la prueba.