13 de mayo 2022

Reducciones

Theoretical computer science:

27.5 Proposition. $\vdash_{\mathbf{K}+\{A3\}} \boxdot(A \leftrightarrow B) \rightarrow \boxdot(F(A) \leftrightarrow F(B))$.

27.16 Lemma. $w \models \boxdot(p \leftrightarrow A) \rightarrow \boxdot(\Box C_i(p) \rightarrow \Box C_i(H_i))$.

Also theoretical computer science:

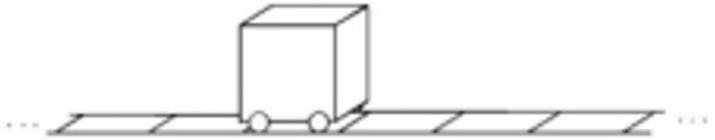


Figure 3-1. A Turing machine.

Definición. Dados lenguajes L_1 y L_2 sobre un alfabeto $\mathbf{A}$, decimos que L_1 puede ser reducido en tiempo polinomial a L_2 si existe una función f computable en tiempo polinomial y tal que para todo $w \in \mathbf{A}^*$ se tiene que $w \in L_1$ si y solo si $f(w) \in L_2$.

Definición. Dados lenguajes L_1 y L_2 sobre un alfabeto $\mathbf{A}$, decimos que L_1 puede ser reducido en tiempo polinomial a L_2 si existe una función f computable en tiempo polinomial y tal que para todo $w \in \mathbf{A}^*$ se tiene que $w \in L_1$ si y solo si $f(w) \in L_2$.

Teorema. Dados lenguajes L_1 y L_2 sobre un alfabeto $\mathbf{A}$, suponga que L_1 puede ser reducido en tiempo polinomial a L_2. Entonces:

$$\text{Si } L_2 \in \text{PTIME entonces } L_1 \in \text{PTIME}.$$

$$\text{Si } L_1 \notin \text{PTIME entonces } L_2 \notin \text{PTIME}.$$

Definición. Sea C un conjunto de lenguajes que contiene a PTIME. Decimos que un lenguaje L es *hard* para C si para todo $L' \in C$ existe una reducción polinomial de L' a L.

Definición. L es completo para C si L es hard para C y a la vez $L \in C$.

Obviamente, si PTIME $\subsetneq C$ y L es completo para C entonces $L \notin$ PTIME.

Si L es hard para C, decimos que L es C-hard. Si es completo para C decimos que es C-completo.

SAT y 3-col son NP Completo

Teorema (Cook-Levin). SAT, CNF-SAT y 3-CNF-SAT son NP-completos.

Ahora ya tenemos un lenguaje NP-completo, podemos pasar a demostrar otros lenguajes NP-completos. Para eso vamos a ayudarnos de un pequeño lema.

Lema. Sea C una clase de complejidad y L_1 y L_2 dos lenguajes. Si L_1 es C-hard, y además hay una reducción polinomial desde L_1 a L_2, entonces L_2 es C-hard también.

4-Col

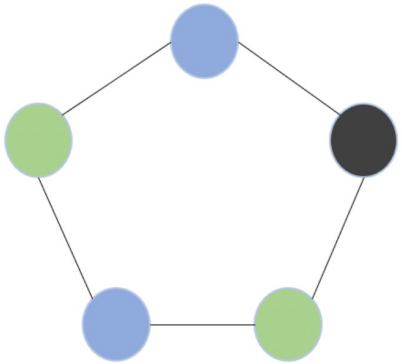
Teorema. Sean C y C' clases de complejidad tales que $\text{PTIME} \subseteq C \subseteq C'$. Si un lenguaje L es C'-completo y además L está en C, entonces $C = C'$.

Considera el lenguaje de 4-coloración: son todos los grafos en los que puedo pintar los nodos con uno de cuatro colores, y tal que ninguna arista tiene sus dos nodos con el mismo color. Muestra que el lenguaje 4COL es NP-completo:

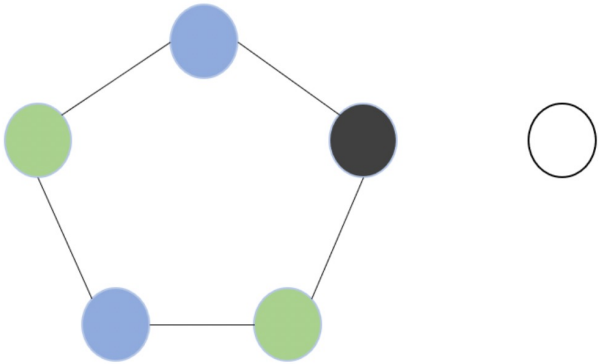
$$\text{4COL} = \{w \mid w = C(G) \text{ y } G \text{ es 4-coloreable}\}$$

Intuicion

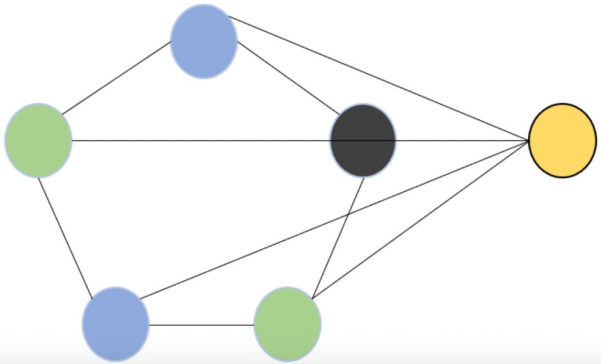
1)



2)



3)



Intuicion

La intuicion es tomar un grafo 3-coloreable y agregar un nodo con un 4to color, entonces este nuevo grafo G' es 4-coloreable valido. Ademas notamos que si el grafo no es 3-colorable, entonces no puede ser 4-coloreable. Finalmente claramente la reducción es valida y se ejecuta en tiempo polinomial.

K-Clique

Definicion (Clique):

Dado un grafo $G = (V, E)$, un clique es un subconjunto de vértices C tal que $\forall u \forall v \in C, (u, v) \in E$. En otras palabras, todos los vértices de C son vecinos

K-Clique

Definicion (Clique):

Dado un grafo $G = (V, E)$, un clique es un subconjunto de vértices C tal que $\forall u \forall v \in C, (u, v) \in E$. En otras palabras, todos los vértices de C son vecinos

Problema (K-Clique)

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

K-Clique

Definicion (Clique):

Dado un grafo $G = (V, E)$, un clique es un subconjunto de vértices C tal que $\forall u \forall v \in C, (u, v) \in E$. En otras palabras, todos los vértices de C son vecinos

Problema (K-Clique)

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

Claramente K-Clique es un problema NP, por lo que no demostraremos eso ahora, si no que demostraremos que es NP-Completo

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

Gracias a las demostraciones de ***Karp***, sabemos que *3-SAT* es NP-Completo. Utilizaremos esto para fundamentar la reducción

Demostración:

Dado una formula 3-CNF llamada F de m clausulas sobre n variables. Construiremos el grafo de la siguiente forma:

Primero, para cada clausula $c \in F$ creamos un nodo por cada asignación de variables a c que satisfaga c: Por ejemplo: Si la formula es

$$F = (x_1 \vee x_2 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_3) \wedge \dots$$

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k.¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

Crearíamos los nodos de la siguiente forma:

$$\begin{array}{lll} (x_1 = 0, x_2 = 0, x_4 = 0) & (x_3 = 0, x_4 = 0) & (x_2 = 0, x_3 = 0) \quad \ldots \\ (x_1 = 0, x_2 = 1, x_4 = 0) & (x_3 = 0, x_4 = 1) & (x_2 = 0, x_3 = 1) \\ (x_1 = 0, x_2 = 1, x_4 = 1) & (x_3 = 1, x_4 = 1) & (x_2 = 1, x_3 = 0) \end{array}$$

Luego, creamos una arista entre dos nodos si las asignaciones hasta ese momento son consistentes.

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k.¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

Crearíamos los nodos de la siguiente forma:

$$\begin{array}{lll} (x_1 = 0, x_2 = 0, x_4 = 0) & (x_3 = 0, x_4 = 0) & (x_2 = 0, x_3 = 0) \quad \ldots \\ (x_1 = 0, x_2 = 1, x_4 = 0) & (x_3 = 0, x_4 = 1) & (x_2 = 0, x_3 = 1) \\ (x_1 = 0, x_2 = 1, x_4 = 1) & (x_3 = 1, x_4 = 1) & (x_2 = 1, x_3 = 0) \end{array}$$

Luego, creamos una arista entre dos nodos si las asignaciones hasta ese momento son consistentes.

Con nuestro grafo generado, podemos ahora notar que el tamaño máximo de un clique de este grafo seria igual a m porque no existen aristas entre dos nodos de la misma clausula c

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

Ahora, además. Si el problema 3-SAT es satisfacible (Posee al menos una asignación valida) entonces, **deberá** existir al menos un *m-clique.*

Sin embargo, recordemos la definición de reducción

Definición. Dados lenguajes L_1 y L_2 sobre un alfabeto $\mathbf{A}$, decimos que L_1 puede ser reducido en tiempo polinomial a L_2 si existe una función f computable en tiempo polinomial y tal que para todo $w \in \mathbf{A}^*$ se tiene que $w \in L_1$ si y solo si $f(w) \in L_2$.

Entonces, al ser un SSI, hemos de demostrar además que dada la función F si es que no existe una asignación valida, entonces el máximo tamaño de un clique en el grafo creado es a lo mas $K - 1$

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ clique\ de\ tamaño \geq k?$$

Demostraremos esto de forma rápida:
Si el grafo posee un z-clique, entonces este clique debe contener un nodo por clausula ***c.*** Entonces si no es satisfacible, $z < m$. Luego demostrando lo que necesitabamos

Finalmente, hemos de mostrar que la reducción ocurre en tiempo polinomial. Aca es simple, ya que sabemos que a lo mas se generara m nodos y a lo mas m^2 aristas. Luego es polinomial y por ende la reduccion es valida

Finalmente K-Clique es NP-Completo por reducción.

Independent Set

Definicion (Independent Set):

Dado un grafo $G = (V, E)$, un conjunto independientes es un subconjunto de vértices C tal que $\forall u \forall v \in C, (u, v) \notin E$. En otras palabras, es un Anticlique

Problema (Independent set)

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ Independent\ Set\ de\ tamaño \geq k?$$

$$Sea\ G\ un\ grafo\ y\ un\ entero\ k. ¿\ G\ contiene\ un\ Independent\ Set\ de\ tamaño \geq k?$$

Realizaremos la reducción desde el ***k-clique*** .

Dada una instancia *(G, k)* del problema k-clique. Extraemos *(H, k)* como la solución del problema *Independent-Set.* Donde definimos H como el complemento de G. Esto es

Si G posee la arista (u,v) Ssi H no posee arista (u,v). Podemos ver fácilmente la doble dirección de la reducción y como se ejecuta en tiempo polinomial.

Luego, Independent-Set es NP-Completo