Trabajo Práctico Nro 6 - NP-completitud

Ejercicio 1 - Consultar el tiempo polinomial

Responder breve y claramente:

1. Dar un esquema de prueba de la transitividad de las reducciones polinomiales. *Ayuda: en clase hicimos lo propio con las reducciones generales.*

Queremos probar que las **reducciones polinomiales** tienen la **propiedad de transitividad** como las generales, es decir, si $L_1 \leq_p L_2$ y $L_2 \leq_p L_3$, entonces $L_1 \leq_p L_3$.

Suponemos lo siguiente:

- $L_1 \leq_p L_2$ mediante una función $f : \Sigma^* \to \Sigma^*$.
- $L_2 \leq_p L_3$ mediante una función $g : \Sigma^* \to \Sigma^*$.
- Por definición de las *reducciones polinomiales*:
  - f y g son computables en *tiempo polinomial* por máquinas de Turing deterministas.
  - Para todo $w, x \in \Sigma^*$:
    - $w \in L_1 \iff f(w) \in L_2$,
    - $x \in L_2 \iff g(x) \in L_3$.

Ahora vamos a crear la **reducción polinomial** que demuestre la **transitividad**, para eso definimos una **nueva función** $h : \Sigma^* \to \Sigma^*$ como $h(w) = g(f(w))$, es decir, h aplica primero f, y luego g al resultado. Una vez definida la nueva función, tenemos que verificar la **correctitud** de la misma y que respete el **tiempo polinomial:**

- *Correctitud:*
  - $w \in L_1 \iff f(w) \in L_2$ y $f(w) \in L_2 \iff g(f(w)) \in L_3$, entonces, por transitividad $w \in L_1 \iff h(w) \in L_3$. Demostramos que h cumple con la *condición de correctitud* para una reducción de L_1 a L_3.
- *Tiempo polinomial:*
  - Como f es computable en *tiempo polinomial*, sabemos que $f(w)$ se computa en un tiempo $\leq poly(|w|)$.
  - A su vez, como g es computable en *tiempo polinomial*, sabemos que $g(x) \text{ donde x ahora es f(w)}$ se computa en un tiempo $\leq poly(poly(|w|))$.
  - Entonces, como f y g son *computables en tiempo polinomial* el $Tiempo(h(w)) \leq poly(poly(|w|))$ que sigue siendo *polinomial.*

2. ¿Cuándo un lenguaje es $NP - difícil$ y cuándo es $NP - completo$?

Un lenguaje L se considera $NP - difícil$ si todo lenguaje de la clase NP se puede **reducir polinomialmente** a él.

Un lenguaje L es $NP - completo$ si cumple dos condiciones:

- L pertenece a la clase NP.
- L es $NP - difícil$. Es decir, todo lenguaje de NP se reduce polinomialmente a L.
  En resumen, *un lenguaje $NP - completo$ es un problema que está en NP y es tan difícil como cualquier otro problema en NP.*

3. ¿Por qué si $P \neq NP$, un lenguaje $NP - completo$ no pertenece a P?

Si $P \neq NP$, un lenguaje $NP - completo$ no pertenece a P porque **si un lenguaje $NP - completo$ L perteneciera a P, entonces se demostraría que $P = NP$.**

Si **L fuera un lenguaje $NP - completo$** entonces sabemos por definición que:

- L pertenece a la clase NP.
- L es $NP - difícil$. Es decir, todo lenguaje de NP se reduce polinomialmente a L.

Ahora, suponiendo que **nuestro lenguaje L también pertenece a la clase P** podríamos considerar un lenguaje L_1 que pertenezca a NP y, debido a que L es $NP - completo$, sabemos que existe una **reducción polinomial** $L_1 \leq_p L$ y como suponemos que L pertenece a P, entonces L_1 también pertenece a P. Como L_1 era un **lenguaje cualquiera** en NP, habríamos demostrado que todo lenguaje en NP pertenece también a P ($NP \subseteq P$), pero sabemos que **$P \subseteq NP$ por definición** (todo lenguaje decidible en tiempo polinomial también puede ser verificado en tiempo polinomial, trivialmente sin necesidad de un certificado, o por una máquina de Turing no determinística que simula la determinística). **Por lo tanto, si un lenguaje $NP - completo$ perteneciera a P, se concluiría que $NP \subseteq P$ y $P \subseteq NP$, lo que implica que $P = NP$.** Pero dado que la condición inicial es que $P \neq NP$, la suposición de que un lenguaje NP-completo pertenece a P debe ser **falsa**. **Por lo tanto, si $P \neq NP$, ningún lenguaje NP-completo puede pertenecer a P.**

4. Enunciar el esquema visto en clase para agregar un lenguaje a la clase NPC.

El esquema nos dice:

- Demostrar que L_1 pertenece a la clase NP.
- Elegir un lenguaje L que ya se sabe que es $NP - completo$ (como SAT o algún otro lenguaje previamente probado como $NP - completo$) y construir una reducción polinomial

de L a L_1. Esto implica encontrar una función f computable en tiempo polinomial tal que para toda cadena w, $w \in L$ si y solo si $f(w) \in L_1$.

Debido a la **transitividad de las reducciones polinomiales**, si todo lenguaje en NP se reduce a L, y L se reduce polinomialmente a L_1, entonces todo lenguaje en NP también se reduce polinomialmente a L_1, cumpliendo así la condición de que L_1 es $NP - difícil$. Al pertenecer también a NP, se concluye que L_1 es $NP - completo$.

5. ¿Cuándo se sospecha que un lenguaje de NP está en NPI?

Sospechamos que un lenguaje de NP está en NPI cuando:

- Se cree que el lenguaje *no puede ser decidido por una Máquina de Turing en tiempo polinomial*, lo que sugiere que no pertenecería a la clase P.
- No se ha encontrado una *reducción polinomial desde algún lenguaje conocido que sea $NP - completo$ hacia este lenguaje*, lo que sugiere que no pertenecería a la clase $NP - completo$.

Ejercicio 2

Probar:

1. Si $L_1 \in NPC$ y $L_2 \in NPC$, entonces $L_1 \leq_P L_2$ y $L_2 \leq_P L_1$.

Sabemos que L_1 pertenece a la clase NP (por la primera condición de su $NP - completitud$). Como L_2 es $NP - completo$, la segunda condición de su definición nos dice que todo lenguaje en NP se reduce polinomialmente a L_2. Dado que L_1 es un lenguaje en NP, podemos tomar que $L_1 \leq_P L_2$.

De manera análoga, **sabemos que L_2 pertenece a la clase NP** (por la primera condición de su $NP - completitud$). Como L_1 es $NP - completo$, la segunda condición de su definición nos dice que todo lenguaje en NP se reduce polinomialmente a L_1. Dado que L_2 es un lenguaje en NP, podemos tomar que $L_2 \leq_P L_1$.

En conclusión, si L_1 y L_2 son ambos lenguajes NP-completos, entonces, se deduce que $L_1 \leq_P L_2$ y $L_2 \leq_P L_1$.

2. Si $L_1 \leq_P L_2$, $L_2 \leq_P L_1$, y $L_1 \in NPC$, entonces $L_2 \in NPC$.
   Ayuda: recurrir directamente a la definición de NPC.

Para demostrar que $L_2 \in NPC$, debemos probar dos cosas para L_2:

- $L_2 \in NP$.
- Para todo $L_i \in NP$, $L_i \leq_P L_2$ (L_2 es $NP - difícil$).

Por **teorema** sabemos que si $L_2 \leq_p L_1$ y $L_1 \in NP$, entonces $L_2 \in NP$. Por lo tanto, **hemos demostrado que L_2 pertenece a la clase NP.**

Por propiedad sabemos que las **reducciones polinomiales son transitivas**. Esto significa que si existe una reducción polinomial de L_i a L_1 ($L_i \leq_P L_1$) y una reducción polinomial de L_1 a L_2 ($L_1 \leq_P L_2$), **entonces existe una reducción polinomial de L_i a L_2 ($L_i \leq_P L_2$).** Como esto se cumple para todo $L_i \in NP$, se demuestra que **todo lenguaje en NP se puede reducir polinomialmente a L_2, lo que significa que L_2 es $NP - difícil$.**

En conclusión, se demuestra la $NP - completitud$ para L_2.

Ejercicio 3

Un lenguaje es $CO - NP - completo$ sii todos los lenguajes de $CO - NP$ se reducen polinomialmente a él. Probar que SAT^C es $CO - NP - completo$. *Ayuda: $L_1 \leq L_2$ sii $L_1^C \leq L_2^C$.*

En la teoría se nos explica que el **lenguaje SAT es $NP - completo$** por lo tanto, **por definición:**

- SAT pertenece a NP.
- Para todo $L_i \in NP$, $L_i \leq_P SAT$ (SAT es $NP - difícil$).

Al SAT pertenecer a NP sabemos que por definición su complemento debe pertenecer a $CO - NP$, es decir, **SAT^C pertenece a $CO - NP$.**

Al saber que para todo $L_i \in NP$, $L_i \leq_P SAT$ (SAT es $NP - difícil$) por definición todo $L_i^C \in CO - NP$, $L_i^C \leq_p SAT^C$, de esta forma demostramos que todos los lenguajes de $CO - NP$ se **reducen polinomialmente** a SAT^C, por lo tanto, **SAT^C pertenece a $CO - NP - completo$.**

Ejercicio 4

Sean los lenguajes A y B, tales que $A \neq \emptyset$, $A \neq \Sigma*$, y $B \in P$. Probar: $(A \bigcap B) \leq_P A$. *Ayuda: intentar con una reducción polinomial que, dada una cadena w, lo primero que haga sea chequear si $w \in B$, teniendo en cuenta que existe un elemento e que no está en A.*

Cosas que sabemos:

- Al $A \neq \emptyset$, $A \neq \Sigma*$ nos damos cuenta que existe al menos una cadena e que no está en A.
- Al $B \in P$ sabemos que existe una máquina de Turing M_B que decide B en tiempo polinomial.

Idea general: Primero tomamos la cadena w y verificamos que $w \in B$ utilizando la máquina M_B que sabemos que decide B en tiempo polinomial, si $w \notin B$ transformamos w a esta cadena e que sabemos que no está en A, si $w \in B$ mantenemos la cadena w para verificar que pertenezca a A.

Reducción: vamos a tener la función $f(w) = w \text{ si w} \in \text{B } o\ e \text{ si w} \notin \text{B}$. La función se computa en **tiempo polinomial** ya que las tareas que realiza pueden ser:

- Mantener la misma cadena w que es *polinomial*.
- Cambiar w por la constante e que tomamos en cuenta, que también es *polinomial*.

Verificación de la correctitud:

1. Si $w \in A \cap B$:
    - Entonces $w \in B \rightarrow f(w) = w$.
    - Como $w \in A \rightarrow f(w) = w \in A$.

- *Entonces $f(w) \in A$.*

2. Si $w \notin A \cap B$:
    - Si $w \notin B \rightarrow f(w) = e$, pero $e \notin A$ por construcción $\rightarrow f(w) \notin A$.
    - Si $w \in B$ pero $w \notin A \rightarrow f(w) = w \notin A$.

- *En ambos casos, $f(w) \notin A$.*

Ejercicio 5

Sea el lenguaje
$SH - s - t = \{(G, s, t) \mid \text{G es un grafo no dirigido y tiene un camino de Hamilton del vértice s al vértice}$. Un grafo $G = (V, E)$ tiene un camino de Hamilton del vértice s al vértice t sii G tiene un camino entre s y t que recorre todos los vértices restantes una sola vez. Probar que $SH - s - t$ es $NP - completo$. *Ayuda: se sabe que CH, el lenguaje correspondiente al problema del circuito hamiltoniano, es $NP - completo$.*

Para verificar que $SH - s - t$ es $NP - completo$ tenemos que verificar 2 cosas:

- $SH - s - t$ es NP, es decir, existe un *certificado sucinto verificable en tiempo polinomial*.

- $SH - s - t$ es $NP - dificil$.

Como **certificado sucinto verificable en tiempo polinomial** nosotros enviamos el camino de aristas desde s hasta t que cumple la condición para pertenecer a $SH - s - t$. Es **sucinto** porque:

- Podemos verificar que todas las aristas del certificado están en la lista de aristas del grafo G que recibimos de entrada en un *tiempo polinomial* $O(|E|^2)$.
- El verificar el camino consiste en recorrer la lista de vértices del grafo G que recibimos de entrada, por cada vértice verificamos que esté una vez dentro de la lista de aristas del certificado. Teniendo en cuenta que el primer vértice a chequear sea el vértice s y el último vértice a chequear sea el vértice t. Todo esto se puede hacer en *tiempo polinomial* con $O(|V| \cdot |E|)$.
  Teniendo en cuenta el certificado que usamos, **el orden de complejidad para la verificación en el peor de los casos es de $O(G^2)$ que es polinomial**. De esta forma verificamos que $SH - s - t \in NP$.

Para verificar que $SH - s - t$ es $NP - dificil$ vamos a hacer una **reducción polinomial** de CH a $SH - s - t$ ($SH - s - t \leq_p CH$), al $CH \in NP - completo$ si llegamos a poder realizar esta reducción demostraríamos que $SH - s - t$ es $NP - dificil$.

Idea general: Si tenemos un grafo G que pertenece a CH, es decir, tiene un circuito hamiltoniano lo que vamos a hacer es:

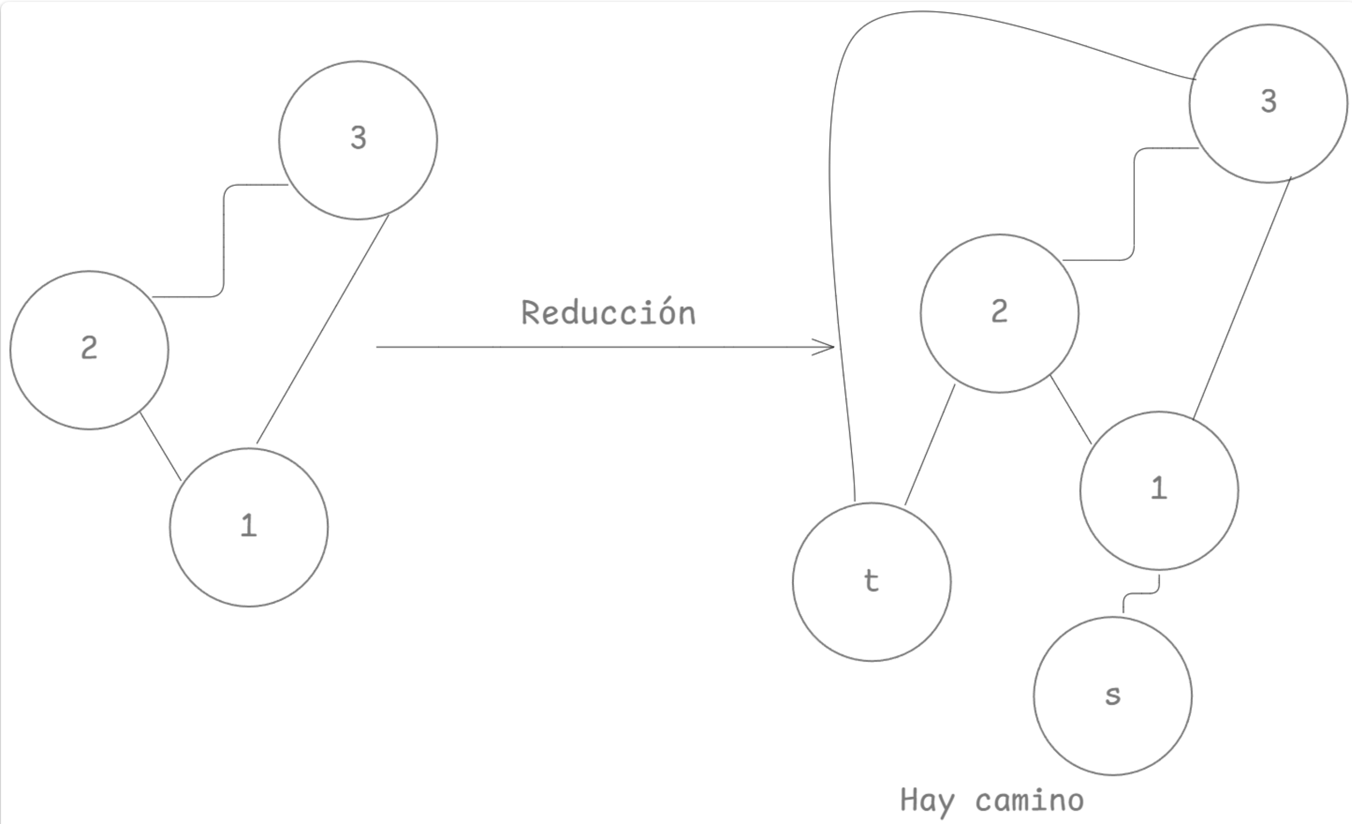
- Agregar el vértice s como un nuevo vecino a un nodo cualquiera de G, modificando las listas V y E del grafo G.
- Agregar el vértice t como un nuevo vecino a todos los vecinos originales del nodo que elegimos anteriormente de G, modificando las listas V y E del grafo G.
  Al hacer estas modificaciones garantizamos que si el grafo G originalmente tenía un circuito hamiltoniano, ahora tendrá un camino hamiltoniano desde el vértice s al vértice t.

Reducción: vamos a tener la función $f((V, E)) = (V', E')$ donde V' es la nueva lista de vértices con s y t, y E' es la nueva lista de aristas con las aristas necesarias para cumplir la idea general de la reducción. La función se computa en un *tiempo polonomial* ya que:

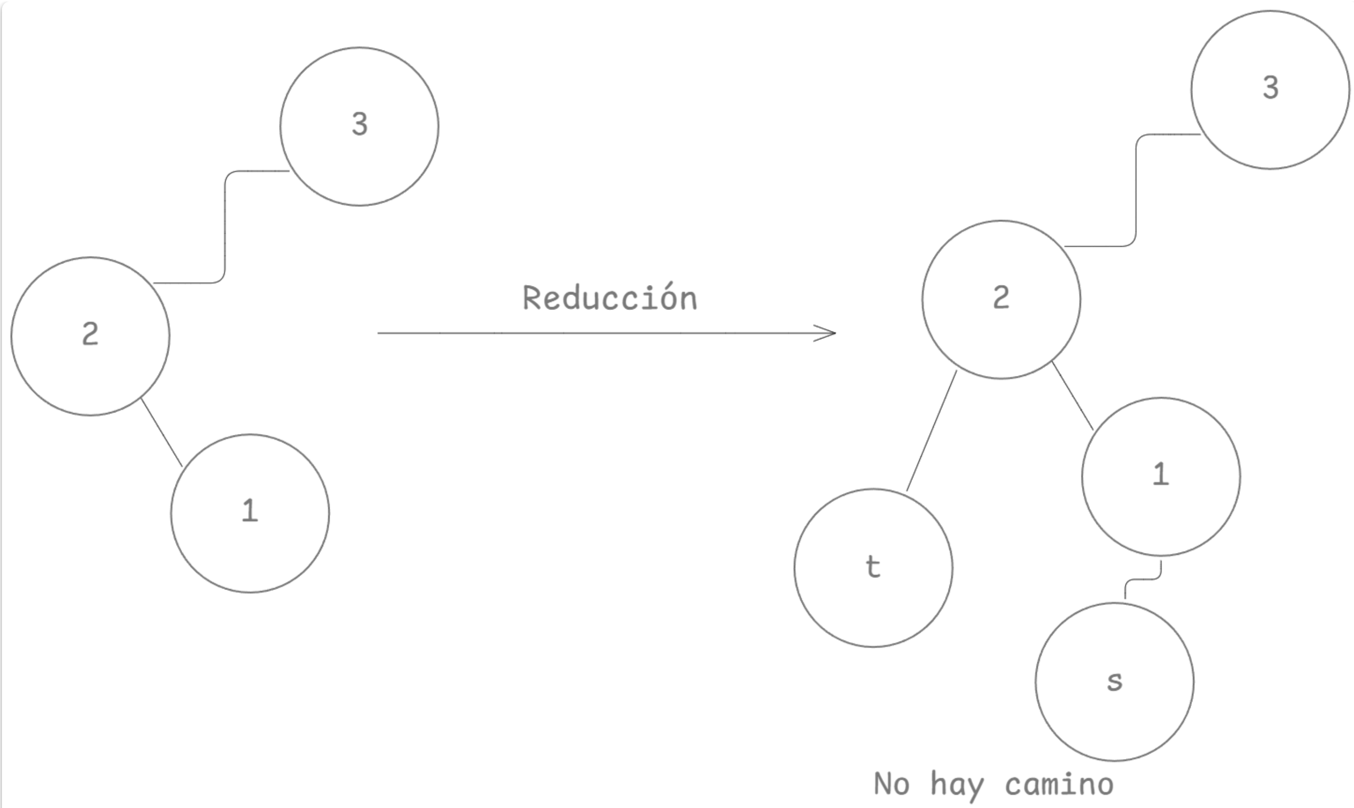
- Agregar 2 vértices nuevos a la lista de vértices se hace en *tiempo polinomial*.
- Seleccionar un vértice cualquiera y agregar una nueva arista $(s, \text{vértice elegido})$ a la lista de aristas se hace en *tiempo polinomial*.
- Verificar los vecinos originales del $\textit{vértice elegido}$ y agregar nuevas aristas $(t, \text{vecino del vértice elegido})$ en la lista de aristas se hace en *tiempo polinomial*.

Verificación de la correctitud:

- Si $G(V, E) \in CH \rightarrow f(V, E) \in SH - s - t$:



- Si $G(V, E) \notin CH \rightarrow f(V, E) \notin SH - s - t$:


3
2
1
Reducción
3
2
t
1
s
No hay camino