P, NP, CoNP y Reducciones

Tecnología Digital V: Diseño de Algoritmos
Universidad Torcuato Di Tella

UNIVERSIDAD TORCUATO DI TELLA

Repaso: Conceptos fundamentales

Problema

Es una función definida por un único input y un único output. Se subdividen en problemas de decisión, optimización, búsqueda, etc.

Instancia

Son las instancias del tipo abstracto de datos que se usa para definir el input.

Problema complemento

El complemento de Π es el problema de decisión Π^c que tiene las mismas instancias que Π, tal que las instancias positivas de Π se corresponden con las negativas de Π^c.

Repaso: Clasificación de problemas en clases

Clase P

Formada por todos los problemas polinomiales. Equivalente a la clase de los problemas cuyos complementos son polinomiales.

Clase NP

Formada por todos los problemas cuya certificación positiva es polinomial.

Clase coNP

Formada por todos los problemas cuya certificación negativa es polinomial. Equivalente a la clase de problemas cuyos complementos pertenecen a NP.

Repaso: Reducciones

Reducción de Π_1 a Π_2

Algoritmo (o función computable) f que transforma instancias de Π_1 a instancias de Π_2 de forma tal que, para toda instancia $I \in \Pi_1$, I es positiva para Π_1 si y sólo si $f(I)$ es positiva para Π_2.

Reducción polinomial

Es una reducción que requiere tiempo polinomial.

Notación: $\Pi \leq_p \Pi'$ cuando existe una reducción polinomial de Π a Π'. Informalmente Π no es más difícil que Π'.

Repaso: NP-hard y NP-Completo

NP-hard

Un problema Γ es NP-hard si todo problema Π reduce polinomialmente a Γ.

$$\forall \Pi \in \text{NP}, (\Pi \leq_p \Gamma)$$

NP-Completo

Un problema Γ es NP-Completo si es NP-hard y está en NP.

Ejercicio I: Propiedades de las clases

Demostrar las siguientes propiedades o dar un contraejemplo en caso de que sean falsas:

- $\mathrm{P} \subseteq \mathrm{NP}$
- $\mathrm{P} \subseteq \mathrm{CoNP}$
- $\mathrm{P} = \mathrm{CoP}$. Ayuda: intentar probar $\Pi \in P \Leftrightarrow \Pi^c \in P$
- $\Pi \in \mathrm{NP} \Leftrightarrow \Pi^c \in \mathrm{CoNP}$

Asumir que $\Pi \leq_p \Pi'$, demostrar las siguientes propiedades o dar un contraejemplo:

- $\Pi \in P \Rightarrow \Pi' \in P$
- $\Pi' \in P \Rightarrow \Pi \in P$
- $\Pi \in \mathrm{NP} \Rightarrow \Pi' \in \mathrm{NP}$
- $\Pi' \in \mathrm{NP} \Rightarrow \Pi \in \mathrm{NP}$
- $\Pi \in \mathrm{CoNP} \Rightarrow \Pi' \in \mathrm{CoNP}$
- $\Pi' \in \mathrm{CoNP} \Rightarrow \Pi \in \mathrm{CoNP}$

Ejercicio I: Soluciones (segunda parte)

Punto 1:

$\Pi \in P \Rightarrow \Pi' \in P$

Falso: Supongamos que Π = 2-coloreo y Π' = 3-coloreo. Sabemos que $\Pi \in P$ y que $\Pi' \in NP$. Pero, podemos reducir de forma polinomial Π a Π' agregando un nodo conectado a todos forzándolo a tener un color único. Por ende, redujimos polinomialmente un problema polinomial a un problema NP completo.

Punto 2:

$\Pi' \in P \Rightarrow \Pi \in P$

Verdadero: Teniendo en cuenta que $\Pi' \in P$ y que tengo una función f polinomial que reduce Π a Π', puedo aplicarle a una instancia I de Π la funcion f y luego resolver Π'. Como todo es polinomial, tengo una forma de resolver Π componiendo polinomios.

Paréntesis: Halting Problem

Importancia

- Es el problema no decidible más conocido y estudiado.
- No existe ningún algoritmo que pueda resolver este problema para todos los casos.
- Fundamental en teoría de la computación y lógica.
- Establece límites fundamentales sobre lo que puede computarse.

Halting Problem

Entrada: Un programa P y una entrada x para ese programa.

Pregunta: ¿El programa P termina su ejecución cuando se ejecuta con la entrada x?

Para saber más recomendamos este video de Computerphile.

Ejercicio I: Soluciones (segunda parte)

Punto 3:

$\Pi \in \text{NP} \Rightarrow \Pi' \in \text{NP}$

Falso: Por más que $\Pi \in NP$, puedo reducirlo polinomialmente a un problema no decidible. El más conocido es halting. Si $\Pi = 3\text{-col} \in NP$ y $\Pi' =$ halting, puedo dar una reducción polinomial de Π a Π'. El programa de entrada de halting va a ser listar todos los posibles 3 coloreos, si encuentra alguno loopear infinitamente y sino frenar. Y su input es el grafo original.

Ejercicio I: Soluciones (segunda parte)

Punto 4:

$\Pi' \in \text{NP} \Rightarrow \Pi \in \text{NP}$

Verdadero: Que $\Pi' \in NP$ significa que existe un certificado y un verificador polinomial para Π'. Luego, dada una instancia I de Π, puedo utilizar como certificado C el certificado de $f(I)$ y escribir el siguiente verificador polinomial, probando que $\Pi \in NP$:

$Checker_{\Pi}(I,C)$:
  if $Checker_{\Pi'}(f(I), C)$ **then return YES**
  else return NO

Donde f es la función de reducción polinomial de Π a Π'.

Punto 5 y 6:

- $\Pi \in \text{CoNP} \Rightarrow \Pi' \in \text{CoNP}$
- $\Pi' \in \text{CoNP} \Rightarrow \Pi \in \text{CoNP}$

Son análogos a los puntos 3 y 4, pero con el complemento de Π y Π'.

Ejercicio II: Problemas de 3-coloreo

Definición de problemas

- **3-coloreo:** Dado G conexo, ¿Se puede particionar $V(G)$ en 3 conjuntos independientes?
- **3-coloreo($\leq k$):** Dado G conexo, ¿Se puede particionar $V(G)$ en tres conjuntos independientes donde algún conjunto tenga a lo sumo k vértices?
- **3-coloreo($\geq k$):** Dado G conexo, ¿Se puede particionar $V(G)$ en 3 conjuntos independientes de tal forma que todos los conjuntos tengan al menos k nodos?

Recordatorio

Un conjunto de nodos de un grafo es independiente cuando no existe una arista en el grafo que conecte a dos nodos del conjunto.

Ejercicio II: Tareas

Sabiendo que 3-coloreo es NPC:

- Proponer certificados y verificadores para los otros dos problemas (probando la pertenencia a NP).
- Dar un algoritmo polinomial para 3-coloreo($\leq k$).
- Demostrar que 3-coloreo($\geq k$) es NPC.

Ejercicio II: Certificados y verificadores

Certificado para 3-coloreo($\leq k$)

- **Certificado:** Un 3-coloreo con algún conjunto que tenga a lo sumo k vértices.
- **Verificador:**
  1. Verificar que es un 3-coloreo válido: $O(n^2)$
  2. Verificar que se usan solo 3 colores: $O(n)$
  3. Verificar que alguna componente tiene $\leq k$ nodos: $O(n)$
- **Complejidad total:** Polinomial

Certificado para 3-coloreo($\geq k$)

- **Certificado:** Un 3-coloreo donde todas las componentes tengan al menos k nodos.
- **Verificador:** Análogo, solo cambia verificar el tamaño de las componentes.

Ejercicio II: Algoritmo polinomial para 3-coloreo($\leq k$)

Observación clave

Podemos iterar sobre todos los subconjuntos de a lo sumo k nodos en tiempo polinomial, dado que el k es un número que está fijo en el problema. Buscar todos los subconjuntos de k elementos es el combinatorio $\binom{n}{k}$, que es menor o igual a n^k.

Algoritmo

1. Para cada subconjunto $V_1 \subseteq V$ con $|V_1| \leq k$:
   - Verificar que V_1 es independiente
   - Verificar que $G \setminus V_1$ es bipartito
2. Si encontramos tal V_1, retornar **SÍ**
3. Si no encontramos ninguno, retornar **NO**

Complejidad

$O(n^k \times (n + m))$, que es polinomial en G (ya que k es constante).

Ejercicio II: Demostrar que 3-coloreo($\geq k$) es NPC

Estrategia

Ya sabemos que 3-coloreo($\geq k$) $\in$ NP. Debemos probar que es NP-hard mostrando que 3-coloreo $\leq_p$ 3-coloreo($\geq k$).

Queremos definir una función f que tome entradas de 3-coloreo (es decir, grafos conexos) y devuelva entradas de 3-coloreo($\geq k$) (una vez más, grafos conexos) tal que G tiene un 3-coloreo si y solamente si $f(G)$ tiene un 3-coloreo donde todas las componentes tienen al menos k nodos.

Ejercicio II: Demostrar que 3-coloreo($\geq k$) es NPC

Función de reducción f

Dado un grafo G (entrada de 3-coloreo):

1. Ejecutar el algoritmo polinomial de 3-coloreo($\leq k$) en G
2. Si encuentra solución: retornar un grafo fijo que sea aceptado por 3-coloreo($\geq k$) (ej: 3 componentes independientes de tamaño k conectadas entre sí)
3. Si no encuentra solución: retornar G

Ejercicio II: Verificación de la reducción

Caso: G es instancia positiva de 3-coloreo

- Si G admite un 3-coloreo con alguna componente $\leq k$: el algoritmo lo detecta y retornamos grafo que acepta 3-coloreo($\geq k$) $\Rightarrow$ **SÍ**
- Si G solo admite 3-coloreos con todas las componentes $> k$: retornamos G, que acepta 3-coloreo($\geq k$) $\Rightarrow$ **SÍ**

Caso: G es instancia negativa de 3-coloreo

- El algoritmo de 3-coloreo($\leq k$) retorna falso
- Retornamos G, que no es 3-coloreable
- Por tanto, no acepta 3-coloreo($\geq k$) $\Rightarrow$ **NO**

Conclusión

La reducción es correcta y polinomial. Por tanto, 3-coloreo($\geq k$) es NPC.

Ejercicio III: Problemas Hamiltonianos

Definición de problemas

- **HAM:** Dado G un grafo no dirigido, ¿G tiene un camino hamiltoniano?
- **HAM-st:** Dados G un grafo no dirigido y s, t dos nodos de G, ¿existe un camino hamiltoniano en G de s a t?

Objetivo

Probar que HAM $\equiv_p$ HAM-st

Ejercicio III: HAM-st $\leq_p$ HAM

Problema

Dados G y nodos s, t, construir H tal que existe camino hamiltoniano de s a t en G si y sólo si H tiene un camino hamiltoniano.

¿Por qué no sirve $H = G$?

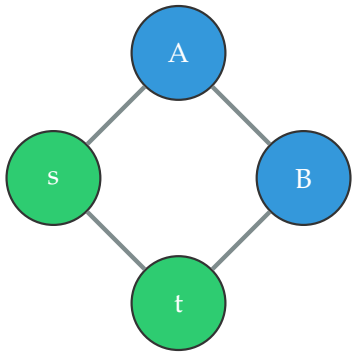
Porque G podría tener caminos hamiltonianos que no vayan de s a t.

Ejercicio III: HAM-st $\leq_p$ HAM

Solución

Para forzar que cualquier camino hamiltoniano en H corresponda a un camino de s a t en G agregamos los nodos s' y t' y los conectamos únicamente a s y t respectivamente.

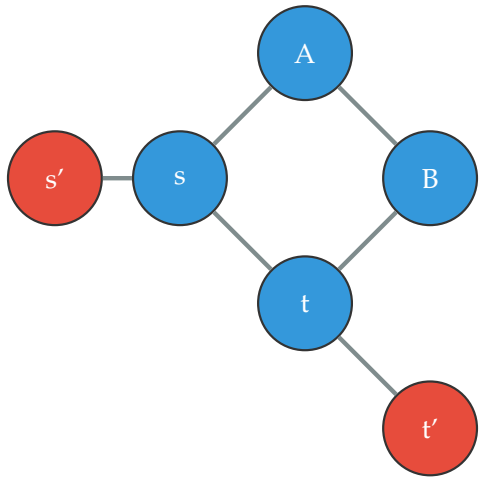
Grafo original G con s,t:



Pregunta: ¿Existe camino hamiltoniano de s a t?

$\Rightarrow$

Grafo construido H:



Pregunta: ¿Existe camino hamiltoniano?

Ejercicio III: HAM $\leq_p$ HAM-st

Problema

Dado H, construir G y elegir nodos s, t tales que H es hamiltoniano si y sólo si G tiene un camino hamiltoniano de s a t.

¿Por qué no sirve $G = H$ con s, t particulares?

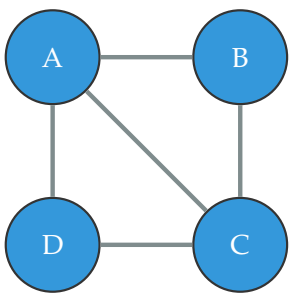
Porque un camino hamiltoniano específico de s a t podría no existir, aunque H sea hamiltoniano.

Ejercicio III: HAM $\leq_p$ HAM-st

Solución

Agregamos los nodos s y t que puedan servir como inicio y final de cualquier camino hamiltoniano en H conectándolos a todos los nodos del grafo.

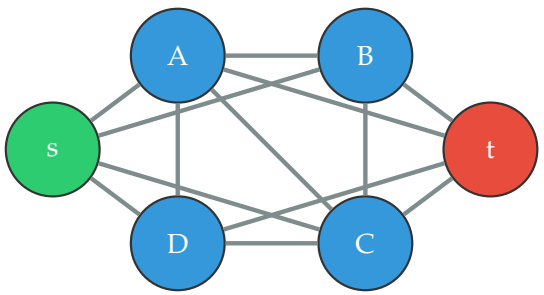
Grafo original H:



Pregunta: ¿Existe camino hamiltoniano?

$\Rightarrow$

Grafo construido G con s,t:



Pregunta: ¿Existe camino hamiltoniano de s a t?