Guía 3

Ej 1:

1. Determinar cuáles de las siguientes afirmaciones son verdaderas y cuáles falsas. Demostrar aquellas que son verdaderas y dar contraejemplos para aquellas que son falsas.

   a) $\mathsf{P} \subseteq \mathsf{NP}$ y $\mathsf{P} \subseteq \mathsf{coNP}$.
   b) Si $\mathsf{P} = \mathsf{NP}$, entonces $\mathsf{coNP} = \mathsf{NP}$.
   c) Si $\mathsf{P} = \mathsf{NP}$, entonces todos los lenguajes pertenecen a P.
   d) Si $\mathsf{coNP} = \mathsf{NP}$, entonces $\mathsf{SAT} \in \mathsf{coNP}$.
   e) Si $\mathsf{coNP} \subseteq \mathsf{NP}$, entonces $\mathsf{NP} = \mathsf{coNP}$.

a) $P \subseteq NP$ y $P \subseteq coNP$ Verdadero

Para ver esto voy a tomar un L genérico tal que $L \in P$ y demostrar que $L \in NP$ y $L \in coNP$.

$P \subseteq NP$:

Tengo una M det. poly. que decide L.

Luego, tomo como certificado un c random (el cual no voy a usar pero tiene tamaño poly) y voy a tener como verificador a M' det. poly. donde se le pase la entrada y c como certificado

M': ⟨x, c⟩

ret M(x) → Es poly

$P \subseteq coNP$:

M decide L.

Luego, uso la misma idea que antes, pero ahora M' es de la forma:

M': ⟨x, c⟩

ret ¬M(x)

Ya que M' sería la máq. que decide $\bar{L}$ tal que $\bar{L} \in NP$.

b) Si $P = NP$, entonces $coNP = NP$ Verdadero

*Notar que sabemos que $P \subseteq NP$ pero no si $NP \subseteq P$

Si tengo que $P = NP$ y sé que $P = coP$, puedo decir que $NP = coP$, pero si en coP por definición tengo todos los complementos de los lenguajes en P (y a su vez los complementos de los lenguajes en NP) obtengo que $coP = coNP$, y como $P = coP$, $P = NP = coP = coNP$.

γ

c) Si $P = NP$, entonces todos los lenguajes pertenecen a P Falso

Sabemos que $ExpTime \notin P$ por jerarquía temporal. Esto hace imposible que todos los lenguajes pertenezcan a P.

Hay muchas formas de encarar esto igual, podría haber dicho que HALTING$\notin$P, también decir que la cantidad de lenguajes es no numerable y la cantidad de máquinas si lo es, etc...

d) Si coNP = NP, entonces SAT ∈ coNP Verdadero

Si coNP = NP, entonces $\forall L.\ L \in NP$ sii $L \in coNP$, en particular SAT está en NP, por lo cual está en coNP.

e) Si coNP ⊆ NP, entonces NP = coNP

Quiero probar que si $coNP \subseteq NP \Rightarrow NP \subseteq coNP$:

Sea $L \in NP$, $\overline{L} \in coNP$, pero como $coNP \subseteq NP$, $\overline{L} \in NP$, entonces $L \in coNP$ por definición de coNP, se encuentran todos los lenguajes que su complemento está en NP.

Ej 2:

2. ¿Es cierto que si dos lenguajes Π y Γ pertenecen a NPC entonces $\Pi \leq_p \Gamma$, y también $\Gamma \leq_p \Pi$? Justificar.

coNP-completo

Sí, es cierto.

Si π y $\Gamma \in$ NP-C, entonces:

$\pi, \Gamma \in NP$ y $\pi, \Gamma \in$ NP-hard ↳ $\forall L.\ L \in NP$, $L \leq_p \pi$ y $L \leq_p \Gamma$

$\pi \leq_p \Gamma$:

$x \in \pi$ sii $f(x) \in \Gamma$, esto se cumple porque $\pi \in NP$ y Γ es NP-hard

$\Gamma \leq_p \pi$:

$x \in \Gamma$ sii $f(x) \in \pi$, esto se cumple porque $\Gamma \in NP$ y π es NP-hard

Ej 3:

3. Sean Π y Γ dos lenguajes tales que $\Pi \leq_p \Gamma$. ¿Qué se puede inferir?

a) Si $\Pi \in$ P entonces $\Gamma \in$ P.
b) Si $\Gamma \in$ P entonces $\Pi \in$ P.
c) Si $\Gamma \in$ NPC entonces $\Pi \in$ NPC.
d) Si $\Pi \in$ NPC entonces $\Gamma \in$ NPC.
e) Si $\Gamma \in$ NPC y $\Pi \in$ NP entonces $\Pi \in$ NPC.
f) Si $\Pi \in$ NPC y $\Gamma \in$ NP entonces $\Gamma \in$ NPC.
g) Π y Γ no pueden pertenecer ambos a NPC.

a) Si $\pi \in P$, entonces $\Gamma \in P$ FALSO

Todo $L \in$ Recursive cumple que $L \leq_p$ HALTING y HALTING no es computable, por lo cual $\notin P$.

b) Si $\Gamma \in P$, entonces $\pi \in P$ Verdadera

Si $\pi \leq_p \Gamma$, entonces hay una f computable polinomialmente tal que:

$$x \in \pi \text{ sii } f(x) \in \Gamma$$

Luego, tengo que la máquina que decide π puede ser aplicar la máquina que decide f (tarda poly en función de x) y luego aplicar la máquina que decide Γ (es poly en función de $f(x)$, pero esta es poly en función de x, así que no hay problema).

c) Si $\Gamma \in NPC$, entonces $\Pi \in NPC$ Falso

Sé que si $\Gamma \in NPC$, entonces $\Pi \in NP$, pero no sé si $\Pi \in NP\text{-hard}$ porque por ejemplo podría suceder que $\Gamma \in P$ y se sigue cumpliendo lo anterior, pero no se sabe si Γ es NP-hard, porque si lo es entonces $P = NP$ y esto es una pregunta abierta.

d) Si $\Pi \in NPC$, entonces $\Gamma \in NPC$ Falso

Es muy similar a la justificación de a)

Γ podría ser HALTING, lo cual es NP-hard, pero no es NP (porque en NP no hay problemas no computables.

e) Si $\Gamma \in NPC$ y $\Pi \in NP$, entonces $\Pi \in NPC$ Falso

No necesariamente, aplica la misma justificación que en el c)

f) Si $\Pi \in NPC$ y $\Gamma \in NP$, entonces $\Gamma \in NPC$ Verdadero

Cubre el error que sucedía en el d)

Si $\Pi \in NPC$ y $\Gamma \in NP$, entonces sucede que $\Gamma \leq_p \Pi$, además como sé que $\Pi \leq_p \Gamma$ puedo decir que:

$$\forall L.\ L \in NP : L \leq_p \Pi \leq_p \Gamma \text{ si\! si } L \leq_p \Gamma$$

Ya que sería la composición de 2 funciones computables polinomialmente y esto ya habíamos visto que era polinomial en lo teórico.

g) Π y Γ no pueden pertenecer ambas a NPC Falso

Si sucede lo que pasa en f ambas pertenecen a NPC. También fue probado en el Ej 2 esto.

Ej 4:

4. Decir si las siguientes afirmaciones son verdaderas o falsas:

   a) Si P = NP, entonces todo problema NP-completo es polinomial.
   b) Si P = NP, entonces todo problema NP-hard es polinomial.
   c) Si las clases NP-completo y coNP-completo son disjuntas entonces P ≠ NP.
   d) HALTING es NP-hard y coNP-hard.

a) Si P=NP, entonces toda problema NP-C es polinomial Verdadera

Si, $\forall L.\ L \in NP \Leftrightarrow L \in P$. O sea, las $L \in NP$ que son NP-hard también se encuentran en P.

b) Si P=NP, entonces toda problema NP-hard es polinomial Falso

HALTING es NP-hard y si P=NP, sigue siendo un lenguaje indecidible, por lo cual no es polinomial y sigue siendo NP-hard.

c) Si NP-C y coNP-C son disjuntas, entonces $P \neq NP$. Verdadera. ($NP \cap coNP = \emptyset$)

Sé que si $P = NP \Rightarrow NP = coNP$. Por esto, si: $NP \neq coNP$ no podría suceder que $P = NP$.

$$S \Rightarrow t \text{ si\! si } \neg S \vee t \text{ si\! si } \neg t \Rightarrow \neg S$$

d) HALTING es NP-hard y coNP-hard Verdadero

En la guía pasada demostramos que si $L \in RECURSIVE \Rightarrow L \leq_p HALTING$ (PZEQ), como se que si $L \in NP$ ó $L \in coNP$, entonces $L \in RECURSIVE$ sé que paso A, o sea, HALTING es NP-hard y coNP-hard.

Ej 5:

5. Suponiendo que P = NP, diseñar un algoritmo polinomial que dada una fórmula booleana ϕ encuentre una asignación que la satizfaga, si es que ϕ es satisfacible.

Si P=NP, entonces tengo una máq. N det. poly. que decide SAT.

Ahora bien, paso a escribir el algoritmo pedido:

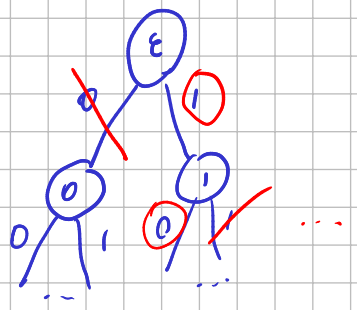
Algoritmo: $\langle \phi \rangle$

```
m := cantidad de var. proposicionales de ϕ        O(|ϕ|)
res := ε.                                          → ε = cadena vacía  O(|ϕ|)

While (m ≠ 0):                                     → Le agrego el 0 a la valuación final   O(|ϕ|) iteraciones
    res.add(0)
    IF (¬N(ϕ|res)):                                → ϕ evaluada en res (no sé si es correcta la notación)
                                                   → Si se vuelve no satisfacible le cambio el último 0 por un 1
        res[|res|-1] = 1

ret res
```

Visual:



Ej 6:

6. Suponiendo que P = NP, diseñar un algoritmo polinomial que dado un grafo G retorne una clique de tamaño máximo de G.

Si P=NP tengo una N máq. det. poly. tal que N decide el problema CLIQUE $\langle G,K \rangle$ de la práctica 2.

Algoritmo: $\langle G \rangle$

```
//Primero busco el tamaño máximo de la clique

K := |V|

while (K ≠ 0)

    IF (N(⟨G,K⟩))                  → Si encuentro un tamaño de clique que es el mayor en G salgo del loop
        break

    K--
```

//Ahora sí busco los nodos de la clique (de manera similar al pto. anterior)

```
i := 1                                   //Iterador sobre los nodos de V   O(|V|)
res := {}                                //Nodos de la clique              O(|V|)
G' := G

while(K≠0):                              //Si no tengo todos los nodos de la clique
  res := res + {V[i]}                    //Agrego otro nodo
  G' := <V'\V[i], E'\{edges de V[i]}>    //Grafo sin el nodo que se está viendo si pertenece a la clique.

  i++

  if(¬N(<G', K>)):                       //Si no existe más la clique máx el que agregue pertenece a la clique máxima.

    K--

  else:                                  //Si sigue existiendo la máxima clique posible, puse uno que podría no estar en el grafo para que exista

    res := res \ {V[i-1]}
```

Ej 7:

7. Sabiendo que CLIQUE es NP-completo, demostrar que SUBGRAPH ISOMORPHISM es NP-completo.

SUBGRAPH ISOMORPHISM: $\{\langle G, H\rangle : G$ es un grafo y H es isomorfo a un subgrafo inducido de $G\}$

CLIQUE: $\{\langle G, K\rangle : G$ es un grafo con subgrafo completo de tamaño $\geq K\}$

Ya demostré en P2E3F que S-I $\in$ NP, ahora QVQ S-I es NP-hard, esto lo logro probando que CLIQUE $\leq_p$ S-I

$x \in$ CLIQUE sii $f(x) \in$ S-I con f computable polinomialmente.

Ahora, ¿qué sería esta f?:

f toma $\langle G, K\rangle$ y declara su salida $(\langle G', H\rangle)$ de manera que $G' = G$ y H = grafo completo de K nodos.

Esta f se computa poly. ya que hacer $G' = G$ toma $O(|V|^2)$ y H toma $O(|V|^2)$.

Luego, tengo que probar que se cumple el sii:

$\Rightarrow$)
Si $\langle G, K\rangle \in$ CLIQUE, entonces existe una clique de tamaño K en el grafo G, luego, hay un grafo isomorfo a esta clique, el cual es un grafo completo de K nodos, o sea, isomorfo a H.

$\Leftarrow$)
Si $\langle G', H\rangle \in$ S-I significa que hay una clique en G' de tamaño $|V_H|$, como sé por construcción que $G' = G$ y $|V_H| = K$, esta significa que $\langle G, K\rangle \in$ CLIQUE.

γ

Ej 8:

8. Considerar los siguientes dos lenguajes:

- SHORTEST PATH (SP) $= \{\langle G, s, t, k\rangle : G$ es un grafo pesado con dos nodos s y t tales que hay un recorrido de s a t de peso menor o igual a $k\}$
- ELEMENTARY SP$= \{\langle G, s, t, k\rangle : G$ es un grafo pesado con dos nodos s y t tales que hay un camino simple de s a t de peso menor o igual a $k\}$

Demostrar que ELEMENTARY SHORTEST PATH es NP-completo y que SHORTEST PATH está en P. ¿Cuál de los dos problemas resuelven los algoritmos de camino mínimo vistos en TDA?

SP $\in$ P: Utilizo Dijkstra ó Bellman-Ford para resolverlo (vistas en TDA), lo cuales poly.

E-SP $\in$ NP-C:

Para esto debo ver q' E-SP $\in$ NP y E-SP $\in$ NP-Hard

E-SP $\in$ NP:

En un rato sigo...