Primer Parcial (1C 2025)

Ayuda: Recordar que $\binom{a}{b} \leq a^b$ → Recordar leer todo el exámen, me bajaron ptos pq' la acoté con cualquier cosa a $\binom{a}{b}$ unu.

Ej 1: Decidir si las siguientes afirmaciones son verdaderas, falsas, o si implican la solución a alguna pregunta abierta. En cualquiera de los casos demostrar.

a) Si $P = NP$, entonces $coNP = NP$
b) $NPSPACE = coNPSPACE$
c) Si $P \neq NP$ entonces cualquier problema en NP q' no esté en P debe ser NP-hard.
d) Todos los problemas en NP (salvo los triviales ($\emptyset$, $\{0,1\}^*$)) son NP-completos.
e) SAT es PSPACE-hard
f) Existe un $K \in \mathbb{N}$ tal que $NP \subseteq NDTime(n^K)$.

Obs: No se da ptos. a respuestas sin justificar.

Ej 1:

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

Si $P = NP$ y sé que $P = coP$, puedo ver que $P = coNP$, porque $coNP = coP$ (asocio 1 a 1 lenguajes de P a NP, los cuales voy complementando y asociando 1 a 1 lenguajes de coP a coNP) por lo cual, $NP = coNP$.

b) $NPSPACE = coNPSPACE$ Verdadero

Como sé que $PSPACE = NPSPACE$ (hay un teo. en la teórica sobre esto).

También sé que $PSPACE = coPSPACE$ (Es la misma M det. con espacio poly y que niega la salida)

Deduzco que $NPSPACE = coNPSPACE$ medio análogamente al punto a)

c) Si $P \neq NP$ entonces cualquier problema en NP q' no esté en P debe ser NP-hard. Falso

Sale la falsedad si planteamos el teorema de Ladner.

Si $P \neq NP \Rightarrow \exists \mathcal{L} . \mathcal{L} \in NP / \mathcal{L} \notin P$ y $\mathcal{L} \notin$ NP-Completo sii (def. NP-C)
$\exists \mathcal{L} . \mathcal{L} \in NP / \mathcal{L} \notin P$ y ($\mathcal{L} \notin NP$ ó $\mathcal{L} \notin$ NP-hard) sii
$\exists \mathcal{L} . \mathcal{L} \in NP / (\mathcal{L} \notin P$ y $\mathcal{L} \notin NP)$ ó ($\mathcal{L} \notin P$ y $\mathcal{L} \notin$ NP-hard) sii (contradicción: $\mathcal{L} \in NP$ y $\mathcal{L} \notin NP$)
$\exists \mathcal{L} . \mathcal{L} \in NP / \mathcal{L} \notin P$ y $\underbrace{\mathcal{L} \notin \text{NP-hard}}_{\text{Contradice el punto c}}$

d) Todos los problemas en NP (salvo los triviales ($\emptyset$, $\{0,1\}^*$)) son NP-completos. Pregunta abierta.

Sabemos q' $P \subseteq NP$, por lo cual, lo anterior implica que si $\mathcal{L} \in P \Rightarrow \mathcal{L} \in NP\text{-}C$, lo cual si lo supiésemos podríamos decir que $P = NP$, ya que reduciríamos cualquier $\pi \in NP$ a cualquier $\mathcal{L} \in P$, o sea que cualquier $\pi \in P$ ($\mathcal{L}$ no trivial).
No sabemos si $P = NP$, por lo que ⓓ es una Preg. abierta.

e) SAT es PSPACE-hard Pregunta abierta.

Si SAT fuese PSPACE-hard podríamos ver que $NP = PSPACE$ y esto no se sabe.
La justificación sale de que podría reducir cualquier problema en PSPACE a NP (por def. de hardness), también ya sé que $NP \subseteq PSPACE$, por lo cual quedaría que $PSPACE = NP$.

f) Existe un $K \in \mathbb{N}$ tal que $NP \subseteq NDTime(n^K)$. Falso

Por el Teorema de jerarquía de tiempos no determinístico.

O sea, sé que si $f(n) = o(g(n))$ con f y g tiempo construibles, entonces sé que

$$NDTime(f(n)) \subsetneq NDTime(g(n))$$

Puedo reemplazar a $f(n)$ por n^K y $g(n)$ por n^{K+1} y como $n^K = o(n^{K+1})$ sé que

$$NDTime(n^K) \subsetneq NDTime(n^{K+1})$$

Pero n^K y n^{K+1} son parte de la definición de $NP = \bigcup_{c \in \mathbb{N}} NDTime(n^c)$.

γ

Ej 2: Enunciar y demostrar el teorema de jerarquía de tiempos determinísticos.

Ej 2:

Sea f, g construibles en tiempo y que $f(n \log n) = o(g(n))$, entonces:

$$DTime(f(n)) \subsetneq DTime(g(n)).$$

Demo:

Construyo D det tal que:

$D: \langle x \rangle$

$t := g(x)$

Correr $U(\langle x, x \rangle)$ por t pasos:

Si no terminó: ret 1

Si no: ret $\neg(U(\langle x, x \rangle))$

Al lenguaje que D decide lo llamo L.

Ahora bien, para probarlo encaro la demo por el absurdo.

Digo que existe $L(M)$ tal que M det, que termina en $f(x)$ pasos, luego tengo que existe M_x por lo visto que hay infinitas escrituras de una máquina que decide el mismo lenguaje.

Luego, si meto a $\langle M_x \rangle$ en D me queda que $D(x) = \neg M_x(x) = \neg M(x)$ Abs!

Ej 3: Considerar los siguientes problemas:

BALANCED-3-SAT = {⟨φ⟩ : φ es una fórmula 3CNF satisfacible por una asignación q' manda tantas variables a falso como a verdadero}

AT-MOST-3-SAT = {⟨φ⟩ : φ es una fórmula 3CNF y hay una asignación q' satisface la fórmula q' manda a lo sumo 10 variables a verdadero}.

Para cada uno decidir si está en P ó si es NP-completo (asumimos q' P≠NP). En cualquiera de los casos, demostrar la afirmación.

Ej 3:

AT-MOST-3-SAT ∈ P → se ve a ojo pq' tiene un un numero fijo de var. q' hay que mandar a verdadero.

Para demostrar su pertenencia a P debo hacer una máq. det. poly que compute AM-3S.

$M: \langle\varphi\rangle$ $\quad \binom{n}{10} \leq n^{10}$ con $n = |\varphi|$

- Genero todas las asignaciones con a lo sumo 10 variables en verdadero y para cada v hago: → $O(n^{10} \cdot n)$
  - $if (v \models \varphi)$: → $O(n)$
    - ret true
  - ret false

Luego, queda mostrar que M decide AM-3S.

Esto se ve ya que M genera todas las asignaciones candidatas a tener 10 variables verdaderas y luego se fija si alguna satisface φ. Si ninguna lo hace, termina de generar todas las posibles y devuelve false.

BALANCED-3-SAT ∈ NP-completo

B-3SAT ∈ NP:

Para ver esto voy a hacer una máq. N det. poly. que use un certificado u de manera que B-3SAT tenga de verificador a N y de certificado a u.

Certificado: Palabra de $\{0,1\}^n$ con $n = |\varphi|$ (para hacer de valuación de φ)

$N: \langle\varphi, u\rangle$

- Verifico que c tenga igual cantidad de ceros que de unos
- Veo que $u \models \varphi$.

B-3SAT ∈ NP-hard:

Busco un problema que sé que es NP-hard tal que lo pueda reducir con una f computable poly. a B-3SAT.

Intuitivamente pienso en $3SAT \leq_p$ B-3SAT.

$x \in 3SAT$ sii $f(x) \in B\text{-}3SAT$

Para que suceda esto necesito q' f transforme a una φ a φ' tal q' si φ era SAT antes ahora φ' tenga una valuación con igual cantidad de 0's que 1's en alguna valuación

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge \dots \wedge (l_{m1} \vee l_{m2} \vee l_{m3}) = \varphi$$

Si tengo:

$$(l'_{11} \vee l'_{12} \vee l'_{13}) \wedge \dots \wedge (l'_{m1} \vee l'_{m2} \vee l'_{m3}) = \phi$$

Siendo que ϕ tiene la misma estructura que φ, con variables frescas y $l_{ij} = \neg l'_{ij}$

O sea, si $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_2) \Rightarrow \phi = (\neg y_1 \vee \neg y_2 \vee \neg y_3) \wedge (\neg y_1 \vee \neg y_2 \vee \neg y_2)$

$\varphi' = \varphi \wedge \phi$

La máq. M det. poly. que uso para computar f es:

M: $\langle \varphi \rangle$

- Genera n variables proposicionales distintas a las que aparecen en φ.
- Itera por φ y crea ϕ de manera que respete la estructura, pero que por cada x_i de φ haya un $\neg y_i$ de ϕ (con x_i las var. prop. de φ y y_i las var. prop. de ϕ)

Ej 4: Dado un lenguaje Π y un número natural $K \in \mathbb{N}$ definimos:

$$\Pi^K = \{x_1 x_2 \cdots x_K : x_i \in \Pi,\ \forall 1 \leq i \leq K\}$$

$$\Pi^* = \bigcup_{K \in \mathbb{N}} \Pi^K$$

(En caso de ser necesario se puede asumir en ⓐ y ⓑ que $\lambda \notin \Pi$, donde λ es la cadena vacía (por más q' el resultado sea independiente a esta suposición)).

a) Probar q' para todo $K \in \mathbb{N}$ vale que si $\Pi \in P$ entonces $\Pi^K \in P$.

b) Probar q' si $\Pi \in NP$, entonces $\Pi^* \in NP$

a)

Si $\Pi \in P$, entonces tenga una máq. M det. poly tal que $\pi(M)$.

Luego, para ver que $\Pi^K \in P$ hago M' det. poly. tal que $\Pi^K(M')$, de la siguiente forma:

$M': \langle X \rangle$

- Hago $\underbrace{\text{todas las posibles particiones}}_{\binom{n}{k} \leq n^K}$ en K palabras y por cada una veo que cada palabra de esta partición $\underbrace{\text{esté en K con M}}$,

$M(Y) \to$ Poly por lo dicho antes ($|Y| \leq |X|$ pq' es una parte de X.

$\to O(Kn^K.\underbrace{poly}_{\text{de la máq. M}})$

O sea, puedo ver todas las particiones posibles de X en en $O(Kn^K)$ $\to$ El K es fija, por lo cual es una cTTe y es poly. con $n = |X|$.
A cada partición le evalúo si todos sus elementos están en Π haciendo $M(x_i)$. $O(\text{poly de}M)$
Si están todos retorna true, si no, repito el proceso con otra partición. $O(1)$
Si ninguna partición retornó true, retorno False. $O(1)$
$\to$ Todo $= O(Kn^K.\text{polyDeM})$.

b)

Si $\Pi \in NP$, entonces:

$$x \in \Pi \quad \text{sii} \quad \exists u \in \{0,1\}^{p(|x|)}.\ M(\langle x,u \rangle) = 1$$

QVQ $\Pi^* \in NP$, o sea que $\bigcup_{K \in \mathbb{N}} \Pi^K \in NP$

Para ver esto qvq existe M' det. poly tal que:

$$y \in \Pi^* \quad \text{sii}\ \exists c \in \{0,1\}^{q(|x|)}.\ M'(\langle y, c \rangle) = 1.$$

Certificado: $\langle U, \ell \rangle$ Donde $U = \{u_1, u_2, \ldots, u_K\}$ donde cada u_i es un certificado para usar con x_i ($\to$ El u puede variar según el x_i), y $\ell = \{s_1, s_2, \ldots, s_K\}$ donde cada s_i es un número que indica el tamaño de la palabra x_i, sirve para dividir x_i.

$|U| = O(K\,p(|x|))$
$|\ell| = O(\log|x|.K)$

$M' : \langle Y, \langle U, l \rangle \rangle$ $\rightarrow O(Kp(|x|) + |Y| + |Y| . polyDeM)$ = Polinomial

- Verifica que U tenga K elementos de tamaño $p(|Y|)$ cada uno $O(Kp(|Y|))$
- Verifico que todos los elementos de l sean > 0, que $l \neq \{\}$ y que $sum(l) = |X|$ $O(|Y|)$
- Particiono Y como indica l y por cada parte Y_i veo que $M(\langle Y_i, u_i \rangle) = 1$ $O(|Y| polyDeM)$
  - Si alguno no lo es retorno falso $O(1)$
- Como toda $Y_i \in \Pi$, entonces, retorno verdadero. $O(1)$

De esta forma tengo una máquina polinomial det. que verifica que $Y \in \Pi^*$ con el certificado $\langle U, l \rangle$ porque es básicamente que hay una partición de z elementos tal que $Y \in \Pi^z$ y si con $\langle U, l \rangle$ ya tengo cómo partir Y y cuáles son los certificados para M tales que me diga si $Y_i \in \Pi$, puedo resolver el resto en tiempo polinomial con M'.

γ