# Teoría de la Computación y Verificación de Programas

Ulises J. Cornejo Fandos[1], Lucas Di Cunzolo[2]

[1] 13566/6, Licenciatura en Informática, Facultad de Informática, UNLP
[2] 13572/5, Licenciatura en Informática, Facultad de Informática, UNLP,

## 1 Ejercicio 1

*Responder y justificar brevemente las siguientes preguntas conceptuales:*

1. *Resolvimos de dos maneras el problema de los palíndromos, una con una MT de 1 cinta y otra con una MT con varias cintas. La primera tarda $O(n^2)$ pasos y la segunda $O(n)$. Al igual que para otros problemas que manifiestan este comportamiento, ¿por qué es indistinta la cantidad de cintas utilizadas, considerando la jerarquía temporal que definimos?*

   Según la jerarquía temporal definida, los problemas se dividen en aquellos que pueden resolverse en tiempo polinomial o exponencial, factibles o no factibles, P o NP. Luego, es indistinto dado que ambas soluciones caen dentro de P.

   ***Regular. Recordar qué ocurre cuando pasamos de modelos de MTs de múltiples cintas a aquellos que tienen una sola cinta, cuál es el retardo en tiempo? influye en la jerarquía temporal que estamos definiendo?***

2. *Vimos que un algoritmo natural para encontrar un divisor que termine en 3 de un número N tarda O(N) pasos. ¿Esto significa que el problema está en P?*

   Para que un problema esté en P, debe existir una MT que lo resuelva en tiempo polinomial. Dado que existe una MT que lo resuelve en O(N) pasos, el mismo está en P.

   ***Revisar, la respuesta no es correcta. Verificar que ocurre con el espacio dependiendo si utilizamos una representación que implica espacio no polinomial. Que ocurre con el tiempo? Sigue siendo polinómico?***

3. *Vimos que utilizando una MTN (MT no determinística), un algoritmo natural para encontrar un circuito de Hamilton en un grafo G tarda $O(n^2)$ pasos.*

*¿Esto significa que el problema está en P?*

No, dado que la simulación de una MTN mediante una MTD tiene un retardo exponencial, por lo que no significa que el problema esté en P.

***Regular. La respuesta está bien, pero la justificación no es el todo completa. Pensar cómo funciona la MT que reconoce CH, el algoritmo completo trabaja en tiempo polinomial?***

4. *El problema de los grafos isomorfos se representa por el lenguaje* $ISO = \{(G_1, G_2) : G_1$ *y* $G_2$ *son grafos isomorfos*$\}$. *Dos grafos son isomorfos si son idénticos salvo por el orden de sus vértices. P.ej. el cuadrado con arcos (1,2), (2,3), (3,4) y (4,1) es isomorfo al cuadrado con arcos (1,2), (2,4), (4,3) y (3,1). Se prueba que* $ISO \in NP$. *Indicar qué certificado sucinto caracteriza al problema ISO.*

   Dado que el output de una MT que resuelva el lenguaje ISO fuesen los 2 grafos etiquetados en sus vértices. El certificado sucinto sería comprobar que el par de vértices con la misma etiqueta en $G_1$ y $G_2$ concuerdan en cuanto a sus vértices adyacentes. $O(|V| * |E|)$.

   ***Regular. Recordar que un certificado es una posible solución. Luego para ver si es sucinto o no hay que ver cuanto ocupa (espacio) y cómo se decide (recordar def. de certificado sucinto).***

5. *Ligado al inciso anterior, considerando ahora el problema de los grafos no isomorfos, ¿cuál sería el certificado asociado? ¿Es suscinto?*

   El certificado es:
   a) Comprobar que tengan la misma cantidad de vértices.
   b) Comprobar que haya la misma cantidad de vértices con el mismo grado en los 2 grafos.
   c) Tomar una permutación de vértices de los grafos y etiquetarlos.
   d) Comprobar por cada par de vértices etiquetados que sus vértices adyacentes sean los mismos. En caso de que todos sean los mismos rechazar.
   e) En caso de que haya alguna diferencia entonces no son isomorfos. Volver al paso *c*.
   f) En caso de que ya no queden permutaciones aceptar.

   En el peor caso se generan $P_{|V_1|+|V_2|} = (|V_1| + |V_2|)!$ permutaciones. Sea, $V = |V_1|+|V_2|$, como $V!$ no tiene la forma $V^k$ con k constante, es decir, el orden no es polinomial, entonces el tiempo es $exp(|V|) <= exp(|G|) = exp(n)$. Por lo que el certificado no seria sucinto.

***Regular. Similar al comentario anterior, en este caso el certificado serían todas las posibles permutaciones, no? No es sucinto porque no es una cadena de tamaño polinomial con respecto al tamaño de la entrada)***

## 2 Ejercicio 2

*Probar que si* $T_1(n) = O(T_2(n))$*, entonces* $TIME(T_1(n))$ *es subconjunto de* $TIME(T_2(n))$*. Ayuda: usar las definiciones estudiadas para probar la inclusión entre los conjuntos indicados.*

Se busca probar que $T_1(n) = O(T_2(n)) \rightarrow TIME(T_1(n)) \subseteq TIME(T_2(n))$.

Sea $L$ un lenguaje cualquiera tal que existe una MT M que lo acepta en tiempo $O(T_1(n))$. Luego $L \in TIME(T_1(n))$, dado que un lenguaje esta en $TIME(T(n))$ si existe una MT que acepte en tiempo $O(T(n))$.

Luego, $(\exists c > 0)(\forall n \in N)(T_1(n) \leq cT_2(n))$, dado que $T_1(n) = O(T_2(n))$ por hipótesis. Entonces el tiempo de $T_1(n)$ está por debajo o es igual al de $c.T2(n)$ para todo $n \in N$, por lo cual podríamos decir que $L \in TIME(T_2(n))$.

Por lo tanto, $TIME(T_1(n)) \subseteq TIME(T_2(n))$.

***Regular. Podemos mejorar la demostración estructurando la prueba de a pasos. Considerando lo que plantea el enunciado lo que debemos probar es*** $TIME(T_1(n)) \subseteq TIME(T_2(n))$***. Es decir, que todo L que está en TIME(T1(n)) también está en T IME(T2(n))***

## 3 Ejercicio 3

*Sea el lenguaje* $SMALL - SAT = \{\varphi | \varphi$ *es una fórmula booleana sin cuantificadores en la forma normal conjuntiva (o FNC), y existe una asignación de valores de verdad que la satisface en la que hay a lo sumo 3 variables con valor de verdad verdadero* $\}$*. Probar que* $SMALL - SAT \in P$*. Comentario:* $\varphi$ *está en la FNC si es una conjunción de disyunciones de variables o variables negadas, como p.ej.* $(x_1 \vee x_2) \wedge x_4 \wedge (\neg x_3 \vee x_5 \vee x_6)$.

Para probar que $SMALL - SAT \in P$ creo una MT M que:

1. Se verifica el input. Comprueba que el input sea una fórmula booleana FNC correcta. $poly(n)$
2. Inicia en 0 un contador en la segunda cinta. $poly(n)$
3. Por cada cláusula separada por $\wedge$:

a) Si leo un símbolo $\neg$ entonces ignoro la variable, ya que no necesito usar uno de mis 3 valores de verdad verdaderos. $poly(n)$
b) Si leo una variable sin $\neg$ entonces incremento en 1 al contador. $poly(n)$
c) Si el contador llega a 4, entonces M rechaza. $poly(n)$
d) Si llego al final de $\varphi$ entonces M acepta. $poly(n)$

***Revisar, la solución no es del todo correcta. Recordar que la MTN tendría que tomar primero 1 variable con valor verdadero (y probar todas las combinaciones), luego dos y finalmente 3. Esto determina la cota de pasos máxima (que es polinomial). Puede ser útil para calcular los tiempos recordar la fórmula de combinatoria (si tengo n variables, primero tomo 1 de n, luego 2 de n y finalmente 3 de n) par ver las asignaciones que deben chequearse.***

## 4 Ejercicio 4

*El problema del conjunto dominante de un grafo se representa por el lenguaje* $DOM - SET = \{(G, K)|G$ *es un grafo que tiene un conjunto dominante de* $K$ *vértices*$\}$. *Un subconjunto de vértices de un grafo* $G$ *es un conjunto dominante de* $G$, *si todo otro vértice de* $G$ *es adyacente a algún vértice de dicho subconjunto. Probar que* $DOM-SET \in NP$. *¿Se cumple que* $DOM-SET \in P$*? ¿Se cumple que* $DOM - SETC \in NP$*? Justificar las respuestas.*

1. Para probar que $DOM - SET \in NP$ debo poder verificar que, sea $D$ el conjunto dominante, es un conjunto dominante de G en tiempo polinomial. Luego,

   a) Chequeo que los vértices de D sean subconjunto de los vértices de G. $O(|V_D| * |V_G|)$
   b) Chequeo que para cada vértice de G se cumpla que algún adyacente pertenezca a D. $O(|V_G| * |E_G|)$

2. La MT natural que aceptaría DOM-SET consiste en:

   a) Validar que el input sea un grafo valido G y un entero válido K.
   b) Tomar K vértices de G.
   c) Chequear con cada vértice de G que se cumpla que algún adyacente pertenezca a los vértices tomados. Si todos chequean verdadero entonces aceptar.
   d) En caso de haber alguno que no pertenezca, si no quedan subconjuntos de G para tomar rechazo.

e) Volver al paso *b*.

En el peor de los casos genera $C_{|V|,K} = \frac{|V|!}{((|V|-K)!K!)}$ subsets de G, por lo que estaría en tiempo exponencial. Luego, $DOM - SET \notin P$.

3. Para probar que $DOM - SETC \in NP$ debo poder verificar que, sea $D$ el conjunto dominante, no es un conjunto dominante de G en tiempo polinomial. Luego,

   a) Se valida que el input sea válido, es decir, que sea un grafo válido y un entero válido.
   b) Tomar K vértices de G.
   c) Chequear con cada vértice de G que se cumpla que algún adyacente pertenezca a los vértices tomados. Si todos chequean verdadero entonces seguir.
   d) En caso de haber alguno que no pertenezca, si no quedan subconjuntos de G para tomar acepto.
   e) Volver al paso *b*.

   En el peor de los casos genera $C_{|V|,K} = \frac{|V|!}{((|V|-K)!K!)}$ subsets de G, por lo que estaría en tiempo exponencial, lo que llevaría a que $DOM - SETC \notin NP$.

## 5 Ejercicio 5

*Sea M una MTN que si acepta una cadena w, al menos en una de sus computaciones lo hace en tiempo polinomial con respecto a —w—, es decir poly(—w—). Probar que $L(M) \in NP$. Ayuda: se sabe que toda función polinomial f(n) es tiempo-construible, es decir que se computa en tiempo f(n).*

Dado una MTN que acepta una cadena w en tiempo polinomial con respecto a $|w|$, para verificar que $L(M) \in NP$, se recorre la rama de la MTN que aceptó w, la cual acepta en tiempo polinomial, por lo que $L(M) \in NP$.

***Revisar, la solución no es correcta. Lo que podemos hacer es "recortar" las computaciones considerando que*** $poly(|w|)$ ***es tiempo-construible. Luego, aún recortando las ramas que tardan más que un tiempo polinómico, sabemos que la MT aceptará cuando deba porque al menos una lo hace en en tiempo polinomial con respecto a*** $|w|$***.***

## 6 Ejercicio 6

*Sea L un lenguaje de cadenas de unos y ceros, sin la cadena vacía $\lambda$. Sea $E(w)$ la cadena espejo de w, que es la que se obtiene reemplazando en w los unos*

*por ceros y los ceros por unos (p.ej. E(1001) = 0110). Se dice que L es un lenguaje espejo si para todo $w \in L$ se cumple $w \in L \leftrightarrow E(w) \in LC$. Construir una reducción polinomial de un lenguaje espejo a su complemento. Comentario: plantear la idea general, definir la función de reducción, probar que la función se computa en tiempo polinomial, y probar que $w \in L \leftrightarrow f(w) \in L^c$.*

**Idea general:**

Para construir una reducción polinomial de un lenguaje espejo $L$ a $L^c$ debo definir una función f en FP que tome la cadena $w$ e invierta los 1 por 0 y 0 por 1.

**Definición de f:**

Sea $f(w) = w'$ tal que $w$ es un cadena válida de 0 y 1, sin la cadena vacía $\lambda$. $f(w)$ se define como una función que recorre el input $w$ mapeando $w_i = \neg w_i, \forall i \in [0, sizeof(w))$.

Luego, f se computa en tiempo polinomial, ya que comprobar que sea una cadena válida es de $O(|w|)$ y el reemplazo de caracteres es de $O(|w|)$.

De esta forma dado un input $w \in L$, al pasarlo la función de reducción w' sería el espejo de w, por lo que $w' \in L^c$.

Si $w \notin L$, al pasarlo por la funcion de reduccion entonces w' sería el espejo de w y $w' \notin L^c$.

## 7 Ejercicio 7

*Sean los lenguajes A y B, tales que $A \neq \emptyset$, $A \neq \Sigma^*$, y $B \in P$. Probar: $(A \bigcap B)\alpha_p A$.*

Se demuestra que dado $A \neq \emptyset$, $A \neq \Sigma^*$, y $B \in P$ entonces $(A \bigcap B)\alpha_p A$.

Sea $f(w) = w'$, f se define de la siguiente forma:

Dado un input w, ejecuta una MT M que acepte el lenguaje B sobre w. Si acepta entonces $w = w'$. Si rechaza entonces le asigna un símbolo válido a w'.

Ya que $B \in P$ entonces existe una MT que lo acepta en tiempo polinomial, por lo que $f \in FP$.

$A\alpha_p A$, pues se cumple la propiedad transitiva sobre la reducción. Luego, el input w no necesita cambiarse en caso de que $w \in B$, pero en caso de no

pertenecer a B, w' debería ser rechazado por A.

## 8 Ejercicio 8

*Se definió en clase el lenguaje $FACT = \{(N, M)|N$ es un número natural y tiene un divisor primo menor que $M\}$, el cual representa el problema de decisión asociado a la factorización. Probar que $FACT \in NP \bigcap CO - NP$. Ayuda: Todo número natural se factoriza de una sola manera (p.ej. 180 = 22 x 32 x 5). Usando esto, y que la longitud de la representación binaria de la factorización de un número es polinomial con respecto a su tamaño, entonces resulta fácil construir una MTN M1 que reconozca FACT en tiempo polinomial y una MTN M2 que reconozca FACTC en tiempo polinomial.*

***Regular. La MTM1 debería, generar una factorización para N y luego que verificar si hay algún factor primo para sea menor a M, no? Luego construimos una MTN M2 que reconozca FACTC en tiempo polinomial.***

Para probar esto, debo demostrar que:

1. $FACT \in NP$

   Para probar esto creare una MTN MF que compruebe que el N sea un número natural y M sea otro numero; genera no deterministicamente los números naturales menores a M; comprueba que el número sea primo, en caso de no serlo rechaza; divide N por el número generado, si el resultado es un número natural entonces acepta, si no lo es entonces rechaza.
2. $FACT \in CO - NP \rightarrow FACTC \in NP$

   Dado que el output de una MT que resuelva FACTC es la factorización del número natural, puedo crear un certificado sucinto buscando entre los números primos de la factorización algún numero menor a M. Dada la existencia de alguno, se rechaza, sino acepta.

## 9 Ejercicio 9

*Se planteó en clase de que si existe $L \in NPC \bigcap CO - NP$ entonces $NP = CO - NP$. Se probó sólo la inclusión $NP \subseteq CO - NP$. Se pide demostrar la otra inclusión, es decir $CO - NP \subseteq NP$.*

Sea $L' \in CO - NP$. Veamos que $L' \in NP$. Se cumple por definición que $L' \alpha_p L$. Luego, $L \in NPC$ y $NPC \subseteq NP$, por hipotesis, entonces $L \in NP$, entonces $L' \in NP$.

***Regular. Quizás pueda servir estructurar la prueba en pasos donde cada paso esté justificado por las hipótesis dadas en el enunciado y definiciones, y donde el último paso sea lo que queremos probar*** $CO - NP \subseteq NP$ ***(en general, lo que estamos probando es que todo L en*** $CO - NP$ ***también está en NP).***

## 10 Ejercicio 10

*Se demuestra que* $NP \subseteq PSPACE$*: una MTN* $M_1$ *que trabaja en tiempo poly(n) se puede simular por una MTD* $M_2$ *que trabaja en espacio poly(n):* $M_2$ *simula una a una las computaciones de* $M_1$ *reusando espacio. Probar que aún calculando el numero de computaciones de* $M_1$ *que aceptan (sumando correspondientemente uno o cero luego de simular cada computación),* $M_2$ *sigue trabajando en espacio poly(n). Ayuda: ¿En qué base debe representarse el contador de* $M_2$*?.*

Sea $k$ la cantidad de ramas de computación que posee la MTN, el número máximo de computaciones aceptadas de $M_1$ sera $k$. Dado que la MTD $M_2$ simula la MTN $M_1$ en tiempo $poly(n)$, entonces la cantidad de ramas de computación de $M_1$ son $poly(n)$, por lo que $k$ seria $poly(n)$. Luego, $M_2$ sigue trabajando en espacio $poly(n)$ aun sumando en base 1 las computaciones aceptadas de $M_1$.

***Regular. Para justificar la prueba propuesta, faltaría mencionar en qué base debe representarse el contador de M2...***