Tiempo polinomial y no polinomial

Comentario: ningún ejercicio reviste mayor dificultad.

Ejercicio 1. Responder breve y claramente los siguientes incisos:

a. ¿Por qué la complejidad temporal sólo trata los lenguajes recursivos?
b. Probar que $n^3 = O(2^n)$. *Ayuda: hay que encontrar un número natural n_0 y una constante $c > 0$ tales que $n^3 \leq c.2^n$ para todo $n \geq n_0$.*
c. Probar que si $T_1(n) = O(T_2(n))$, entonces $TIME(T_1(n)) \subseteq TIME(T_2(n))$. *Ayuda: hay que probar que si un lenguaje L está en $TIME(T_1(n))$, también está en $TIME(T_2(n))$ - recurrir directamente a la definición de orden O de una función T y de clase TIME(T(n)) -.*
d. ¿Cuándo un lenguaje pertenece a P, a NP y a EXP? ¿Por qué si un lenguaje pertenece a P también pertenece a NP y a EXP?
e. ¿Qué formula la Tesis Fuerte de Church-Turing?
f. ¿Por qué es indistinta la cantidad de cintas de las MT que utilizamos para analizar los lenguajes, en el marco de la jerarquía temporal que definimos?
g. ¿Qué codificación de cadenas se descarta en la complejidad temporal?
h. ¿Por qué si un lenguaje L pertenece a P, también su complemento L^C pertenece a P? *Ayuda: hay que probar que si existe una MT M que decide L en tiempo poly(n), también existe una MT M´ que decide L^C en tiempo poly(n).*
i. Sea L un lenguaje de NP. Explicar por qué los certificados de L miden un tamaño polinomial con respecto al tamaño de las cadenas de entrada.

Ejercicio 2. Sea el lenguaje SMALL-SAT = {φ | φ 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: las fórmulas φ en la forma FNC son conjunciones de disyunciones de variables o variables negadas; p.ej. $(x_1 \vee x_2) \wedge x_4 \wedge (\neg x_3 \vee x_5 \vee x_6)$.
Ayuda: una MT que decide SMALL-SAT debe contemplar asignaciones con cero, uno, dos y hasta tres valores de verdad verdadero.

Ejercicio 3. Dados los dos lenguajes siguientes, (1) justificar por qué no estarían en P, (2) probar que están en NP, (3) justificar por qué sus complementos no estarían en NP:

a. *El problema del conjunto dominante de un grafo* consiste en determinar si un grafo no dirigido tiene un conjunto dominante de vértices. Un subconjunto D de vértices de un grafo G es un *conjunto dominante* de G, sii todo vértice de G fuera de D es adyacente a algún vértice de D. El lenguaje que representa el problema es DOM-SET = {(G, K) | G es un grafo no dirigido y tiene un conjunto dominante de K vértices}.
b. *El problema de los grafos isomorfos* consiste en determinar si dos grafos son isomorfos. Dos grafos son *isomorfos* si son idénticos salvo por la denominación de sus arcos. P.ej., dado el grafo G_1 = ({1, 2, 3, 4}, {(1, 2), (2, 3), (3, 4), (4, 1)}), el grafo G_2 = ({1, 2, 3, 4}, {(1, 2), (2, 4), (4, 3). (3, 1)}) es isomorfo a G_1. El lenguaje que representa el problema de los grafos isomorfos es ISO = {(G_1, G_2) | G_1 y G_2 son grafos isomorfos}.

Ejercicio 4. Se prueba que NP $\subseteq$ EXP. La prueba es la siguiente. Si L $\in$ NP, entonces existe una MT M que, para toda cadena de entrada w, verifica en tiempo poly(|w|) si w $\in$ L, con la ayuda de un certificado x tal que $|x| \leq p(|w|)$ - p es un polinomio -, y de esta manera, se puede construir una MT M´ que decida en tiempo exp(|w|) si w $\in$ L, sin usar ninguna cadena adicional: M´ simplemente barre todos y cada uno de los certificados posibles x de w. Se pide explicar por qué M´ efectivamente tarda tiempo exp(|w|). *Ayuda: como $|x| \leq p(|w|)$ y los símbolos de x pertenecen a un alfabeto de k símbolos, ¿cuántos certificados x de tamaño p(|w|) tiene a lo sumo una cadena w?*