Modelos de Computación y Complejidad

(4º curso del Grado en Ingeniería Informática. Tecnologías Informáticas)

Algunos ejemplos de Ejercicios del EXAMEN ONLINE INDIVIDUAL

PARTE TERCERA : Ejercicios

1. Diseñar un programa GOTO sin usar macros (salvo GOTO L), que calcule la función total f de $\mathbb{N}$ en $\mathbb{N}$ definida por: $f(x) = 2x$, para cada $x \in \mathbb{N}$.

2. Diseñar un programa GOTO que verifique, simultáneamente, las condiciones siguientes: (a) contiene, al menos, 2 instrucciones de tipo condicional; (b) **toda computación** del programa realiza, exactamente, 5 pasos de transición; y (c) **no calcula** una **función constante**. Justifíquese la respuesta.

3. Diseñar un programa GOTO sin usar macros (salvo GOTO L), que calcule la función parcial f de $\mathbb{N}$ en $\mathbb{N}$ definida por: $f(x) = x + 1$, si $x \in \mathbb{N}$ es un número impar y, caso contrario, no está definida.

4. Diseñar un programa GOTO sin usar macros (salvo GOTO L), que calcule el siguiente predicado binario: $\theta(x_1, x_2) = 1$ si y sólo si $x_1 = x_2$.

5. Diseñar un programa GOTO sin usar macros (salvo GOTO L), que calcule el siguiente predicado binario: $\theta(x_1, x_2) = 1$ si y sólo si $x_1 \leq x_2$.

6. Diseñar un programa GOTO sin usar macros (salvo GOTO L), que calcule el siguiente predicado binario: $\theta(x_1, x_2) = 1$ si y sólo si $x_1 < x_2$.

7. Diseñar programa GOTO sin usar macros (salvo GOTO L) que calcule la función total f de $\mathbb{N}$ en $\mathbb{N}$ caracterizada por las condiciones siguientes:

$$\begin{cases} f(0) & = & 1 \\ f(x+1) & = & f(x) + 2 \end{cases}$$

8. Diseñar un programa GOTO tal que contenga, al menos, 2 instrucciones de tipo condicional y, además, **toda computación** del programa realice, exactamente, 5 pasos de transición. Justifíquese la respuesta.

9. Diseñar un programa GOTO que verifique, simultáneamente, las condiciones siguientes: (a) contiene **exactamente 6 instrucciones**; (b) contiene **exactamente** 3 instrucciones de tipo **condicional**; y (c) toda computación del programa realiza, exactamente, 5 pasos de transición. Justifíquese la respuesta.

10. Diseñar un programa GOTO que verifique, simultáneamente, las condiciones siguientes: (a) contiene **exactamente 5 instrucciones**; (b) contiene **exactamente** 4 instrucciones de tipo **condicional**; y (c) toda computación del programa realiza, exactamente, 4 pasos de transición. Justifíquese la respuesta.

11. Diseñar un programa GOTO que verifique, simultáneamente, las condiciones siguientes: (a) contiene **exactamente 6 instrucciones**; (b) contiene **alguna** instrucción de tipo **condicional**; y (c) toda computación del programa realiza, exactamente, 4 pasos de transición. Justifíquese la respuesta.

12. Demostrar que la función factorial ($f(n) = n!$, $\forall n \in \mathbb{N}$) es GOTO-computable.

13. Sea función $f : \mathbb{N} - \rightarrow, \mathbb{N}$ definida por $f(n) = 5$, si $\varphi_n^{(1)}(n) \uparrow$, y $f(n) = \uparrow$, si $\varphi_n^{(1)}(n) \downarrow$. Demostrar, por reducción al absurdo, que la función f **no es** `GOTO`-computable.

14. Probar que es `GOTO`-computable la función total f de $\mathbb{N}$ en $\mathbb{N}$ definida como sigue: $f(0) = 0$ y $f(x) =$ el número de **divisores impares** de x, para cada $x > 0$.

15. Sea P un programa `GOTO` cuyo código es $1 + 10 + \cdots + 10^n$ (para un cierto número natural n). Demostrar que el código de ese programa **no puede coincidir** con el código de una **configuración de parada** de P.

16. Sea P un programa `GOTO` cuyo código es $1 + 2 + \cdots + 2^{45}$ (para un cierto número natural n). Hallar explícitamente ese programa y demostrar que **no existe** ninguna configuración del programa cuyo código **coincida** con el código del propio programa P. ¿Existe algún **estado** cuyo código coincida con el código del programa?

17. Sea P el programa `GOTO` de código 19.999. Se pide: (a) describir explícitamente P; (b) probar que la configuración de código 803 **no es** una configuración de parada de P y hallar el estado asociado a esa configuración (¿es inicial?); y (c) hallar una configuración de parada de P cuyo código sea lo más cercano posible a 803.

18. Sea P el programa `GOTO` de código 69999. Se pide: (a) describir explícitamente P; (b) probar que la configuración de código 239 **no es** una configuración de parada de P y hallar el estado asociado a esa configuración (¿es inicial?); y (c) hallar una configuración de parada de P cuyo código sea lo más cercano posible a 239.

19. Dado el programa `GOTO`, P, de código $153055007 = 2^5 \cdot 3^{14} - 1$, se pide: (a) describir explícitamente P; (b) determinar si la configuración de código 803 es una configuración de parada de P

20. Probar que: (a) es `GOTO`–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* `GOTO` *de código* n *carece de instrucciones del tipo CONDICIONAL*$\}$; y (b) es recursivamente enumerable el conjunto $B = \{\langle x, y\rangle \in \mathbb{N} \mid$ *el programa de código* x **para** *sobre* y *en,* **exactamente**, *un número impar de pasos*$\}$.

21. Probar que: (a) es `GOTO`–computable el conjunto $A = \{n \in \mathbb{N} \mid n$ *es el código de un programa* `GOTO` *tal que todas sus variables son de entrada* $\}$, y (b) es recursivamente enumerable el conjunto $B = \{\langle x, y\rangle \in \mathbb{N} \mid$ *el programa de código* x **para** *sobre* y*, siendo el resultado un número par*$\}$.

22. Probar que: (a) es `GOTO`–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* `GOTO` *de código* n *posee* **alguna** *instrucción* **etiquetada** *a la izquierda*$\}$, y (b) es recursivamente enumerable el conjunto $B = \{\langle x, y\rangle \in \mathbb{N} \mid$ *el programa de código* x **para** *sobre* y *en,* **exactamente**, $x + y$ *pasos y, además, el resultado es un número impar*$\}$.

23. Probar que: (a) es `GOTO`–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* `GOTO` *de código* $2n + 3$ *posee, a lo sumo, 7 instrucciones y alguna de ellas es un DECREMENTO*$\}$, y (b) es recursivamente enumerable el

24. Probar que: (a) es `GOTO`–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* `GOTO` *de código* $7n + 7$ *posee un número par de instrucciones y todas ellas son de tipo CONDICIONAL* $\}$, y (b) es recursivamente enumerable el conjunto $B = \{n \in \mathbb{N} \mid \varphi_n^{(1)}$ **no es la función vacía**$\}$. conjunto $B = \{n \in \mathbb{N} \mid \varphi_n^{(1)}$ **no es inyectiva**$\}$.

25. Probar que: (a) es `GOTO`–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* `GOTO` *de código* n *posee 10 instrucciones y ninguna de ellas es SKIP*$\}$, y (b) es recursivamente enumerable el conjunto $B = \{n \in \mathbb{N} \mid$ *la computación* $\varphi_n^{(1)}(2)$ **para** *y el resultado es* $2\}$.

26. Probar que: (a) es GOTO–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* $n+8$ *posee, a lo sumo, 8 instrucciones*$\}$, y (b) es recursivamente enumerable el conjunto $B = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* $n+8$ **para** *y el resultado es* $n+8\}$.

27. Probar que: (a) es GOTO–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* n^2 **para** *sobre* $n+4$ *en,* **exactamente**, *n pasos* $\}$, y (b) es recursivamente enumerable el conjunto $B = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* n^2 **para** *sobre* $n+4$ *y el resultado es* $n^3\}$.

28. Probar que: (a) es GOTO–computable el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* $3n$ **para** *sobre* $4n$ *en,* **exactamente**, $5n$ *pasos* $\}$, y (b) es recursivamente enumerable el conjunto $B = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* $3n$ **para** *sobre* $4n$ *y el resultado es* $5n\}$.

29. Probar que las siguientes funciones f y g de $\mathbb{N}$ en $\mathbb{N}$ son GOTO–computables:

$$\begin{cases} f(0) &= 0 \\ f(n+1) &= 2+3g(n) \end{cases} \qquad \begin{cases} g(0) &= 0 \\ g(n+1) &= 4+5f(n) \end{cases}$$

Indicación: Considérese la función auxiliar $\psi : \mathbb{N} \to \mathbb{N}$ definida por $\psi(n) = [f(n), g(n)]$, para cada $n \in \mathbb{N}$, y pruébese que ψ está definida por recursión primitiva, a partir de funciones GOTO-computables. Finalmente, teniendo presente que $f(n) = (\psi(n))_1$ y que $g(n) = (\psi(n))_2$, concluir que las funciones f y g son GOTO-computables.

30. Probar que existen funciones con rango finito que **no** son GOTO–computables.

31. Probar que es GOTO–computable la función parcial f de $\mathbb{N}$ en $\mathbb{N}$ definida por:

$$f(x) = \begin{cases} 1 & \text{si } \varphi_x(x) \downarrow \\ \uparrow & \text{si } \varphi_x(x) \uparrow \end{cases}$$

32. Probar que el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* GOTO *de código* $n+5$ *posee, a lo sumo, 5 instrucciones*$\}$ es GOTO–computable y que, en cambio, el conjunto $B = \{\langle x, y\rangle \in \mathbb{N} \mid$ *el programa de código* x *para sobre* y *en, exactamente, un número impar de pasos y el resultado es un número par*$\}$ es recursivamente enumerable.

33. Probar que son recursivamente enumerables los siguientes conjuntos: $A = \{\langle x, y\rangle \in \mathbb{N} \mid$ *el programa de código* $2x$ **para** *sobre* $3y$ *en,* **exactamente**, *un número impar de pasos y, además, el resultado es un número primo* $\}$ y $B = \{\langle x, y\rangle \in \mathbb{N} \mid$ *el programa de código* $x+y$ **para** *sobre* $x+2y$, *siendo el resultado un número par*$\}$.

34. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid \varphi_n^{(1)}$ *es una aplicación sobreyectiva* $\}$ **no** es GOTO–computable. ¿Es recursivamente enumerable?

35. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid \varphi_n^{(1)}$ *es una función constante de aridad 1*$\}$ **no** es GOTO–computable.

36. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid \varphi_n^{(1)}$ **no** *es un predicado*$\}$ **no** es GOTO–computable.

37. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid \varphi_n^{(1)}$ **no** *es inyectiva*$\}$ **no** es GOTO–computable. Demostrar que, en cambio, el conjunto A **sí** es recursivamente enumerable. ¿Qué se puede deducir acerca del conjunto $\overline{A}$?

38. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid \forall x \in \mathbb{N}\ (\varphi_n^{(1)}(x) = x!)\}$ **no** es GOTO–computable.

39. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid$ *el programa* `GOTO` *de código* n **no calcula** *la función identidad en* $\mathbb{N}$, **ni** *la aplicación vacía*$\}$ **no** es `GOTO`–computable. Demostrar que, en cambio, el conjunto A **sí** es recursivamente enumerable. ¿Qué se puede deducir acerca del conjunto $\overline{A}$?

40. Aplicando el teorema de Rice, probar que el conjunto $A = \{n \in \mathbb{N} \mid \forall x \in \mathbb{N}\ (\varphi_n^{(1)}(x)$ **no** *es una potencia de* $5)\}$ **no** es `GOTO`–computable. Demostrar que, en cambio, el conjunto A **sí** es recursivamente enumerable. ¿Qué se puede deducir acerca del conjunto $\overline{A}$?

41. Probar que es **indecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si el dominio de la función* $\varphi_n^{(1)}$ *es un conjunto* `GOTO`*-computable"*

    ¿Es **semidecidible** dicho problema?

42. Probar que es **indecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si existe un programa* `GOTO` *sin instrucciones condicionales que calcula la función* $\varphi_n^{(1)}$*"*

    ¿Es **semidecidible** dicho problema?

43. Probar que es **indecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la función* $\varphi_n^{(1)}$ *se puede calcular mediante un programa* `GOTO` *que sólo contiene instrucciones* **decremento***"*

    ¿Es **semidecidible** dicho problema?

44. Probar que es **indecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la función* $\varphi_n^{(1)}$ *se puede calcular mediante un programa* `GOTO` *que sólo contiene instrucciones* **decremento***"*

    ¿Es **semidecidible** dicho problema?

45. Probar que es **indecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la función* $\varphi_n^{(1)}$ *se puede calcular mediante un programa* `GOTO` *que sólo contiene instrucciones* **condicional***"*

    ¿Es **semidecidible** dicho problema?

46. Probar que es **indecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la función* $\varphi_n^{(1)}$ *se puede calcular mediante un programa* `GOTO` *que sólo contiene instrucciones* **skip***"* ¿Es **semidecidible** dicho problema?

47. Consideremos los siguientes problemas de decisión:

    $X \equiv$ *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la función* $\varphi_n^{(1)}$ **no es** *inyectiva"*
    $Y \equiv$ *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la función* $\varphi_n^{(1)}$ **sí es** *inyectiva"*

    Demostrar que el problema X es indecidible pero que, en cambio, **sí es** *semidecidible*.

    Demostrar que el problema Y es indecidible e, incluso, **no es** *semidecidible*.

48. Probar que **no es semidecidible** el siguiente problema de decisión: *"Dado un número natural* $n \in \mathbb{N}$, *determinar si la computación del programa* `GOTO` *de código* n^2, *con dato de entrada* $3n$, **no es de parada***"*

49. Diseñar una MTD, M, cuyo alfabeto de trabajo sea $\{0, 1, B, \triangleright\}$ y que decida el lenguaje $L = \{u = \{0,1\}^+ \mid u$ representa un número impar$\}$. Hállese el coste en tiempo de la máquina diseñada.

50. Diseñar una MTD, M, cuyo alfabeto de trabajo sea $\{0, 1, B, \triangleright\}$ y que decida el lenguaje $L = \{u \in \{0,1\}^+ \mid$ la logitud de u es 4 $\}$. Hállese el coste en tiempo de la máquina diseñada.

51. Diseñar una MTD, M, cuyo alfabeto de trabajo sea $\{0, 1, B, \triangleright\}$ de tal manera que:

    (a) Calcule la función constante e igual a 3 (de aridad 1).

    (b) Para cada dato de entrada, realice el menor número posible de transiciones (es decir, pasos de computación).

    Hallar el coste en tiempo de la máquina diseñada y justifíquese que la máquina diseñada satisface los requisitos exigidos.

52. Diseñar una MTD cuyos datos de entrada sean números naturales expresados en binario y de tal manera que calcule la función idénticamente nula (de aridad 1). Ilustrar el diseño realizado, considerando **dos** datos de entrada de tamaño 3 y detallando las correspondientes computaciones.

53. Diseñar una MTD cuyos datos de entrada sean números naturales expresados en binario y de tal manera que calcule la función constante e igual a 2 (de aridad 1). Ilustrar el diseño realizado, considerando **dos** datos de entrada de tamaño 3 y detallando las correspondientes computaciones.

54. Diseñar una MTD cuyos datos de entrada sean números naturales expresados en binario y de tal manera que: (a) si el dato de entrada es **par**, entonces **para y devuelve 1**; y (b) si es **impar** entonces **no para**. Ilustrar el diseño realizado, considerando **dos** datos de entrada de tamaño 3 y detallando las correspondientes computaciones.

55. Sean $\{f_e \mid e \in \mathbb{N}\}$ y $\{g_e \mid e \in \mathbb{N}\}$ dos medidas de complejidad. Consideremos la sucesión infinita de funciones 1-arias sobre $\mathbb{N}$, $\{h_e \mid e \in \mathbb{N}\}$, definida como sigue: $h_e(x) = f_e(x) + g_e(x)$, para cada $x \in \mathbb{N}$. Demostrar que la sucesión $\{h_e \mid e \in \mathbb{N}\}$ también es una medida de complejidad.

56. EL problema **TRIÁNGULO** es el siguiente: *"Dado un grafo no dirigido, determinar si contiene algún triángulo (subgrafo completo de tamaño 3)".*

    Probar que dicho problema pertenece a la clase de complejidad **P**.

57. Dado un conjunto finito no vacío A, una **función peso** w sobre A es una aplicación de $\mathbf{P}(A)$ en $\mathbb{N}$ que verifica la siguiente propiedad: para cada $B \subseteq A$, $w(B) = \sum_{x \in B} w(\{x\})$ (es decir, el peso de B es la suma de los pesos de sus elementos).

    EL problema **SUBSET-SUM** es el siguiente: *"Dado un un conjunto finito no vacío A, una función peso w sobre A y un número natural k, determinar si existe un subconjunto $B \subseteq A$ tal que $w(B) = k$"*

    Demostrar que el problema **SUBSET-SUM** pertenece a la clase de complejidad **NP**.

58. EL problema **PARTITION** es el siguiente: *"Dado un un conjunto finito no vacío A y una función peso w sobre A, determinar si existe una partición $\{B, C\}$ de A tal que $w(B) = w(C)$"*

    Demostrar que el problema **PARTITION** pertenece a la clase de complejidad **NP**.

59. Sean X_1, X_2, X_3 problemas de decisión tales que X_1 es reducible en tiempo polinomial a X_2 y X_2 es reducible en tiempo polinomial a X_3. Demostrar que el problema X_1 es reducible en tiempo polinomial al problema X_3.

60. Si $X = (\Sigma_X, E_X, \theta_X)$ es un problema de decisión, entonces el **problema complementario** de X es $\overline{X} = (\Sigma_X, E_X, \neg\theta_X)$. Si $\mathcal{C}$ es una clase de complejidad entonces la **clase complementaria** de $\mathcal{C}$ es **co**–$\mathcal{C} = \{X \mid \overline{X} \in \mathcal{C}\}$. Demostrar que: **P** = **co-P** y que **P** $\subseteq$ **co-NP**.