EJEMPLOS DE PROBLEMAS DE LA CLASE P

Ejemplo 1. El problema del camino mínimo en un grafo está en P

El problema (de decisión) del camino mínimo en un grafo consiste en determinar si en un grafo existe un camino entre dos vértices v_1 y v_2, de longitud a lo sumo K.
Un grafo se representará por un par (V, E) como se describió previamente, utilizando números en binario para la identificación de los vértices, y el símbolo # como separador.
La idea general del algoritmo propuesto se basa en lo siguiente. Si $A^h(i, j)$ es la longitud del camino mínimo entre los vértices i y j que no pasa por ningún vértice mayor que h, entonces se cumple

$$A^{h+1}(i, j) = \min(A^h(i, h+1) + A^h(h+1, j), A^h(i, j))$$

Naturalmente, el camino mínimo entre i y j que no pasa por ningún vértice mayor que h+1, pasa o no pasa por el vértice h+1. La siguiente MTD M, basada en la igualdad anterior, trabaja en tiempo polinomial y reconoce el lenguaje SP (por *shortest path* o camino mínimo) que representa el problema, siendo SP = {(G, v_1, v_2, K) | G es un grafo y tiene un camino entre sus vértices v_1 y v_2 de longitud a lo sumo K}. Dada una entrada w = (G, v_1, v_2, K), M obtiene $A^m(v_1, v_2)$, el camino mínimo en G entre v_1 y v_2, y acepta si y sólo si $A^m(v_1, v_2) \leq K$. Se utilizan matrices A^i de m x m para almacenar los valores que se van calculando:

1. Si w no es una entrada válida, rechaza.
2. Para todo i, j $\leq$ m, hace:
   Si G incluye un arco (i, j), entonces $A^1[i, j] := 1$, si no $A^1[i, j] := m$.
3. Para todo h = 2, 3, …, m, hace:
   Para todo i, j $\leq$ m, hace:
   $A^h[i, j] := \min(A^{h-1}[i, h] + A^{h-1}[h, j], A^{h-1}[i, j])$.
4. Si $A^m[v_1, v_2] \leq K$, entonces acepta, si no rechaza.

En el paso 1 hay que verificar fundamentalmente que G es válido (los vértices de V son 1, …, m, los arcos de E no se repiten y sus vértices están en V), lo que lleva tiempo O(|V|) + O(|E|^2) + O(|V||E|), y que v_1 y v_2 son vértices distintos válidos y K un número natural menor que m, lo que lleva tiempo lineal. Así, el tiempo consumido por este paso es O(n^2). La asignación A^1[i, j] := m en el paso 2 significa que A^1[i, j] recibe un valor muy grande, seguro que mayor que la longitud del camino mínimo. Claramente, el tiempo consumido por los pasos 2 a 4 es O(m^3) = O(n^3). Por lo tanto, la MTD M hace O(n^2) + O(n^3) = O(n^3) pasos. Queda como ejercicio probar que efectivamente L(M) = SP.

Ejemplo 2. El problema del máximo común divisor está en FP

El máximo común divisor de dos números naturales *a* y b, denotado con mcd(a, b), es el máximo número natural que divide a los dos. Por ejemplo, mcd(30, 24) = 6. Se cumple que si r es el resto de la división de *a* sobre b, es decir si r = a mod b, con a ≥ b, entonces

mcd(a, b) = mcd(b, r)

En base a esta idea se presenta la siguiente MTD M, que trabaja en tiempo polinomial. M calcula en la variable x el valor mcd(a, b), con a ≥ b:

1. Si b = 0, entonces hace x := a, y acepta.
2. Hace r := a mod b.
3. Hace a := b.
4. Hace b := r.
5. Vuelve al paso 1.

Veamos que la MT M itera a lo sumo log b veces. Sean (a_{k-1}, b_{k-1}), (a_k, b_k), (a_{k+1}, b_{k+1}), tres pares sucesivos calculados por M:

- Se cumple $a_k = q.b_k + b_{k+1}$, para algún q ≥ 1. Por lo tanto, $a_k \geq b_k + b_{k+1}$.
- Como $b_{k-1} = a_k$, entonces $b_{k-1} \geq b_k + b_{k+1}$.

- A partir de lo anterior se puede probar que $b = b_0 \geq 2^{k/2}b_k$, para todo número natural par $k \geq 2$. Esto significa que k, que representa el número de pasos de la MT M, está acotado por $\log_2 b$.

Así, M trabaja en tiempo determinístico lineal con respecto a la longitud de sus entradas. Queda como ejercicio probar que efectivamente M calcula en x el valor mcd(a, b).

Ejemplo 7.3. El problema 2-SAT está en P

El problema 2-SAT consiste en determinar si una fórmula booleana φ (con una sintaxis determinada que enseguida especificamos) es satisfactible, es decir, si existe una asignación de valores de verdad para φ que la hace verdadera. Una fórmula booleana se define inductivamente de la siguiente manera:

1. Una variable x es una fórmula booleana.
2. Si φ_1 y φ_2 son fórmulas booleanas, también lo son $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \wedge \varphi_2)$ y $\neg\varphi_1$. Los paréntesis redundantes pueden omitirse.

Si una fórmula boolana φ es una conjunción de disyunciones de *literales*, siendo un literal una variable o una variable negada, se dice que φ tiene o está en la *forma normal conjuntiva*. Es el caso, por ejemplo, de

$$\varphi = (x_1 \vee x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_2) \wedge (x_1 \vee x_7 \vee x_5)$$

Las disyunciones se denominan *cláusulas*. En particular, 2-SAT considera sólo fórmulas booleanas en la forma normal conjuntiva con dos literales por cláusula. El lenguaje que representa el problema es 2-SAT = {φ | φ es una fórmula booleana satisfactible, en la forma normal conjuntiva, con dos literales por cláusula}. Para probar que 2-SAT está en P, vamos a construir una MTD M que lo reconoce en tiempo polinomial. La idea general es la siguiente. M empieza asignando arbitrariamente el valor verdadero a alguna variable x, completa consistentemente todas las asignaciones que puede, barre una a una las cláusulas c

= ($\varphi_1 \vee \varphi_2$) con al menos un literal asignado, y las procesa adecuadamente (por ejemplo, si uno de los dos literales tiene el valor verdadero, declara satisfecha a la cláusula). Si no rechaza por detectar la instatisfactibilidad de la fórmula completa, M repite el proceso a partir de otra asignación arbitraria a alguna variable x, hasta decidir que la fórmula está o no en 2-SAT. Formalmente, dada una entrada φ, el conjunto C de cláusulas (al comienzo todas declaradas insatisfechas), y el conjunto V de variables (al comienzo todas declaradas no asignadas), la MT M hace:

1. Si φ no es una entrada válida sintácticamente, rechaza.
2. Si el conjunto V está vacío, acepta.
3. Dada una variable x de V, hace x := verdadero.
   Hace primer-valor := verdadero.
   Mientras C tenga una cláusula c = ($\varphi_1 \vee \varphi_2$) insatisfecha con al menos un literal asignado, hace:
      Si φ_1 = verdadero, o bien φ_2 = verdadero, entonces declara a la cláusula c satisfecha.
      Si no, si φ_1 = falso, y también φ_2 = falso, entonces:
         Si primer-valor $\neq$ verdadero, rechaza.
         Si no, declara insatisfechas a todas las cláusulas de C
            declara no asignadas a todas las variables de V
            hace x := falso
            hace primer-valor := falso.
      Si no, si φ_1 = falso, entonces hace φ_2 := verdadero
      Si no, entonces hace φ_1 := verdadero.
   Elimina de C las cláusulas satisfechas, y de V las variables asignadas.
4. Vuelve al paso 2.

El análisis sintáctico de la fórmula φ en el paso 1 es lineal, y el tiempo consumido por los pasos 2 a 4 es $O(|V||C|) = O(n^2)$. Por lo tanto, la MTD M trabaja en tiempo $O(n^2)$. Queda como ejercicio probar que efectivamente L(M) = 2-SAT.