UNIVERSIDAD DE CONCEPCION
FACULTAD DE CIENCIAS
FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA MATEMATICA

Optimización III (525551)

Ejercicios de complejidad temporal

P1) La operación de reducción polinomial $\leq_p$ induce una relación en los problemas combinatoriales de decisión, i.e. dos problemas de decisión Q y R están relacionados si $Q \leq_p R$.

a) Pruebe que la relación inducida por $\leq_p$ es refleja y transitiva.

b) Muestre que a relación inducida por $\leq_p$ no es antisimétrica.

c) Pruebe que si la relación inducida por $\leq_p$ es simétrica, entonces P=NP.

P2) Sea Q y R problemas combinatoriales de decisión. Pruebe que:

a) $Q \in P \iff \overline{Q} \in P$.

b) $Q \leq_p R \iff \overline{Q} \leq_p \overline{R}$.

c) $(Q \leq_p R) \ \wedge \ (R \in \text{NP}) \Longrightarrow Q \in \text{NP}$.

d) Todos los lenguajes NP-completos están en P o ninguno de ellos lo está.

P3) Pruebe, a partir de resultados vistos en clase, que los problemas CLIQUE y VERTEX-COVER son ambos NP-completos.

P4) Se sabe que el siguiente problema en grafos dirigidos es NP-hard.
HAMPATH: Dado G un grafo dirigido ¿Tiene G un camino Hamiltoniano?

a) Muestre que HAMPATH es NP-Completo. Concluya que HAMILTONIAN en grafos dirigidos es también NP-Completo.

b) Muestre que NP=co-NP si y sólo si HAMPATH es co-NP.

P5) Muestre que los siguientes problemas son NP-Completos.

a) **TOUR CON RESTRICCIÓN**: Dado $k \in \mathbb{N}$ y $C = \{c_1, \ldots, c_n\}$ un conjunto de $n \in \mathbb{N}$ ciudades diferentes donde las distancia $d(c_i, c_j)$ del trayecto (no necesariamente simétrica) entre cualquier par de ellas es conocida.
¿Es posible definir un orden $c_{i_1}, c_{i_2}, \ldots, c_{i_n}$ de todas las ciudades de C para visitarlas de manera que la distancia del trayecto entre dos ciudades consecutivas a visitar sea menor o igual a k?

b) **FEEDBACK VERTEX SET**: Dado $G = (V, A)$ un grafo dirigido y $k \in \mathbb{N}$ ¿Existe $V' \subseteq V$ un feedback vertex set (FVS) de G, i.e. un subconjunto de vértices tal que todo ciclo dirigido de G contiene algún vértice de V', con $|V'| \leq k$?

Solución:

P1) *a*) *Refleja*: Sea Q un problema de decisión. Definamos $f : I_Q \to I_Q$ la función identidad dada por: $\forall x \in I_Q,\ f(x) = x$. Luego, se tiene que:

$$\forall x \in I_Q,\ Q(x) = s \iff Q(f(x)) = Q(x) = s.$$

Además, f puede ser calculada por el siguiente algoritmo A:

Algorithm A(x)
Input: $x \in I_Q$
1: **return** x

Como A ejecuta una sola operación elemental, entonces A es $O(1)$ y por lo tanto polinomial en el tamaño de la entrada. De aquí, f es una reducción polinomial y así $Q \leq_p Q$.

Transitividad: Sea Q, R y S problemas de decisión tal que $Q \leq_p R$ y $R \leq_p S$. Luego, existen $f : I_Q \to I_R$ y $g : I_R \to I_S$ reducciones polinomiales calculadas por algoritmos polinomiales A_f y A_g respectivamente. Luego, definamos la función $h : I_Q \to I_S$ por:

$$\forall x \in I_Q,\ h(x) = g(f(x)).$$

Así se tiene que:

$$\forall x \in I_Q :\ S(h(x)) = s \iff S(g(f(x))) = s \iff R(f(x)) = s \iff Q(x) = s.$$

Además, h puede ser calculada por el siguiente algoritmo:

Algorithm A'(x)
Input: $x \in I_Q$
1: $y \leftarrow A_f(x)$
2: $z \leftarrow A_g(y)$
3: **return** z

Como A_f y A_g son algoritmos polinomiales, entonces $\exists r, s \in \mathbb{N}$, tal que el tiempo de ejecución de A_f y A_g son $O(size(x)^r)$ y $O(size(y)^s)$ respectivamente, donde $x \in I_Q$ e $y \in I_R$. Por otro lado, si $y = f(x)$, entonces $size(y) = O(size(x)^t)$ para algún $t \in \mathbb{N}$ pues de lo contrario sólo escribir y tomaría un tiempo no polinomial. Luego, el tiempo de ejecución de A' es: $O(size(x)^r) + O(size(x)^{ts}) = O(size(x)^r + size(x)^{ts})$. Como $size(x)^r + size(x)^{ts} = O(size(x)^{r+ts})$, entonces el tiempo de ejecución de A' es polinomial en el tamaño de $x \in I_Q$. De aquí, A' es un algoritmo polinomial que calcula h y por lo tanto $Q \leq_p S$.

b) Para mostrar que la relación inducida por $\leq_p$ no es antisimétrica basta considerar dos problemas NP-completos distintos: Q y R (por ejemplo: CLIQUE y SAT).
Como ambos son NP-hard se tiene que:

$$\forall S \in \text{NP}, \quad S \leq_p Q\ \wedge\ S \leq R.$$

Además, Q y R son NP. Luego se tiene en particular que: $R \leq_p Q$ y $Q \leq_p R$ con $Q \neq R$. Por consiguiente, la relación no es antisimétrica.

c) Sea $Q \in$ P y R es NP-hard. Como P $\subseteq$ NP, entonces $Q \in$ NP y así $Q \leq_p R$. Si la relación inducida por $\leq_p$ fuera simétrica se tendría que $R \leq_p Q$ con $Q \in$ P. Luego, por resultados vistos en clases, $R \in$ P y por lo tanto NP=P.

P2) *a*) ($\Rightarrow$) Supongamos que $Q \in P$. Luego, existe un algoritmo polinomial A con entrada $x \in I_Q$ y salida $y \in \{s, n\}$ que resuelve Q, i. e. $\forall x \in I_Q,\ Q(x) = s \iff A(x) = s$.
Sea $\overline{A}$ el algoritmo con entrada $x \in I_Q$ tal que su salida $y \in \{s, n\}$ es de valor contrario a la salida de A, i.e. $\forall x \in I_Q$, $\overline{A}(x) = s \iff A(x) = n$. Obviamente $\overline{A}$ es polinomial, pues su tiempo de ejecución es igual al tiempo de ejecución de A, y

$$\forall x \in I_Q = I_{\overline{Q}},\ \overline{A}(x) = s \iff A(x) = n \iff Q(x) = n \iff \overline{Q}(x) = s.$$

Así, $\overline{A}$ resuelve $\overline{Q}$, lo que implica que $\overline{Q} \in P$.
($\Leftarrow$) Supongamos $\overline{Q} \in P$, usando la implicancia anterior se tiene que $\overline{Q} \in P \Rightarrow \overline{\overline{Q}} = Q \in P$.

b) Supongamos que $Q \leq_p R$. Luego, existe $f : I_Q \to I_R$ función que puede ser calculada en tiempo polinomial por un algoritmo A_f y tal que:

$$\forall x \in I_Q,\ Q(x) = s \iff R(f(x)) = s.$$

Luego, si usamos la misma función f se tiene que:

$$\forall x \in I_{\overline{Q}} = I_Q, \quad \overline{Q}(x) = s \iff Q(x) = n \iff R(f(x)) = n \iff \overline{R}(f(x)) = s.$$

Luego, f es una reducción polinomial de $\overline{Q}$ a $\overline{R}$. Por lo tanto, $\overline{Q} \leq_p \overline{R}$. La otra implicancia es análoga considerando como problemas iniciales $\overline{Q}$ y $\overline{R}$. Así, se tiene que $Q = \overline{\overline{Q}} \leq_p \overline{\overline{R}} = R$.

c) Supongamos que $(Q \leq_p R) \ \wedge \ (R \in \text{NP})$. Luego, como $Q \leq_p R$, existe $f : I_Q \to I_R$ una función tal que puede ser calculada por un algoritmo polinomial A_f y que verifica:

$$\forall x \in I_Q,\ Q(x) = s \iff R(f(x)) = s.$$

Notar que como f es calculada en tiempo polinomial por A_f, entonces se tiene que para todo $x \in I_Q,\ size(f(x))$ es polinomial con respecto a $size(x)$, i. e.

$$\exists r \in \mathbb{N},\ \forall x \in I_Q,\ size(f(x)) = O(x^r),$$

pues de lo contrario la sola escritura de $f(x)$ por A_f tomaría un tiempo no polinomial, lo cual es contradictorio con ser A_f un algoritmo polinomial.
Por otro lado, dado que $R \in \text{NP}$, existe $k \in \mathbb{N}$ y un verificador polinomial A_R que recibe como entrada $z \in I_R$ y un certificado $y \in \{0,1\}^*$ con $size(y) = O(size(z)^k)$ tal que $\forall z \in I_R$:

$$R(z) = s \iff \exists y \in \{0,1\}^*,\ size(y) = O(size(z)^k),\ A_R(z, y) = s.$$

Observación: Note que el algoritmo $A_R(z, y)$ verifica las instancias afirmativas de R con certificados de tamaño acotado superiormente por $O(size(z)^k)$, i. e. $size(y) \leq c \cdot size(z)$ para algún $c > 0$ a partir de $size(z) \geq n_0$. Por lo tanto, podemos suponer sin pérdida de generalidad, que nuestro verificador $A_R(z, y)$ recibe sólo entradas y tal que su tamaño es polinomial en $size(z)$. Así, se tiene que: $\forall z \in I_R$:

$$R(z) = s \iff \exists y \in \{0,1\}^*,\ A_R(z, y) = s.$$

Dado $r, k \in \mathbb{N}$ como el definido antes para A_f y A_R respectivamente, se define el algoritmo A_Q, que recibe como entrada $x \in I_Q$ e $y \in \{0,1\}^*$ un certificado con $size(y) = O(size(x)^{rk})$, de la siguiente manera:

Algorithm $A_Q(x, y)$

Input: $x \in I_Q$, $y \in \{0,1\}^*$

1: $z \leftarrow A_f(x)$
2: **if** $A_R(z, y) = s$ **then**
3: **return** s
4: **else**
5: **return** n
6: **end if**

Luego, como A_f es polinomial en el tamaño de x y A_R es polinomial en el tamaño de x e y, entonces A_Q es polinomial en el tamaño de x e y. Además, $\forall x \in I_Q$:

$$\begin{aligned} Q(x) = s &\iff R(f(x)) = s \\ &\iff \exists y \in \{0,1\}^*,\ size(y) = O(size(f(x))^k),\ A_R(f(x), y) = s \\ &\iff \exists y \in \{0,1\}^*,\ size(y) = O(size(f(x))^k),\ z = f(x)\ \wedge\ A_R(z, y) = s \\ &\iff \exists y \in \{0,1\}^*,\ size(y) = O(size(f(x))^k) = O(x^l), l \in \mathbb{N},\ A_Q(z, y) = s \\ &\iff \exists y \in \{0,1\}^*,\ size(y) = O((size(x)^r)^k),\ A_Q(x, y) = s. \end{aligned}$$

Por lo tanto, A_Q es un verificador polinomial de las instancias afirmativas de Q y así Q es NP.

d) Sea la proposición a:= "Todos los lenguajes NP-completos están en P" y
b:= "Ninguno los lenguajes NP-completos están en P".
Luego, debemos probar que la proposición $a \vee b$ es verdadera. Notar que $\neg a \neq b$.
Si b es verdadera, entonces $a \vee b$ es verdadera y tenemos el resultado buscado. Supongamos entonces que b es falsa, luego $\neg b \iff$ existe un lenguaje NP-completo que es P es verdadera. Sea Q un problema de decisión que verifica lo anterior, i. e. Q es NP-completo y está en P. Sea ahora R en NP. Así, por ser Q NP-hard, $R \leq_p Q$ y como por hipótesis $Q \in$P, entonces por resultado visto en clase, $R \in P$. Dado que R es cualquier problema NP, entonces se tiene que todos los problemas NP están en P. Finalmente, como los problemas NP-completos son problemas NP, entonces se tiene que todos los lenguajes NP-completos están en P, i.e. la proposición a es verdadera y por lo tanto, $a \vee b$ es verdadera.

P3) En clase se vió que CLIQUE es NP y CLIQUE $\leq_p$ VERTEX-COVER. Luego, sólo falta probar que CLIQUE es NP-hard y que VERTEX COVER es NP-completo. Para probar que VERTEX COVER es NP-hard se puede usar una clásica y conocida reducción: 3-SAT $\leq_p$ CLIQUE, explicada en clase y descrita por ejemplo en el libro de Sipser. Para mostrar que VERTEX COVER es NP se puede usar el siguiente verificador polinomial.

Algorithm A(G, k, U)

Input: $G = (V, E)$ un grafo no dirigido, $k \in \mathbb{N}$ y U un conjunto de l vértices.

```
1: if l ≤ k y U ⊆ V then
2:    for all {u, v} ∈ E do
3:       if {u, v} ∩ U = ∅ then
4:          return n
5:       end if
6:    end for
7:    return s
8: end if
9: return n
```

Luego, notar que determinar si $U \subseteq V$ requiere $O(|V|^2)$ operaciones de comparación en el peor caso. El ciclo for son $O(|E|)$ iteraciones en el peor caso y determinar si $\{u, v\} \cap U \neq \emptyset$ requiere comparar u y v con los elementos de U lo que requiere $O(|V|)$ operaciones de comparación en el peor caso. Así, el tiempo de ejecución de A es $O(|V|^2) + O(|E|) \cdot O(|V|) = O(|V|^3)$, i.e A es polinomial y con certificado U que verifica $size(U) = O(|V|)$ (polinomial en el tamaño de G). Además, es claro que G tiene un VERTEX COVER U con $|U| \leq k$ si y sólo si A(G, k, U)=s. Por lo tanto, VERTEX COVER es NP y por consiguiente NP-completo. Por último, para probar que CLIQUE es NP-hard se puede mostrar que VERTEX-COVER $\leq_p$ CLIQUE de una manera análoga a lo hecho en clases para probar que CLIQUE $\leq_p$ VERTEX-COVER (ejercicio) y usar el hecho que VERTEX-COVER es NP-hard.

P4) *a*) Recordar que un problema es NP-completo si es NP-hard y es NP. Por hipótesis HAMPATH es NP-hard. Para mostrar que HAMPATH es NP podemos construir un verificador polinomial similar al exhibido para el problema HAMILTONIAN, donde el certificado es una secuencia de vértices de un grafo dado, que podemos verificar en tiempo polinomial si es un camino Hamiltoniano.

Otra forma de probar que HAMPATH es NP-hard es usando el resultado de la P2 *b*). Para ello definamos la siguiente función $f : I_{\text{HAMPATH}} \to I_{\text{HAMILTONIAN}}$ definido por: $\forall\ G = (V, E)$ grafo dirigido con n vértices, $f(G) = G'$ donde $G' = (V', E')$ es un digrafo con $V' = V \cup \{x\}$ tal que $x \notin V$ y $E' = E \cup \{(x, v), (v, x) : v \in V\}$. Luego, si G tiene un camino Hamiltoniano $p : v_1, v_2, \ldots, v_n$ con $V = \{v_1, \ldots, v_n\}$, entonces $p' : x, v_1, v_2, \ldots, v_n, x$ es un ciclo Hamiltoniano en G'. Análogamente si $q : u_1, u_2, \ldots, u_{n+1}, u_1$ es un ciclo Hamiltoniano, entonces sin pérdida de generalidad podemos suponer que $u_1 = x$ y así $q' : u_2, \ldots, u_{n+1}$ es un camino Hamiltoniano en G. En resumen, $f(G) = G'$ es Hamiltoniano si y sólo si G tiene un camino Hamiltoniano. Por último, como la construcción de G' puede ser realizada fácilmente a partir de G, por ejemplo a partir de la inclusión de una fila y columna a la matriz de adyacencia de G, lo que requiere $O(n)$ operaciones elementales, entonces f puede ser calculada en tiempo polinomial en el tamaño de G. Por lo tanto, HAMPATH $\leq_p$ HAMILTONIAN. Como en clase de vió que HAMILTONIAN es NP y usando el resultado de la pregunta P2 *b*) se tiene que HAMPATH es NP y así HAMPATH es NP-completo.

Por otro lado, dado que HAMPATH $\leq_p$ HAMILTONIAN y HAMPATH es NP-hard, entonces por resultado visto en clase, HAMILTONIAN es NP-hard. Además, como HAMILTONIAN es NP, entonces HAMILTONIAN es también NP-completo.

b) ($\Rightarrow$) Supongamos que NP=co-NP, como por *a*) HAMPATH es NP, entonces directamente se tiene que HAMPATH es co-NP.

($\Leftarrow$) Supongamos que HAMPATH es co-NP. Sea $Q \in$ NP. Como por a) HAMPATH es NP-hard, entonces $Q \leq_p$ HAMPATH. Por resultado de la tarea 2, se sabe que para todo problema de decisión A y B: $A \leq_p B \iff \overline{A} \leq_p \overline{B}$. Luego,

$$\overline{Q} \leq_p \overline{\text{HAMPATH}}.$$

Como por hipótesis HAMPATH es co-NP, entonces $\overline{\text{HAMPATH}}$ es NP y por resultado de la P2 b), $\overline{Q}$ es NP, i.e. Q es co-NP, luego NP $\subseteq$ co-NP.
Sea ahora $Q \in$ co-NP, luego $\overline{Q}$ es NP. Por ser HAMPATH NP-hard, $\overline{Q} \leq_p$ HAMPATH $\iff Q \leq_p \overline{\text{HAMPATH}}$ y al ser también HAMPATH co-NP, i.e. $\overline{\text{HAMPATH}}$ es NP, entonces por resultado de la P2 b) se tiene que Q es NP. Así, co-NP$\subseteq$ NP y por lo tanto, NP=Co-NP.

P5) a) Mostremos primero que TOUR CON RESTRICCIÓN es NP. En efecto, dada una instancia (C, d, k) de TOUR CON RESTRICCIÓN con $C = \{c_1, \ldots, c_n\}$ un conjunto de $n \in \mathbb{N}$ ciudades diferentes y $d = \{d(c_i, c_j)\}_{i,j=1}^{n}$ un conjunto con las distancias de los trayectos entre cualquier par de ciudades y $k \in \mathbb{N}$, podemos escoger como certificado una secuencia $y = c_{i_1}, \ldots, c_{i_n}$ de n ciudades tal que $\{c_{i_1} \ldots, c_{i_n}\} = C$ y $\forall j = 1, \ldots, n-1,\ d(c_{i_j}, c_{i_{j+1}}) \leq k$. Notar que otro certificado posible es el orden de los índices de las ciudades a visitar, i.e. $y = (i_1, i_2, \ldots, i_n)$ una permutación del conjunto $\{1, \ldots, n\}$. En cualquiera de los dos casos $size(y) = O(n)$ y $size(C, d, k) = O(n^2 + \log(k))$, pues $size(k) = O(\log(k)),\ size(C) = O(n)$ y $size(d) = O(n^2)$. Luego, obviamente $size(y)$ es polinomial en $size(C, d, k)$. Así definamos el siguiente algoritmo:

Algorithm A(C, d,k, y)

Input: $k \in \mathbb{N}$, $C = \{c_1, \ldots, c_n\}$ un conjunto de n elementos distintos e $y = c_{i_1} \ldots, c_{i_m}$ una secuencia de elementos de C.

```
1: if m ≠ n ∨ {c_{i_1}, ..., c_{i_n}} ≠ C then
2:    return n
3: end if
4: for j = 1 to n − 1 do
5:    if d(c_{i_j}, c_{i_{j+1}}) > k then
6:       return n
7:    end if
8: end for
9: return s
```

Saber si $m \neq n$ requiere solo una operación elemental de tipo comparación y $\{c_{i_1}, \ldots, c_{i_n}\} \neq C$ requiere $O(n^2)$ operaciones de comparación, entonces la línea 1del algoritmo anterior requiere $O(n^2)$ operaciones elementales. Además, en el ciclo for se realizan $O(n)$ operaciones elementales en el peor caso. Por lo tanto, el tiempo de ejecución del algoritmo A es $O(n^2)$, luego A es polinomial.
Por otro lado, obviamente $A(C, d, k, y) =$s si y sólo si existe una secuencia de las n ciudades de C a visitar $y = c_{i_1} \ldots, c_{i_n}$ y tal que $\forall j = 1, \ldots, n-1,\ d(c_{i_j}, c_{i_{j+1}}) \leq k$ si y sólo si (C, d, k) es una instancia afirmativa de TOUR CON RESTRICCIÓN. En resumen, A es un verificador polinomial de TOUR CON RESTRICCIÓN. Por lo tanto, TOUR CON RESTRICCIÓN es NP.

Mostremos ahora que TOUR CON RESTRICCIÓN es NP-hard. Para ello probaremos que HAMPATH $\leq_P$ TOUR CON RESTRICCIÓN. Sea $G = (V, E)$ un grafo dirigido con $V = \{v_1, \ldots, v_n\}$ definamos la función f definido por $f(G) = (C, d, k)$ una instancia de TOUR

CON RESTRICCIÓN definido por: $C = V$, $k = 1$ y d como sigue:

$$\forall i, j = 1, \ldots, n, \qquad d(v_i, v_j) = \begin{cases} 0 & \text{si } i = j, \\ 1 & \text{si } (i,j) \in E, \\ 2 & \text{si } (i,j) \notin E. \end{cases}$$

Luego, G es una instancia afirmativa de HAMPATH si y sólo si existe $p : v_{i_1}, v_{i_2}, \ldots, v_{i_n}$ es un camino Hamiltoniano en G si y sólo si $\{v_{i_1}, \ldots, v_{i_n}\} = V$ y $\forall j = 1, \ldots, n-1,\ (v_{i_j}, v_{i_{j+1}}) \in E$ si y sólo si $\{v_{i_1}, \ldots, v_{i_n}\} = V$ y $\forall j = 1, \ldots, n-1,\ d(v_{i_j}, v_{i_{j+1}}) = 1$ si y sólo si $f(G) = (k, C, d)$ es una instancia afirmativa de TOUR CON RESTRICCIÓN. Por otro lado, un algoritmo que calcula f puede ser el siguiente:

Algorithm A'(G)

Input: $G = (V, E)$ un digrafo con $V = \{v_1, \ldots, v_n\}$

```
1: C ← ∅, k ← 1
2: for i = 1 to |V| do
3:     C ← C ∪ {v_i}
4: end for
5: for i, j = 1 to |V| do
6:     if i = j then
7:         d(v_i, v_j) ← 0
8:     else if (i, j) ∈ E then
9:         d(v_i, v_j) ← 1
10:    else
11:        d(v_i, v_j) ← 2
12:    end if
13: end for
14: return (k,C,d)
```

De esta forma las operaciones elementales para calcular $f(G) = (C, d, k)$ en A' son todas de tipo asignación de valores, luego en la definición de $k = 1$ hay 1 operación, en la de C hay $O(size(V)) = O(n)$ operaciones y en la de d hay $O(n^2)$ operaciones elementales. Así, f puede ser calculada con $1 + O(n) + O(n^2) = O(n^2)$ operaciones elementales, ie. puede ser calculada en tiempo lineal con respecto al tamaño de G (que es $size(G) = O(n^2)$). Por lo tanto, f es una reducción polinomial y como HAMPATH es NP-hard, entonces TOUR CON RESTRICCIÓN es NP-hard y por lo tanto, es NP-completo.

b) Mostremos primero que FEEDBACK VERTEX SET es NP. Dado un grafo dirigido $G = (V, E)$ y $k \in \mathbb{N}$ podemos escoger como certificado un conjunto de vértices $y = \{u_1, \ldots, u_l\} \subseteq V$ que es un FVS de G con $l \leq k$. Notar que la condición de que cada ciclo contiene al menos un vértice de y es equivalente a que el grafo dirigido resultante de eliminar los vértices de y en G, i.e. $G - \{u_1, \ldots, u_l\}$, no tiene ciclos, esto es que $G - y$ es acíclico, lo que puede ser resuelto por un algoritmo polinomial como se demostró en P1 *a*). De esta forma un verificador polinomial para FEEDBACK VERTEX SET puede ser el siguiente algoritmo $A(G, k, y)$.

Algorithm A(G,k, y)

Input: $G = (V, E)$ un digrafo, $k \in \mathbb{N}$ e $y = \{v_1, \ldots, v_l\}$ un conjunto de vértices.

```
1: if l ≤ k y {v_1, ..., v_l} ⊆ V then
2:    G' ← G − y
3:    if Acyclic(G')=s then
4:       return s
5:    else
6:       return n
7:    end if
8: end if
9: return n
```

Las condiciones de la línea 1 pueden ser chequeadas fácilmente con $O(|V|^2)$ operaciones elementales (verificar que un vértices de y está en los vértices de V significa comparar este elemento con todos los otros en tiempo $O(|V|)$ y la condición $l \leq k$ es una operación elemental de comparación entre enteros). Notar que primero se chequea que $l \leq k$ para que el certificado sea de largo a lo más el número de vértices de G. La obtención de $G' = G - y$ se puede hacer eliminando cada vez de la matriz de adyacencia de G la fila y la columna correspondiente a un vértice de y. Luego en total se requiere $O(|V|^2)$ operaciones elementales para obtener G'. En resumen las líneas 1 y 2 puede ser ejecutadas en tiempo polinomial en el tamaño de G. Como además el algoritmo *Acyclic* es polinomial en el tamaño de G' y este a su vez verifica que $size(G') = O(size(G))$ entonces *Acyclic* es polinomial en el tamaño de G. Por lo tanto, FEEDBACK VERTEX SET es NP.

Por otro lado, para probar que FEEDBACK VERTEX SET es NP-hard mostraremos que VERTEX COVER $\leq_p$ FEEDBACK VERTEX SET.
Sea $f : I_{\text{VERTEX COVER}} \to I_{\text{FEEDBACK VERTEX SET}}$ la función definida por: $\forall G = (V, E)$ un grafo no dirigido y $k \in \mathbb{N}$, $f(G, k) = (G', k')$ donde $G' = (V', E')$ es un grafo dirigido con $V' = V$ y $E' = \{(u, v), (v, u) : \{u, v\} \in E\}$ (i.e. por cada arista $\{u, v\} \in E$ se crea los arcos (u, v) y (v, u)) y $k' = k$. Luego, si $U = \{u_1, \ldots, u_l\} \subseteq V$ es un vertex cover de G (i.e. $\forall \{a, b\} \in E$, $\{a, b\} \cap U \neq \emptyset$) con $l \leq k$, entonces U es un FVS de G con $|U| = l \leq k = k'$. Por contradicción, supongamos que existe un ciclo $c : v_1, v_2, \ldots, v_r, v_1$ tal que $\{v_1, \ldots, v_r\} \cap U = \emptyset$. Luego, en particular se tiene que $\{v_1, v_2\} \in E$ verifica que $\{v_1, v_2\} \cap U = \emptyset$ lo que es una contradicción con U un vertex cover. En la otra dirección, si $W \subseteq V'$ es un FVS de G' con$|W| \leq k'$ entonces para todo ciclo $c' = a_1, a_2 \ldots, a_s, a_1$, $\{a_1 \ldots, a_s\} \cap U = \emptyset$. En particular, se tiene que para todo $\{b_1, b_2\} \in E$ existe el ciclo de largo dos en G' $c' = b_1, b_2, b_1$, así $b_1 \in U$ o $b_2 \in U$, luego $\{b_1, b_2\} \cap U \neq \emptyset$. Por lo tanto, W es un vertex cover de G con $|W| \leq k = k'$. En resumen, G tiene un vertex cover U con $|U| \leq k$ si y solo si G' tiene un FVS W con $|W| \leq k'$. Es decir, (G, k) es una instancia afirmativa de VERTEX COVER si y sólo si $f(G, k) = (G', k')$ es una instancia afirmativa de FEEDBACK VERTEX SET. Por último es fácil ver que f puede ser calculada en tiempo polinomial pues el número de operaciones elementales requeridas para construir (G', k') a partir de (G, k) es constante ya que la matriz de adyancencia de G' es igual a la matriz de adyacencia de G, sólo hay que agregar una codificación para indicar que es un grafo dirigido, y $k' = k$. Así, f es una reducción polinomial y como se vió en clase que VERTEX COVER es NP-hard, entonces FEEDBACK VERTEX SET es también NP-hard y como también es NP, entonces es NP-completo.