UNIVERSIDAD DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

Matemáticas Discretas para la Computación - CC3101
Control 3 - Semestre Otoño 2015

1. Sea G un grafo simple y conexo. Decimos que G es *k-crítico*, para $k \geq 1$, si (a) G es coloreable con k pero no con $k-1$ colores, y (b) al sacar cualquier arco de G se obtiene un grafo que se puede colorear con $k-1$ colores.

   Demuestre que si G es k-crítico, entonces cada uno de sus vértices tiene grado al menos $k-1$.

   Solución: Asuma por contradicción que G es k-crítico pero tiene un vértice v de grado estrictamente menor a $k-1$. Saque un arco e cualquiera que sea incidente a v. Por definición, el grafo resultante puede colorearse con $k-1$ colores $c_1, \ldots, c_{k-1}$. Utilicemos ahora esta coloración para todos los vértices de G excepto v (note que esta es una coloración de G sin v). Dado que v tiene a lo más $k-2$ vecinos existe un color disponible para v dentro de $c_1, \ldots, c_{k-1}$. Esto quiere decir que G es $k-1$ coloreable, lo que implica una contradicción.

2. El propósito del siguiente algoritmo es computar caminos de menor peso en grafos dirigidos cuyos pesos pueden ser negativos. En particular, el peso de un camino entre dos vértices puede ser negativo y existen casos en que el peso del camino de menor peso entre dos vértices puede ser $-\infty$ (cuando existen circuitos dirigidos de peso negativo).

   Sea $G = (V, E)$ un grafo dirigido (sin loops ni multiarcos) con $n \geq 1$ vértices y s un vértice en V. Asuma que $w : E \to \mathbb{Z}$ una función que asigna un peso $w(u, v)$ a cada arco $(u, v) \in E$. Nuestro algoritmo computa iterativamente un valor $d(v)$ para cada vértice $v \in V$ de la siguiente forma:

   a) Inicialmente $d(s) := 0$ y $d(v) := \infty$ para cada $v \in V$ que no es s.
   b) Luego se realizan $n-1$ iteraciones del siguiente proceso: Por cada arco $(u, v) \in E$, si $d(v) > d(u) + w(u, v)$ entonces se cambia el valor de $d(v)$ a $d(u) + w(u, v)$.
   c) Finalmente, por cada arco $(u, v) \in E$ se verifica si $d(v) > d(u) + w(u, v)$. De ocurrir esto para algún arco, se retorna el valor `falso` y se detiene. De no ser así, se retorna el valor `verdadero` y se detiene.

   Demuestre lo siguiente:

   a) (2pts) Si G no contiene un circuito dirigido de peso negativo que pueda ser alcanzado desde s, entonces al terminar el paso (b) se cumple lo siguiente para cada $v \in V$ tal que existe camino dirigido de s a v en G: El valor de $d(v)$ es el peso del camino de menor peso de s a v en G.

   Solución: Considere el camino $v_1, \ldots, v_k$ de menor peso que lleva de s a v (es decir, $s = v_1$ y $v = v_k$). Este camino no pasa dos veces por el mismo vértice (ya que no hay circuitos de peso negativo); por tanto, tiene a lo más $n-1$ arcos (i.e., $k \leq n$). Además, $v_1, \ldots, v_i$ es un camino de menor peso posible de $v_1 = s$ a v_i, para cada $1 \leq i \leq n$. En la primera iteración del paso (b)

obtenemos $d(v_2) = w(v_1, v_2)$, lo que corresponde al peso del camino de menor peso de v_1 a v_2. Asuma ahora que en la $i-1$-ésima iteración se obtiene que $d(v_i)$ corresponde al peso del camino de menor peso de $v_1 = s$ a v_i ($1 < i < k$). Es decir, $d(v_i) = w(v_1, v_2) + \cdots + w(v_{i-1}, v_i)$. En la i-ésima iteración obtenemos entonces que $d(v_i) = d(v_{i-1}) + w(v_i, v_{i+1})$, lo que corresponde al peso del camino de menor peso de $v_1 = s$ a v_{i+1}. El resultado se sigue del hecho que $k \leq n$.

b) (1,5pts) Existe camino dirigido de s a v en G si y solo si $d(v) < \infty$ al terminar el algoritmo.

Solución: Claramente, $d(v)$ corresponde al peso de algún camino de s a v si $d(v) < \infty$. Por tanto, en este caso v puede ser alcanzado desde s mediante camino dirigido. Por otro lado, asuma que existe camino dirigido de s a v. Por tanto, existe camino simple de s a v. Tal camino contiene a lo más $n - 1$ arcos; por tanto, su peso es un valor finito. Una simple demostración inductiva demuestra que $d(v)$ debe ser menor o igual al peso de tal camino al finalizar el paso (b).

c) (2,5pts) Si G no contiene circuitos de peso negativo que puedan ser alcanzados desde s, entonces el valor retornado es `verdadero`. En caso contrario, el algoritmo retorna el valor `falso`.

Solución: Asuma que G no contiene circuitos de peso negativo que pueden alcanzarse desde s. De las dos partes anteriores podemos deducir que $d(v)$ es el peso del camino de menor peso de s a v para cada $v \in V$. Por tanto, al comenzar el paso (c) del algoritmo se cumple que $d(v) \leq d(u) + w(u, v)$ para cada $(u, v) \in E$. Por tanto, el algoritmo no retorna el valor `falso` (y entonces retorna `verdadero`).

Asuma, por otro lado, que G contiene un ciclo negativo que se puede alcanzar desde s. Supongamos que este ciclo es $v_0, v_1, \ldots, v_k$, donde $v_0 = v_k$. Por tanto, $\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$. Asumamos, por contradicción, que el algoritmo retorna el valor `verdadero`. Por tanto, al terminar paso (b) se cumple que $d(v_i) \leq d(v_{i-1}) + w(v_{i-1}, v_i)$ para todo $1 \leq i \leq k$. Esto implica que

$$\sum_{i=1}^{k} d(v_i) \;\leq\; \sum_{i=1}^{k} d(v_{i-1}) + \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Pero dado que $v_0 = v_k$ se cumple que $\sum_{i=1}^{k} d(v_i) = \sum_{i=1}^{k} d(v_{i-1})$. Por la pregunta anterior cada $d(v_i)$ es un valor finito (ya que v_i puede ser alcanzado desde s en G), lo que implica que $\sum_{i=1}^{k} w(v_{i-1}, v_i) = 0$. Esto es una contradicción.

3. Un *grafo etiquetado* es un par (G, ℓ), donde $G = (V, E)$ es un grafo simple y $\ell : V \to \{0, 1\}$ es una función que etiqueta con valor 0 o 1 a cada vértice de G. Un árbol etiquetado es un grafo etiquetado que satisface que G es un árbol. En tal caso denotamos a G por T.

   Considere dos grafos etiquetados (G_1, ℓ_1) y (G_2, ℓ_2), donde $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$. Un *homomorfismo* de (G_1, ℓ_1) a (G_2, ℓ_2) es una función $h : V_1 \to V_2$ tal que:

   a) Para cada $v \in V_1$ se cumple que $\ell(v) = 0$ si y solo si $\ell(h(v)) = 0$.

   b) Para cada $u, v \in V_1$ se cumple que si $(u, v) \in E_1$ entonces $(h(u), h(v)) \in E_2$.

   Construya un procedimiento que dados un grafo etiquetado (G, ℓ) y un árbol eqtiquetado (T, ℓ') determine si existe homomorfismo de (T, ℓ') a (G, ℓ). Su procedimiento debe realizar a lo más $O((nm)^c)$ operaciones, donde n es el número de vértices en G, m es el número de vértices en T, y $c \geq 1$ es la constante que usted desee. (Es importante notar que tal procedimiento existe debido a que T es un árbol. En particular, su procedimiento no puede funcionar cuando T es un grafo cualquiera).

Hint: Utilice un procedimiento recursivo desde las hojas del árbol T hasta su raíz. Este procedimiento va asignando a cada vértice t de T un subconjunto $S(t)$ de los vértices de G de tal forma que se cumple lo siguiente: Sea H el conjunto de todos los homomorfismos que existen desde el subárbol de T cuya raíz es t al grafo G. Entonces $S(t) = \{h(t) \mid h \in H\}$.

Solución: Asigne a cada hoja t en T etiquetada con 0 el conjunto $S(t)$ de todos los vértices de G que están etiquetados con 0. Análogamente, asigne a cada hoja t de T etiquetada con 1 el conjunto de todos los vértices de G que están etiquetados con 1. Claramente, esto satisface la propiedad especificada en el *Hint* a nivel de las hojas.

Sea t un vértice de T cuyos hijos son $t_1, \ldots, t_k$. Asuma inductivamente que el procedimiento ya ha asignado subconjuntos $S(t_1), \ldots, S(t_k)$ a $t_1, \ldots, t_k$, respectivamente. Asumimos inductivamente que cada $S(t_i)$ cumple la propiedad especificada en el *Hint*. Si t está etiquetado con 0 entonces $S(t)$ es el conjunto de todos los vértices v en G que están etiquetados con 0 y satisfacen lo siguiente: Para cada $1 \leq i \leq k$ existe un nodo $v_i \in S(t_i)$ tal que (v, v_i) es un arco de G. Análogamente si t está etiquetado con 1. No es difícil demostrar que $S(t)$ cumple la propiedad especificada en el *Hint*.

Sea r la raíz de T. Si $S(r) \neq \emptyset$ entonces el procedimiento acepta y declara la existencia de un homomorfismo de (T, ℓ') en (G, ℓ). En caso contrario, declara que no existe tal homomorfismo. La correctitud y completitud del algoritmo se sigue de que la propiedad especificada en el *Hint* es cierta para r (debido a que el subárbol de T con raíz r es precisamente T).

Es claro que el procedimiento corre en tiempo $O((nm)^c)$ para alguna constante $c \geq 1$.