PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 2133 — Estructuras de Datos y Algoritmos
Interrogación 3
Primer Semestre, 2017

Duración: 2 hrs.

1. *a*) (1/3) Sea $G = (V, E)$ un grafo *dirigido acíclico*. ¿Cuántas componentes fuertemente conexas encuentra el algoritmo de Kosaraju al ser ejecutado en G? Justifique.

   Respuesta (1 punto): El algoritmo de Kosaraju encuentra todas las componentes fuertemente conexas de un grafo.
   Un grafo dirigido acíclico es un grafo que (valga la redundancia) no posee ningún ciclo y sus nodos se conectan en un solo sentido. Al no existir ciclos, no existe camino que permita a un nodo conectarse con si mismo.

   Por otro lado, una componente fuertemente conexa, se define como un grupo de nodos en el que todos los nodos pertenecientes a esta son alcanzables desde cualquier nodo de la componente.

   Dado esto, no pueden haber componentes fuertemente conexas si no se permiten ciclos en un grafo. Por ende, El grafo G no posee componentes fuertemente conexas. *Hasta acá se obtiene todo el puntaje.* Si se define que un nodo es alcanzable por si mismo, entonces la cantidad de componentes fuertemente conexas es $|V|$.

   b) (2/3) Suponga que el algoritmo de abajo se ejecuta con un grafo **dirigido acíclico** G.
   i. Ejecute DFS sobre G, almacenando los tiempos de finalización.
   ii. Ejecute DFS-visit(G, u), donde u es el nodo de G que tiene el mayor tiempo de finalización.
   iii. Si todos los nodos están marcados negros, retorne TRUE. En caso contrario, retorne FALSE.

   ¿Qué propiedad tiene G cuando el algoritmo retorna TRUE? Justifique.

   Respuesta (2 puntos): El nodo u cumple la propiedad de no tener ancestros y ser ancestro de todos los nodos del grafo (por ende todos los nodos $-\{u\}$ tienen mínimo un ancestro). **(1 punto)** ¿Por qué?

   Por el paso i (DFS), se almacenan todos los tiempos de finalizacón. El nodo que se cierra último (último en volverse negro), en un grafo acíclico, es el primero en el orden topológico correspondiente y este nodo no tiene ancestros. **(0,25 puntos)** DFS-visit es el algoritmo DFS pero que parte de un nodo y sigue solo los caminos alcanzables por ese nodo. Luego se detiene. **(0,25 puntos)**

   Un nodo se marca negro cuando visitó a todos sus descendientes. Si todos los nodos están marcados negros, se visitó a todos los nodos del grafo. Por ende, si un DFS-visit(G,u) marca a todos los nodos negros, significa que se logró alcanzar a todos los nodos del grafo desde u. Dado que este grafo no tiene ciclos, u no tiene ancestros. **(0,5 puntos)**

   Estos son puntajes parciales, en caso de no tener la respuesta completa o correcta.

2. Hashing universal es una técnica para generar buenas funciones de hash. Dado un universo de claves U, se definen la funciones $g_{a,b}(k) = (ak + b) \bmod p$ y $h_{a,b}(k) = g_{a,b}(k) \bmod m$, donde p es un primo tal que $p > k$, para cada $k \in U$, y $a \in \{1, \dots, p-1\}$ y $b \in \{0, \dots, p-1\}$.

   a) Una de las razones porque $h_{a,b}$ es "buena" es porque "evita colisiones antes de módulo m". Esto significa que si $k \neq k'$, entonces $g_{a,b}(k) \neq g_{a,b}(k')$. Demuestre este resultado.

   Respuesta (2 puntos): Esto se puede demostrar por contradicción, asumiremos lo contrario, es decir $k \neq k'$ y $g_{a,b}(k) = g_{a,b}(k')$. Desarrollamos la igualdad:

$$
\begin{aligned}
g_{a,b}(k) &= g_{a,b}(k') \\
g_{a,b}(k) - g_{a,b}(k') &= 0 \\
(ak + b) \bmod p - (ak' + b) \bmod p &= 0
\end{aligned}
$$

   Utilizando las propiedades del módulo y que la resta siempre estará en el rango $\{0, p-1\}$ llegamos a que:

$$
\begin{aligned}
((ak + b) - (ak' + b)) \bmod p &= 0 \\
(a(k - k')) \bmod p &= 0
\end{aligned}
$$

   Luego, para que se cumpla la igualdad se debe cumplir alguna de las siguientes condiciones:

   - $a \bmod p = 0$, lo cual no es cierto debido a que $a \in \{1, \dots, p-1\}$ y $b \in \{0, \dots, p-1\}$.
   - $(k - k') \bmod p = 0$, lo cual no es cierto ya que $p > k$ y por lo tanto la diferencia nunca estará fuera del rango $\{0, p-1\}$. Además como $k \neq k'$ la resta nunca será 0.
   - $a(k - k') = cp$ con c entero. Tampoco es cierto, ya que si lo fuera a o $(k - k')$ serían divisores de p, lo cual sumando el hecho de que $a < p$, y $(k - k') < p$ contradice el hecho de que p es primo.

   Por lo tanto llegamos a una contradicción, ya que es imposible que se cumpla la igualdad.

   Asignación de puntaje:

   - **1 punto** por demostrar que la expresión no es igual a 0 antes del módulo.
   - **1 punto** por demostrar que la expresión no es igual a un múltiplo de p.

   b) El teorema de *a*) se puede demostrar sin obligar a que p sea primo, pero imponiendo otras restricciones sobre $g_{a,b}(k)$. Diga cuáles y demuestre su respuesta.

   Respuesta (2 puntos): Se mantienen la condiciones anteriores pero agregando la condición de que a sea primo relativo a p, es decir que a no tenga divisores en común con p. La demostración es equivalente a la de *a*), solo que ahora $a(k - k') = cp$ no puede ser cierto ya que significaría que un factor primo de a está en p.

   Asignación de puntaje:

   - **2 puntos** por respuesta correcta junto a su demostración.
   - **1.5 puntos** por respuesta correcta con errores en la justificación.
   - **1 punto** por solo decir que a no sea un divisor de p (Esto no es suficiente, podría pasar que a sea un factor de cp)
   - **0.5 puntos** por dar condiciones demasiado restrictivas, por ejemplo restringir que $|a(k - k')|$ sea menor a p.

c) Muestre que si relajamos la restricción "$p > k$, para cada $k \in U$" es posible que $g_{a,b}(k) = g_{a,b}(k')$ incluso cuando $k \neq k'$.

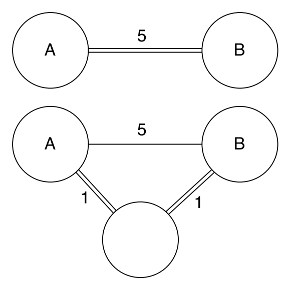
Respuesta (2 puntos):
Ahora se puede dar que $k - k' = cp$ y por lo tanto es posible que $(a(k - k')) \bmod p = 0$. También se puede demostrar con un contraejemplo, un caso sería $a = 1, b = 0, p = 3, k = 1, k' = 4$.

$$g_{a,b}(k) = (1 \cdot 1 + 0) \bmod 3 = 1$$
$$g_{a,b}(k') = (1 \cdot 4 + 0) \bmod 3 = 1$$

La asignación de puntos es binaria, se dan 0 o 2 puntos.

3. Sea $G = (V, E)$ un grafo no dirigido y $T \subseteq E$ el único árbol de cobertura de costo mínimo (MST) de G. Suponga ahora que G' se construye agregando nodos a G y aristas que conectan estos nuevos nodos con nodos de G. Finalmente, sea T' un MST de G'.

a) (1/3) Demuestre que no necesariamente $T \subseteq T'$.
Respuesta: Basta con dar un contraejemplo:



Las aristas dobles muestran el árbol del grafo. En la primera imagen el árbol T es $\{(A, B)\}$, mientras que en el segundo grafo el árbol T' es $\{(A, C), (B, C)\}$. Es evidente que $T \notin T'$

Asignación de puntaje: Si se da un contraejemplo o se hace una demostración formal: **1pto**. Si se hace una demostración formal pero tiene algun error pequeño: **0.5pts**. Else: **0pts**.

b) (2/3) Dé una condición necesaria y suficiente para garantizar que $T \subseteq T'$. Demuéstrela.
Respuesta: Para asegurar que $T \in T'$, se debe asegurar para cada arista $(u, v) \in T$ que: Si se genera un camino c nuevo que conecta u con v, entonces al menos existe una arista $a \in c$ tal que el costo de a es mayor al costo de (u, v).

Demostración:
Suficiencia: Dado que el algoritmo de kruscal es correcto, podemos decir lo siguiente:
Dados dos nodos $u, v \in G$ tal que $(u, v) \in T$, se tiene que existe un camino nuevo que conecta u con v en el cual existe una arista a más cara que (u, v). En algun paso de la ejecución del algoritmo de kruscal se tendrá que u y v están en dos grupos no conectados de nodos. Se puede asegurar que el algoritmo de kruscal no conectará los grupos de nodos de u con el de v por la arista a ya que primero se revisan las aristas más bartas, por lo que primero se conectarían a través de (u, v). Por lo tanto, esta propiedad es suficiente.

Necesaria: Si no se cumple esta propiedad entonces ejecutando el algoritmo de kruscal se conectaría u con v a través del nuevo camino, por lo que no se agregaría (u, v) al árbol T'. Por lo tanto, si la propiedad, puede pasar que $T \notin T'$.

Respuesta equivalente: Para todo corte del grafo G' tal que existen nodos de G en ambos lados del corte, se debe cumplir que la arista más barata que cruza el corte pertenece al árbol T.

Asignación de puntaje:

- (1pto) Enunciar una propiedad necesaria y suficiente. (si la propiedad es solo suficiente o solo necesaria, entonces 0 pts)
- (0.5pto) Demostrar suficiencia (Si la propiedad es suficiente y no necesaria igual va el puntaje *).
- (0.5pto) Demostrar que es necesaria (Si la propiedad es necesaria pero no es suficiente, igual va el puntaje **).
- * Si la propiedad enunciada es suficiente pero muy innecesaria no tiene puntaje.
- ** Si la propiedad es muy insuficiente, no tiene puntaje.