Examen de Febrero 2014

Cómputo de las Componentes Fuertemente Conexas de un digrafo.

Dado un digrafo g, se dice que dos vértices v y w están fuertemente conectados (escrito $v\ FC\ w$) si existe un camino desde v hasta w y otro camino desde w hasta v. Por ejemplo, para el grafo de la izquierda de la Figura 1, los vértices A y B están fuertemente conectados, pero los vértices F y B no lo están.

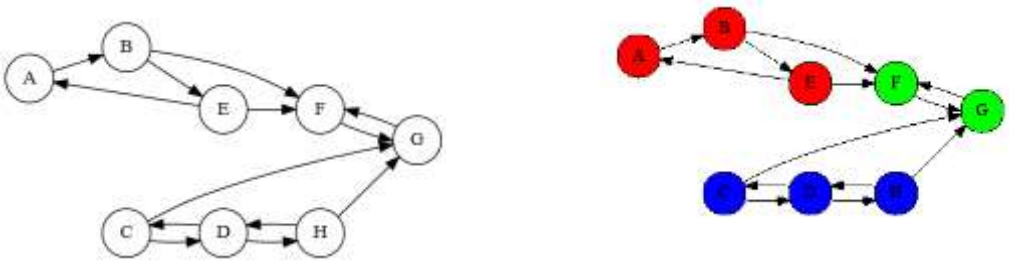

Figura 1. Un digrafo (izquierda) , y sus tres componentes fuertemente conexas.

La relación FC es una relación de equivalencia y sus clases de equivalencia se llaman Componentes Fuertemente Conexas o SCCs (Strongly Connected Components). Por ejemplo, la colección $[A,E,B]$ es una clase de equivalencia, o sea, una componente fuertemente conexa. En la parte derecha de la Figura 1 aparecen del mismo color los vértices de cada una de las tres componentes.

Para computar la SCC (Strongly Connected Component) de un vértice v podemos seguir los siguientes pasos:

1. Exploramos en profundidad el grafo g desde v; sea vs la lista de vértices obtenida. Entre estos vértices estará la componente fuertemente conexa de v ya que la lista vs contiene solo y únicamente los vértices alcanzables desde v.
2. Por ello, nos limitaremos al subgrafo de g con vértices en vs; sea pues el grafo gr, restricción de g al conjunto de vértices de vs.
3. Ahora computamos g', el grafo inverso de gr. Entonces, para cada vértice w visitable desde v en el grafo g', existirá un camino desde w hasta v en el grafo original g.
4. Si exploramos en profundidad el grafo g' desde v, la lista de vértices obtenida es la Componente Fuertemente Conexa (SCC) de v.

Por ejemplo, para el grafo de la Figura 1, la exploración en profundidad desde A proporciona la lista de vértices $[A,B,F,G,E]\ =\ vs$ coloreados en rojo en la parte izquierda de la Figura 2. La restricción de g sobre esta lista, es decir el subgrafo gr obtenido en el paso 2, aparece en el centro de la Figura 2. Su grafo inverso g' obtenido en el paso 3, aparece en la parte derecha de la Figura 2. Al explorar en profundidad g' partiendo del vértice A (paso 4) obtenemos la lista $[A,E,B]$, que es la componente fuertemente conexa del vértice A.

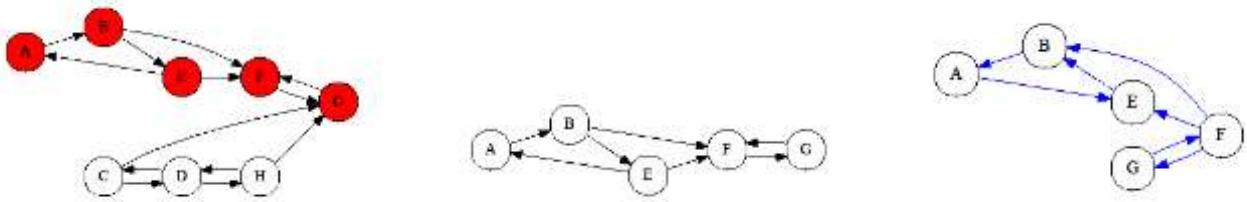

Figura 2. Exploración en profundidad desde A, grafo restringido y grafo inverso del restringido.

Ejercicio en Haskell

(A) Escribe la función que devuelve el grafo inverso de un digrafo:

```
reverseDiGraph :: Eq a => DiGraph a -> DiGraph a
```

(B) Escribe la función que toma un grafo *g* y una lista de vértices *vs* y devuelve el subgrafo de *g* con vértices en *vs:*

```
restrictDiGraph :: Eq a => DiGraph a -> [a] -> DiGraph a
```

(C) Con ayuda de las funciones anteriores, siguiendo los pasos 1-4 descritos anteriormente, escribe una función para computar la SCC sobre un grafo de un determinado vértice:

```
type SCC a = [a]
sccOf :: Ord a => DiGraph a -> a -> SCC a
```

(D) Aplicando reiteradamente la función anterior, podemos obtener todas las componentes del grafo original eliminando en cada paso los vértices de la componente computada. Escribe la función correspondiente a este cómputo:

```
sccs :: Ord a => DiGraph a -> [SCC a]
```

Ejercicio en java

(A) Define un método que tome un digrafo como argumento y devuelve su grafo inverso:

```
public static <V> DiGraph<V> reverseDiGraph(DiGraph<V> g)
```

(B) Escribe el método que toma un grafo *g* y una lista de vértices *vs* y devuelve el subgrafo de *g* con vértices en *vs:*

```
public static <V> DiGraph<V> restrictDiGraph(DiGraph<V> g, Set<V> vs)
```

(C) Con ayuda de los métodos anteriores, siguiendo los pasos 1-4 descritos anteriormente, escribe un método para computar la SCC sobre un grafo *g* de un determinado vértice *src*:

```
public static <V> Set<V> sccOf (DiGraph<V> g, V src)
```

(D) Aplicando reiteradamente el método anterior, podemos obtener todas las componentes del grafo original eliminando en cada paso los vértices de la componente computada. Escribe el método correspondiente a este cómputo:

```
public static <V> Set<Set<V>> stronglyConnectedComponentsDiGraph(DiGraph<V> g)
```