Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2693 - Laboratorio de Algoritmos y Estructuras III
Trimestre Abril-Julio 2016

Laboratorio de la semana 11: Algoritmo de Bellman y Grafos de Precedencia

1. Introducción

El primer objetivo del laboratorio es el de implementar algoritmos de búsqueda de caminos de costo mínimo y de caminos de costo máximo en grafos dirigidos, sin arcos múltiples y sin circuitos, conocidos como grafos de precedencia. El segundo objetivo es la implementación del algoritmo de Bellman para obtener un camino de costo mínimo entre dos pares de vértices, en un grafo dirigido que puede tener costos negativos en los lados, pero no tiene circuitos de costo negativo.

2. Actividad a realizar

La primera actividad es la implementación de la clase `GrafoPrecedencia`, cuyo constructor recibe como entrada un grafo de precedencia $G = (V, E)$ y un vértice de inicio, y que es capaz de determinar un camino de costo mínimo y de costo máximo en un tiempo proporcional a $V + E$. Es decir, en el peor caso los algoritmos de costo mínimo y máximo son $O(V + E)$. El código base con los métodos requeridos se muestran en el Listado 1.

```
1  import java.util.*;
2
3  public class GrafoPrecedencia {
4
5      public GrafoPrecedencia(EdgeWeightedDigraph G, int s) {
6      }
7
8      public double caminoCostoMinimo(int s) {
9      }
10
11     public ArrayList<Integer> obtenerCaminoMimino(int v) {
12     }
13
14     public double caminoCostoMaximo(int s) {
15     }
16
17     public ArrayList<Integer> obtenerCaminoMaximo(int v) {
18     }
19
20     public boolean exitsteCaminoHasta(int s) {
21     }
22 }
```

Listado 1: Esqueleto del código de los procedimientos sobre grafos de prededencia a implementar.

La segunda actividad consiste en la implementación de una clase que contiene el algoritmo de Bellman. El constructor de la clase recibe como entrada un grafo dirigido y un vértice de inicio, el grafo dirigido puede tener arcos con costos negativos, pero el grafo no puede tener un ciclo negativo. El Listado 2 muestra el código parcial con los métodos obligatorios que debe completar.

```
1  import java.util.*;
2
3  public class Bellman {
4
5      public Bellman(EdgeWeightedDigraph G, int s) {
6      }
7
8      public double caminoCostoMinimo(int s) {
9      }
10
11     public ArrayList<Integer> obtenerCaminoMimino(int v) {
12     }
13
14     public boolean exitsteCaminoHasta(int s) {
15     }
16 }
```

Listado 2: Esqueleto del código del algoritmo de Bellman a implementar.

A continuación se explican los métodos obligatorios que deben contener las clases mostradas en el Listado 1 y en el Listado 2.

caminoCostoMinimo: Retorna el costo del camino de costo mínimo hasta el vértice indicado.

caminoCostoMaximo: Retorna el costo del camino de costo máximo hasta el vértice indicado.

obtenerCaminoMimino: Retorna la secuencia de vértices que forman parte del camino de costo mínimo hasta el vértice indicado.

obtenerCaminoMaximo: Retorna la secuencia de vértices que forman parte del camino de costo máximo hasta el vértice indicado.

exitsteCaminoHasta: Retorna "True" en caso de que exista un camino hasta el vértice indicado, retorna "False" en caso contrario.

La implementación de los algoritmos de los Listados 1 y 2, debe estar basada en los algoritmos presentados en el libro de Meza y Ortega [1].

La tercera actividad es la creación de un cliente que muestre el correcto funcionamiento de la clase del Listado 1. El cliente tiene como nombre `ClienteGrafoPrecedencia.java` y recibe como entrada un archivo que contiene los datos de un grafo no dirigido con pesos con el formato visto en clase. Por ejemplo, el cliente se ejecuta de la siguiente manera:

```
>java ClienteGrafoPrecedencia <archivo_con_grafo_dirigido>
```

La cuarta actividad consiste en la implementación de un cliente que pruebe el código del Listado 2. El cliente debe llamarse `ClienteBellman.java`, el cual recibe como entrada un archivo con un grafo dirigido con costos, con el formato visto en clase. Un ejemplo de la ejecución del cliente es como se indica a continuación:

```
>java ClienteBellman <archivo_con_grafo_dirigido>
```

3. Detalles de implementación

El código debe seguir la guía de estilo de Java. También debe crear un archivo Makefile que compile cada uno de los archivos Java entregados.

4. Condiciones de entrega

Debe entregar el día 22 de Junio de 2016 antes de las 2:30 pm, un archivo comprimido llamado **LabSem11-X-Y.tar.gz**, con todos los códigos Java y el archivo Makefile. Las letras **X** y **Y** del archivo comprimido son los número de carné de los integrantes del equipo.

Referencias

[1] Meza, O., and Ortega, M. *Grafos y Algoritmos*, segunda ed. Editorial Equinoccio, 2004.

Guillermo Palma / gvpalma@usb.ve / Junio 2016