Algoritmos y Programación II (75.41)
Trabajo Práctico Nº3

21 de noviembre de 2014

1 Introducción

En cierta universidad de música hay una crisis en una de sus orquestas. No se sabe bien quiénes son los músicos que tocan en la misma, hay un déficit en la comunicación de información importante, los músicos no se conocen entre sí, se juntan a conspirar en grupos aislados, no se reconoce el liderazgo de los mismos, etc.

Se plantea el problema y un maestro propone la solución a todos esos problemas mediante el uso de la red profesional LinkedIn si es que lo nombran director de la orquesta. A todos les parece una idea tan magnífica que lo nombran director por los próximos cuatro años sin dudarlo un instante.

Ahora bien, una vez que llega al cargo, se da cuenta de que necesita una herramienta para explotar esa red profesional y poder llegar a todos los músicos y que él no tiene los conocimientos necesarios para programarla. Por este motivo decide convocar a los alumnos del curso de Algoritmos II de Rosita para que le resuelvan la implementación.

2 Consigna

Implementar una herramienta que permita resolver las siguientes operaciones sobre la red profesional de la orquesta:

Obligatorias:

- Para poder expandir su red a todos los músicos, recomendarle al director nuevos contactos (`recomendar`).
- Como contactar músico por músico es muy engorroso, saber con difundir a qué músicos a cuales músicos se alcanza (`difundir`).
- (Uno de dos) O conocer cuál es el músico más popular de la orquesta[1]

[1] Nunca se sabe para qué puede servir...

(`centralidad`) o conocer cuál es la conexión más débil entre los subgrupos músicos de la orquesta[2] (`debilidad`).

Optativas:

- Distancia mínima de un músico a otro (`camino`).
- Distancia máxima entre todos los músicos (`diametro`).
- Calcular el coeficiente de agrupamiento (`agrupamiento`).
- Cantidad de músicos a distancia 1, 2, 3, etc. de un determinado músico (`distancias`).
- Identificar cuántos subgrupos disconexos de músicos hay (`subgrupos`).
- Identificar puntos de articulación (`articulacion`).

Para aprobar el trabajo práctico deberán implementarse las tres operaciones obligatorias y al menos tres de las optativas.

Además, deberá implementarse un TDA grafo **genérico** con todas las primitivas correspondientes del mismo.

2.1 Grafo

La red de contactos se modelará como un grafo no dirigido donde los nodos serán personas y las aristas indicarán que existe una relación entre ellos.

El formato del grafo será el siguiente:

```
N
0 nombre0
1 nombre1
...
N-1 nombre_N-1
M
0 id_x0 id_y0
1 id_x1 id_y1
...
M-1 id_xM-1 id_yM-1
```

donde `N` es un entero que representa la cantidad de nodos en el grafo, `nombreX` es una cadena que representa el nombre de ese nodo, `M` es un entero que representa la cantidad de aristas del grafo y `id_xX id_xY` son dos enteros que representan una arista entre el nodo `id_xX` y el nodo `id_yY`.

El nombre del archivo conteniendo el grafo a cargar se recibirá como un argumento en línea de órdenes al invocar la aplicación.

[2] Idem músico más popular...

2.2 Protocolo

Este programa formará parte de un sistema automatizado. En lugar de presentar un menú con opciones en la pantalla, se esperará que el programa respete un *protocolo de comunicación*, en el cual se reciben comandos por la entrada estándar (`stdin`). Cada comando recibido debe ser procesado y su resultado debe ser escrito en la salida estándar (`stdout`).

Cada línea de la entrada corresponde a un comando, y todos los comandos respetan un determinado formato. El comando está separado de los parámetros (si existen) por un espacio, y los parámetros están separados entre sí por comas:

```
comando parametro1,parametro2,parametro3,...
```

2.3 Comandos

2.3.1 Comando: `recomendar`

Formato: `recomendar músico cantidad`

Descripción: Se le recomiendan cantidad de contactos a agregar al músico indicado. Los mismos deberán ser ordenados por orden de relevancia decreciente. Basados en el lema *los amigos de mis amigos son mis amigos* la recomendación de un contacto nuevo se hace contando cuántos contactos del músico en común tienen a los músicos que aún no son contacto del mismo.

Parámetros:

`músico`**:** Nombre del músico.

`cantidad`**:** Un número entero positivo mayor a 0 que representa la cantidad de contactos a sugerir.

Salida: Un nombre de músico por línea, ordenados por relevancia decreciente.

Ejemplo:

```
recomendar John 3
Paul
George
Ringo
```

2.3.2 Comando: `difundir`

Formato: `difundir músico1,músico2,...,músicoN`

Descripción: Analiza cómo se difunde una información puntual si es publicada por esa lista de músicos. Estadísticamente se sabe que los músicos no le prestan atención a nada hasta que evalúan que esa información es importante. Consideran que algo es importante si más de la mitad de sus contactos la publicaron, y en ese caso no sólo se enteran sino que la republican. Conocer esto le permitiría al director de la orquesta estar seguro de que todos se enteran de las cosas y sin la necesidad de gastar su valioso tiempo contactando a cada uno de los músicos.

Parámetros:

`músico1`**:** Nombre del primer músico.

`músico2`**:** Nombre del segundo músico.

`músicoN`**:** Nombre del músico N.

Salida: La cantidad, y la lista de todos los músicos que se enterarían de la información difundida.

Ejemplo:

```
difundir Cucho,Perro Viejo
9
Cucho
Perro Viejo
Gustavo
Diego
Gaston
Eduardo
Guillermo
Claudio
Braulio
```

2.3.3 Comando: `centralidad`

Formato: `centralidad cantidad`

Descripción: Calcula el grado de centralidad de todos los músicos e imprime la cantidad pedida en orden de centralidad decreciente. La centralidad da cuenta de qué tan popular es alguien contando, si se computan los caminos mínimos de todos los nodos contra todos los nodos, cuántos de estos caminos pasan por él.

Parámetros:

`cantidad`: Un número entero positivo mayor a 0 que representa la cantidad de músicos populares a mostrar.

Salida: La lista de los músicos más populares en sentido decreciente.

Ejemplo:

```
centralidad 3
Mike
Keith
Charlie
```

2.3.4 Comando: `debilidad`

Formato: `debilidad cantidad`

Descripción: Calcula el grado de debilidad de todas las conexiones e imprime la cantidad pedida en orden de debilidad decreciente. La debilidad da cuenta de que tan necesaria es una conexión, si se computan los caminos mínimos de todos los nodos contra todos los nodos, cuántos de estos caminos pasan por ella.

Parámetros:

`cantidad`: Un número entero positivo mayor a 0 que representa la cantidad de pares de músicos a listar.

Salida: Lista de los pares más débiles en orden decreciente.

Ejemplo:

```
debilidad 2
John, Yoko
Ritchie, Ian
```

2.3.5 Comando: `camino`

Formato: `camino músico1,músico2`

Descripción: Indica el camino a recorrer para llegar de un músico al otro incluyendo los extremos.

Parámetros:

`músico1`: Nombre del músico de partida.

`músico2`: Nombre del músico de llegada.

Salida: La lista de músicos que los relacionan.

Ejemplo:

```
camino David,Pipo
David
Pedro
Fito
Fabiana
Pipo
```

2.3.6 Comando: `diametro`

Formato: `diametro`

Descripción: Indica el máximo de todos los caminos mínimos entre dos músicos de toda la red.

Salida: La longitud del camino, yla lista de los músicos que vinculan ese camino.

Ejemplo:

```
3
Nito
Charly
Luis Alberto
Black
```

2.3.7 Comando: `agrupamiento`

Formato: `agrupamiento`

Descripción: Calcula el coeficiente de agrupamiento[3] promedio de toda la red. Para cada nodo el coeficiente de agrupamiento consiste en cuantos de sus contactos son a su vez contactos entre si y dividirlo por la cantidad maxima de conexiones que puede haber entre estos contactos (sin incluir las conexiones al nodo al cual se le esta calculando el coeficiente).

Salida: El coeficiente de agrupamiento.

Ejemplo:

```
agrupamiento
0.43
```

2.3.8 Comando: `distancias`

Formato: `distancias músico`

Descripción: Lista la cantidad de músicos a distancia 1, a distancia 2, etc. del músico dado.

Parámetros:

`músico`**:** Nombre del músico.

Salida: La cantidad y la lista de músicos por cada nivel de distancia hasta cubrir todos los niveles.

Ejemplo:

```
distancias Hendrix
3 Lennon Mercury Dio
15 Ward Powell Star Taylor May Page Plant Morse (...)
123 (...)
1234 (...)
2456 (...)
```

2.3.9 Comando: `subgrupos`

Formato: `subgrupos`

Descripción: Agrupa los nodos según las particiones de la red. Si el grafo es conexo todos los nodos pertenecerán a un grupo. Si el grafo estuviera partido en dos habrían dos grupos de nodos, en tres tres grupos, etc.

Salida: La cantidad de músicos en cada grupo ordenados en orden decreciente de integrantes.

Ejemplo:

```
subgrupos
123
50
4
2
```

```
2
1
1
1
```

2.3.10 Comando: `articulacion`

Formato: `articulacion`

Descripción: Calcula todos los puntos de articulación del grafo.

Salida: Lista todos los nodos que son puntos de articulación del grafo.

Ejemplo:

```
articulacion
Flavio
Ricardo
```

3 Criterios de aprobación

A continuación describimos criterios y lineamientos que deben respetarse en el desarrollo del trabajo.

3.1 Utilización de estructuras de datos

Para la realización de este trabajo es necesario utilizar una estructura grafo además de crear las estructuras adicionales que se consideren necesarias siempre justificando la complejidad necesaria para la operación.

Todas las estructuras deben estar implementadas de la forma más genérica posible y correctamente documentadas.

3.2 Programa

El programa debe cumplir los siguientes requerimientos:

- Debe estar adecuadamente estructurado y modularizado, utilizando funciones definidas de la forma más genérica posible, sin caer en lo trivial.
- El código debe ser claro y legible.
- El código debe estar comentado y las funciones definidas, adecuadamente documentadas.

- El programa debe compilar o ser interpretado sin advertencias ni mensajes de error, debe correr sin pérdidas de memoria[4], uso de valores sin inicializar, o errores en general.
- Además, claro, debe satisfacer la especificación de la consigna y resolverla en tiempo razonable.

3.3 Informe

El informe deberá consistir de las siguientes partes:

- **Carátula** con la información del alumno/a y el ayudante asignado (en caso de saberlo de antemano).
- **Análisis y diseño**: Describir la solución elegida para resolver cada problema propuesto, y cómo se lleva a cabo. En particular, mencionar cómo es el flujo del programa, qué algoritmos y estructuras de datos se utilizan y por qué, y cuál es el orden de ejecución en tiempo y espacio de cada operación.
- **Implementación**: Incluir aquí *todo* el código fuente utilizado (en formato `monoespaciado`, para facilitar su lectura).
- También *opcionalmente,* toda explicación adicional que consideren necesaria, referencias utilizadas, dificultades encontradas, cambios o mejoras que se podrían hacer a futuro y conclusiones.

El informe debe estar lo más completo posible, con presentación y formato adecuados. Por ejemplo, este enunciado cumple con los requerimientos de un informe bien presentado.

4 Entrega

El trabajo consiste en:

- El informe impreso.
- El informe digital, en formato `.pdf`
- Una versión digital de **todos** los archivos de código fuente, separados del informe, en un archivo comprimido (`.zip` o `.tar.gz`).

Los dos últimos deben enviarse a la dirección `tps.7541rw@gmail.com`, colocando como asunto:

```
TP3 - Padron - Apellido
```

[4]Si aplica considerando el lenguaje utilizado.

Se aclara que por código fuente se entiende todos los archivos `.h` y `.c`, el archivo `Makefile` para poder compilar en el caso de C y de todos los archivos `.py` en el caso de Python, etc., y todos los archivos adicionales que sean necesarios para ejecutar el programa. No deben entregarse nunca archivos `.o` u otros archivos compilados.

El informe impreso debe entregarse en clase. El plazo de entrega vence el **Viernes 5 de Diciembre de 2014**.