Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Estructuras de Datos y Algoritmos - Algoritmos I
Segundo Cuatrimestre de 2020

Trabajo Práctico No 2: Grafos

La resolución de este trabajo práctico debe ser enviada a través de GitHub Classroom.
El código provisto como parte de la solución a los ejercicios deberá estar documentado apropiadamente (por ejemplo, con comentarios en el código). Aquellas soluciones que no requieran programación, como así también la documentación adicional de código que se desee proveer, debe entregarse en archivos de **texto convencionales**, en el mismo repositorio de entrega de las soluciones, con nombres que permitan identificar fácilmente su contenido.
Los grupos para la resolución del trabajo deben ser conformados por dos o tres integrantes.
El trabajo práctico deberá ser entregado la semana previa al examen final al que se presente el primer integrante del grupo que rinda la materia. Una vez finalizado el trabajo, se pide informar por correo electrónico a los docentes de la materia para su corrección.

1. En economía y finanzas, el **arbitraje**, es el uso de discrepancias en los tipos de cambio de moneda para transformar una unidad de una moneda en más de una unidad de la misma moneda, esta tipo de operaciones son a veces conocidas con el nombre de *bicicleta financiera*.

   Por ejemplo, suponga que 1 dólar estadounidense (USD) compra $46,4$ rupias indias (INR), 1 rupia india compra $2,5$ yenes japoneses (JPY) y 1 yen japonés compra $0,00955$ dólares estadounidenses. Luego, al convertir monedas, un comerciante puede comenzar con 1 dólar estadounidense y comprar $46,4 \times 2,5 \times 0,00955 = 1,1078$ dólares estadounidenses, obteniendo así una ganancia del $10,78$ por ciento.
   Suponga que tenemos n monedas $c_1, c_2, ..., c_n$ y una tabla R de tipos de cambio $n \times n$, de manera que una unidad de moneda c_i compra $R[i, j]$ unidades de moneda c_j.

   Para resolver este problema se necesita un algoritmo eficiente que determine si existe una secuencia de ocurrencias $c_{i_1}, c_{i_2}, c_{i_k}$ tal que:

   $$R[i_1, i_2] \times R[i_2, i_3] \times \cdots \times R[i_{k-1}, i_k] \times R[i_k, i_1] > 1.$$

   Equivalentemente, usando logaritmos, esto puede expresarse como:

   $$\log R[i_1, i_2] + \log R[i_2, i_3] + \cdots + \log R[i_{k-1}, i_k] + \log R[i_k, i_1] > 0.$$

   Si multiplicamos por -1

   $$-\log R[i_1, i_2] - \log R[i_2, i_3] - \cdots - \log R[i_{k-1}, i_k] - \log R[i_k, i_1] < 0.$$

   Esta transformación permite utilizar algoritmos que buscan caminos con costos mínimos, como Warshall. Resumiendo, el algoritmo debe detectar ciclos negativos.

   En caso de existir la secuencia el algoritmo deberá retornar la diferencia porcentual entre el valor inicial y el valor obtenido, junto con la secuencia de monedas de la transacción.

   Se describe a continuación los archivos que se provee para la solución de este trabajo y que podrá encontrar en el repositorio git correspondiente a su grupo:

   - Una Interface Genérica `Grafo<V>`, que contiene la especificación de las operaciones públicas necesarias para manipular el tipo de datos *Grafo* .

- Una Clase `Moneda`, con información del país y la abreviatura o sigla que simboliza la moneda.
- Una Clase CargarDatos que permite cargar un grafo a partir de archivos conteniendo los datos de Monedas y la relación de cambio. En esta Clase se instancia el tipo genérico `<V>` con `<Moneda>`.
- Un archivo `monedas.csv` que contiene información asociada a las monedas de los diferentes países.
- Un archivo `cambio.csv` que contiene información de los diferentes cambios, tomados de distintas páginas web[1] a la fecha de la entrega de este enunciado.
- Una clase útil `Par`, representada por sus campos `primero` y `segundo` de tipos genéricos.

A continuación se detallan los ítems con la tarea a realizar para la solución de este trabajo práctico:

(a) Definir una implementación de la Interface `Grafo` dirigido y con peso, respetando siempre la genericidad del tipo `<V>`. No se aceptarán soluciones de la implementación de la clase `Grafo` que se definan para un tipo concreto. En el repositorio del trabajo encontrará, dentro del paquete colecciones, el template de la clase que debe implementar: *GrafoDirigido*.

(b) Proveer un método `cambioPositivo`, con el siguiente perfil:

```
public static Optional<Par<List<Moneda>, Double>> cambioPositivo(Grafo<Monedas> grafo)
```

El tipo de retorno es, en caso de existir, un par con las siguientes componentes: la primera componente es una secuencia de monedas que permite capitalizar el desequilibrio de precios y, como segunda componente, retorna la diferencia porcentual obtenida.
Este método permite resolver el problema planteado en el enunciado de arbitraje y para su definición se deberá hacer uso de la clase Grafo implementada en el inciso anterior.

(c) Proveer un método que permita, sólo en caso que no existan ciclos negativos, obtener el costo entre dos monedas dadas, con el siguiente perfil:

```
public static Optional<Par<List<Moneda>, Double>> costoTransaccion(Grafo<Monedas>
    grafo, Moneda inicio, Moneda fin)
```

Ayuda importante para la solución del trabajo:

- El algoritmo de grafos que deberá adaptar para resolver el problema de **arbitraje** es el algoritmo de *warshall* (para detectar ciclos negativos).
- Para que el algoritmo resultante sea eficiente, deberá usar la versión iterativa del algoritmo de warshall.

[1] `https://www.bloomberg.com/markets/currencies`, `https://www.exchange-rates.org/MajorRates.aspx`