Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-1

25 de abril de 2019
Actividad Sumativa

Actividad 04

Estructuras basadas en nodos: Parte 2

Introducción

El DCC[1] ha comenzado un nuevo ~~e innovador~~ emprendimiento de comercio internacional, **DCComercio**. El ~~desalmado~~ **Doctor Valdivieso** está en búsqueda de más mangas y novelas ligeras que leer así que intenta aprovecharse apoderándose de este nuevo proyecto.

Para eso te han contratado, como ávido programador y experto en estructuras nodales de grafos, para que le ayudes a modelar las rutas de abastecimiento mundial de varios productos.

1. DCComercio

Deberás modelar la red de comercio mundial, que corresponde a los intercambios (importaciones y exportaciones) de distintos productos entre países. Cada país es capaz de exportar uno o más productos hacia distintos países. A la vez pueden importar uno o más productos desde otros países. Según como importan y exportan productos, se definen cuatro posibles roles para un país, en relación a un producto intercambiado:

- **País con `producto` disponible**: son países que importan `producto` desde otro país o exportan `producto` hacia otro país.
- **País productor de `producto`**: son países que exportan `producto` hacia otro país pero que no lo importan desde un tercer país.
- **Cliente intermedio de `producto`**: son países que importan `producto` pero también lo exportan. Un país cliente intermedio no se considera productor de `producto`.
- **Cliente final de `producto`**: son países que importan `producto` pero no lo exportan.
- **País con `producto` no disponible**: son países donde `producto` no es obtenible dentro de sus fronteras. Es decir, no es importado ni exportado.

La Figura 1 muestra un ejemplo de la red de comercio. Siguiendo las definiciones, se puede notar que: Venezuela es productor de petroleo; Alemania es productor de autos; Chile es productor de cobre; Albania y Estonia son productores de oro. Por otro lado: Bélgica es cliente intermedio de oro; USA es cliente final de petroleo, cobre y autos; Dinamarca, España y Finlandia son clientes finales de oro. Hay muchos más roles en el ejemplo, como que Dinamarca no tiene disponible petroleo, cobre ni autos...

[1] *Departamento de Ciencias del Comercio*

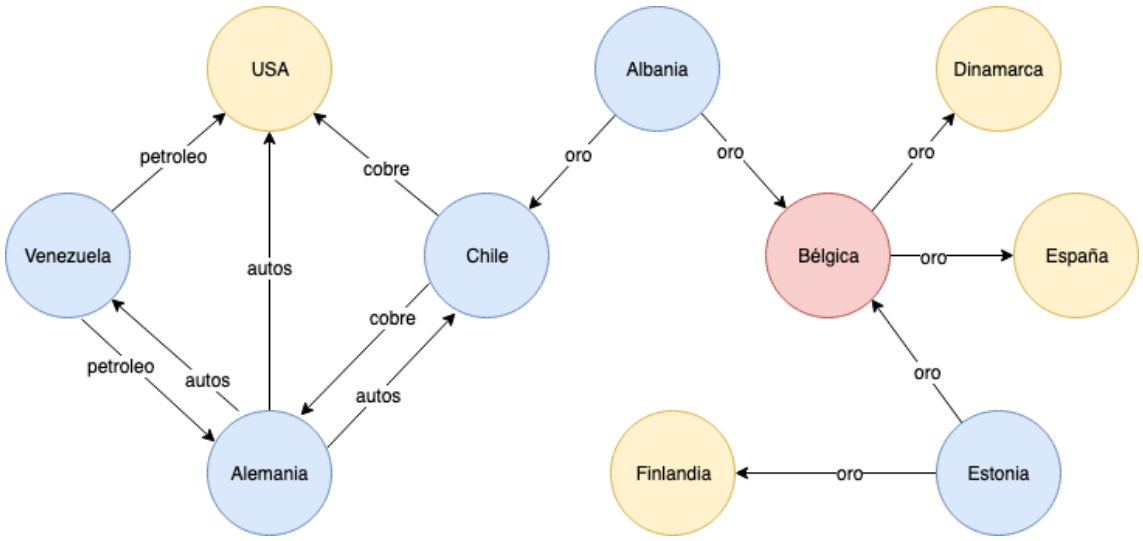

Figura 1: Ejemplo de grafo de red de comercio.

2. Nodos y aristas

Para modelar el problema, se utilizará una representación **basada en nodos**. Para lograrlo usarás dos clases incompletas, que encontrarás en `main.py`: `Pais` y `RedComercio`. Deberás completar estas clases y poblar el grafo a partir de información de *input* que se proveerá (esto se detalla en la sección 4). Tu representación debe cumplir con lo siguiente:

- Cada nodo en el grafo representa a un país, y las aristas del grafo representan intercambios de productos entre países, los que llamamos "exportaciones".
- La clase `Pais` modela un nodo del grafo, y posee dos atributos: `nombre` y `exportaciones`.
- El atributo `nombre` es un *string* correspondiente al nombre del país.
- El atributo `exportaciones` representa las aristas conectadas al nodo. `exportaciones` es una lista de tuplas donde cada tupla representa una exportación de productos hacia otros países.
- Cada tupla de `exportaciones` contiene el nombre del producto exportado (un *string*) y un nodo destino (un objeto `Pais`), en ese orden.

  Por ejemplo, para un país que exporta carbón a Eslovenia, pieles a Túnez y oro a Bélgica, su atributo `exportaciones` sería: `[("Carbón", eslovenia), ("Pieles", tunez), ("Oro", belgica)]`, donde `eslovenia`, `tunez` y `belgica` son instancias de la clase `Pais`.
- La clase `Pais` tiene dos métodos incompletos que debes implementar y serán de ayuda para tu objetivo.
  - `agregar_exportacion` recibe un *string* de producto y un objeto de tipo `Pais` que es el destino de la exportación. El método agrega una arista al atributo `exportaciones`.
  - `exporta_esto` recibe un *string* de producto y retorna un *booleano* que indica si el país exporta (`True`) o no (`False`) ese producto.
- El grafo se modela mediante la clase `RedComercio` la cual utiliza la clase `Pais` para representar los nodos internamente. Se le entregará una clase `RedComercio` cuyos métodos debe completar de acuerdo a lo indicado en la siguiente sección.

3. Grafo y consultas

La clase `RedComercio` representa un grafo de países y sus exportaciones. En su método `def __init__` se debe construir el grafo utilizando la clase `Pais` para cada uno de los países entregados y almacenarlos en algún atributo interno.

A continuación, debes implementar, para DCComercio, cuatro métodos en `RedComercio` que representan consultas a realizar. Te recomendamos hacerlas en orden pues el resultado de una consulta previa te puede ayudar para las siguientes. También te recomendamos revisar la distribución del puntaje al final del enunciado, la cual te puede orientar en cómo enfrentar las consultas.

1. `def productos_importados(self, nombre_pais)`: Función que retorna un *set* con todos los nombres de productos que son **importados** por el país de nombre `nombre_pais` (*string*).
2. `def productos_producidos(self, nombre_pais)`: Función que retorna un *set* con los nombres de todos los productos que **produce** el país de nombre `nombre_pais` (*string*). Recuerda que la definición de país **productor** está en la sección 1.
3. `def paises_clientes_finales(self, producto, nombre_pais)`: Función que recibe el nombre de un producto (*string*) y el nombre de un país (*string*). Si el producto está **disponible** en el país, la función debe retornar el *set* de los nombres de todos los **clientes finales** del producto que proviene del país especificado. Si el producto no está disponible en el país, entonces entrega un *set* vacío. **Esta consulta debe implementarse mediante un algoritmo de navegación de grafos.**

   Consideremos el grafo de la Figura 2 para la distribución mundial del oro (se omiten las aristas de otros productos). La consulta `red.paises_clientes_finales("oro", "Albania")` debe obtener a qué países llega el **oro** que posee **Albania**.

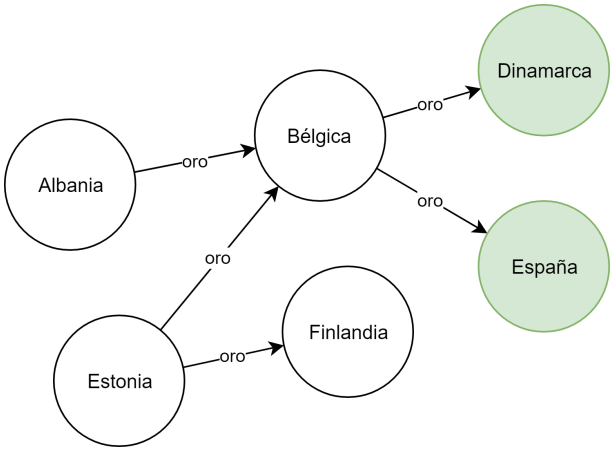

Figura 2: Ejemplo de red de comercio de solo el oro.

En este grafo Albania y Estonia producen oro; Bélgica es un cliente intermedio; y España, Dinamarca y Finlandia son clientes finales. Sin embargo, Finlandia no es cliente final de Albania, sino de Estonia, por lo que el resultado de la consulta `red.paises_clientes_finales("oro", "Albania")` debe ser el *set*: `{"Dinamarca", "España"}`.

4. Data

El primer paso, antes de construir las consultas sobre el grafo, debe ser poblar el grafo (agregar elementos). Para esto recibirás un archivo `exportaciones.json` con la información de todos los países y sus exportaciones, es decir, qué productos exporta y hacia dónde. No necesitas abrir dicho archivo ya que en `main.py` hay código que crea la variable `EXPORTACIONES` que almacena la información del archivo. La variable `EXPORTACIONES` se utiliza como parámetro para inicializar una instancia de la clase `RedComercio`.

`EXPORTACIONES` es un diccionario cuyas *keys* son los nombres de todos los paises y sus *values* son un diccionario con todas las exportaciones de dicho país. El diccionario de exportaciones de un país es de la forma `{pais_importador: producto}`. Un ejemplo de la variable `EXPORTACIONES` es:

```
EXPORTACIONES = {
    "Afganistán": {
        "Costa Rica": "Perlas",
        "Malta": "Uranio",
        "Eslovenia": "Carbón",
        "Túnez": "Pieles"
    },
    "Albania": {
        "Dinamarca": "Azúcar",
        "Fiyi": "Marfil",
        "Singapur": "Marfil",
        "Estonia": "Nitrato"
    },
    ...
}
```

Por lo tanto, las *keys* de `EXPORTACIONES` contienen los nombres de **todos** países en la red. Y en el ejemplo anterior, acceder `EXPORTACIONES["Afganistán"]` retornará el diccionario:

`{"Costa Rica": "Perlas", "Malta": "Uranio", "Eslovenia": "Carbón", "Túnez": "Pieles"}`

Lo que significa que Afganistán exporta perlas a Costa Rica, uranio a Malta, carbón a Eslovenia y pieles a Túnez. Lo que nos da la información base para las aristas que salen del nodo para Afganistán.

Notas

- El uso de los métodos para diccionarios `keys`, `values` e `items` les puede ser de ayuda para iterar sobre diccionarios usando `for`.
- Cuando se itera mediante `for` sobre una lista de tuplas de tamaño dos, es posible utilizar la siguiente notación para desempaquetar ambos componentes de la tupla:

```
lista_de_tuplas = [(1, 2), (3, 4), (5, 6)]
for primer, segundo in lista_de_tuplas:
    print(primer + segundo)
```

  Esto imprime:

```
3
7
11
```

Requerimientos

- (0.50 pts) Completar métodos de clase `Pais`.
- (1.50 pts) Poblar el grafo.
- (1.25 pts) Primera consulta: se retorna correctamente el *set* de productos.
- (1.25 pts) Segunda consulta: se retorna correctamente el *set* de productos.
- (1.50 pts) Tercera consulta:
  - (0.25 pts) Retorna un *set* vacío si el producto no está disponible en el país.
  - (0.25 pts) Retorna un *set* con solamente el nombre del país si este ya es un cliente final.
  - (1.00 pts) Para el resto de los casos retorna el *set* de países requeridos.

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC04/`
- **Hora del *push*:** 16:30