PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 - PROGRAMACIÓN AVANZADA

Actividad 03

1^o semestre 2018
5 de abril

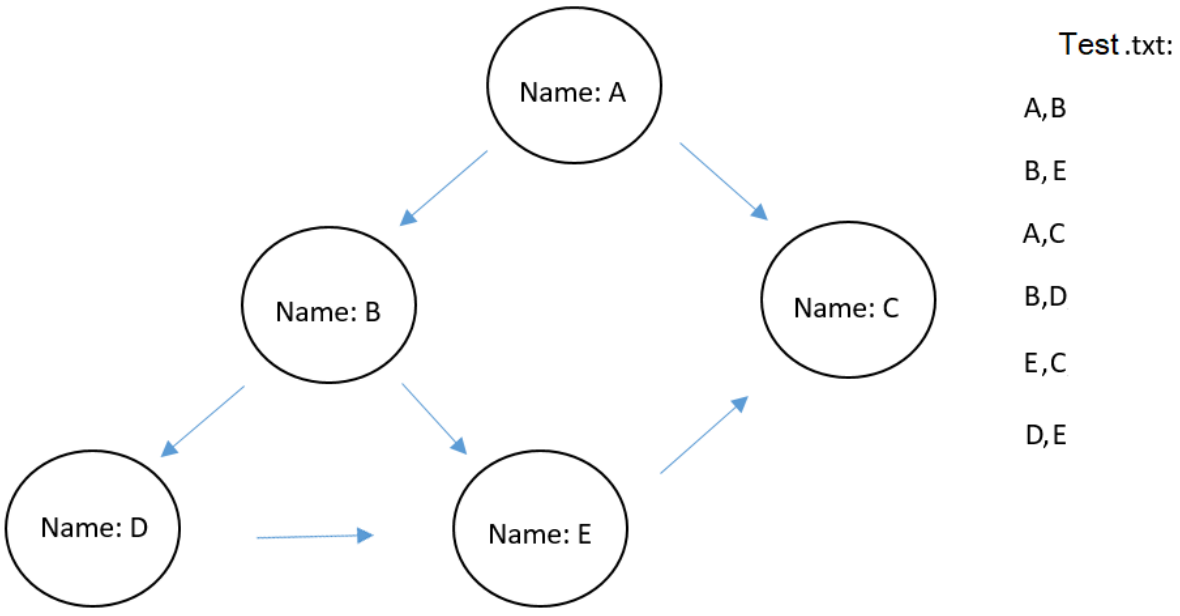
Estructura de datos

Introducción

Estás en problemas. Luego de una discusión con un familiar, tomas tu pistola interdimensional y huyes lo más lejos que puedes. Lamentablemente, caíste en el peor lugar posible y no puedes ir directo a casa porque tiraste la pistola, haciendo que trace una serie de rutas interconectadas que pueden llevarte a casa o a la muerte. Por suerte, tienes acceso a un conveniente (pero arcaico) computador con Python que puedes utilizar para encontrar el camino a casa.

Instrucciones

Se les entregará un archivo, donde cada línea tendrá el formato `origen,destino`. Este contiene todas las relaciones que deben presentarse en el grafo **dirigido**. A continuación, se presenta un ejemplo de la relación que se crea a partir de un archivo `test.txt`.



Se espera que tu programa sea capaz de leer cualquier archivo CSV con el formato mencionado y crear un grafo **dirigido** que pueda encontrar si existe un camino desde un nodo en particular a otro. Utilizando como ejemplo la imagen anterior, si se busca llegar a `C` desde `A` el *output* debiese ser `True`, pues sabemos que existen tres caminos entre ellos (`[A,B,D,E,C]`, `[A,C]`, `[A,B,E,C]`). Si quisiéramos buscar el camino más corto entre ese par de nodos, deberíamos retornar `[A,C]`.

Se espera que la clase `Grafo` tenga, como mínimo, los siguientes métodos:

- `agregar_conexion(origen, destino)` Recibe como parámetro el nombre del nodo de origen y el nombre del nodo de destino entre los que se debe crear una conexión. En caso de que uno de los nodos no se encuentre en el grafo, este debe ser creado y luego se debe establecer la conexión entre ambos.
- `quitar_conexion(origen, destino)` Recibe el nombre del nodo de origen y del nodo de destino al cual se eliminará la conexión entre ambos sin borrar las relaciones que pueda tener el nodo de destino.
- `encontrar_camino(origen, destino)` Recibe el nombre del nodo de origen y del nodo destino para retornar `True` en caso de que exista un camino entre los nodos, y `False` en caso contrario. En caso de que el origen y el destino sean el mismo, se debe retornar `True`.
- `exportar_csv(path_archivo)` Recibe un *path* donde deberá ser generado un archivo CSV con el estado actual del grafo. El formato del archivo debe ser igual al mencionado en las instrucciones.

Para probar su programa, junto con el enunciado se han subido tres archivos con **tres** grafos de distintos tamaños. En el anexo (al final de este enunciado), se muestra el *output* esperado con las consultas incluidas en un `main.py` que se entrega. Esto es sólo para facilitarles el *testing*; tener bueno el *output* en esos grafos en particular no es sinónimo de tener un siete. Los archivos de prueba al momento de corregir serán distintos.

Notas

- La solución puede ser tanto iterativa como recursiva.
- Sea ordenado y escriba funciones auxiliares si las necesita.
- En esta actividad debe tomar en cuenta los conocimientos adquiridos respecto a las estructuras de datos que nos provee Python. No utilizar estructuras adecuadas, significará no obtener todo el puntaje.

Bonus

Saber que puedes llegar a casa no es lo mismo que saber cómo llegar, ¿no es así? Como bonus se te pide agregar la función `encontrar_camino_corto(origen, destino)` a tu clase `Grafo` y retornar el camino más corto (si es que existe) entre ese par de nodos. En caso contrario, retornar `None`. En el caso de pedir un camino desde un nodo hacia sí mismo se debe retornar una lista vacía `[]`.

Requerimientos

- (6,00 pts) Clase `Grafo`
  - (1,20 pts) Cargar grafo desde archivo `CSV`
  - (1,00 pts) Crear `agregar_conexion(origen, destino)`
  - (2,40 pts) Crear `encontrar_camino(origen, destino)`
    - (0,40 pts) Camino a distancia 1
    - (0,40 pts) No existe el camino
    - (0,60 pts) Camino a distancia mayor a uno

        - (1,00 pts) Camino con ciclos
    - (1,00 pts) Guardar grafo en archivo `CSV`
    - (0,40 pts) Crear `quitar_conexion(origen, destino)`
- (0,50 pts) Bonus
    - Encontrar el camino óptimo e imprimirlo
    - Obtener calificación sobre 5.0

Entrega

- **Lugar:** En su repositorio de GitHub en la **carpeta** `Actividades/AC03/`
- **Hora:** 16:20

Anexo

Se te entrega un `main.py` que carga 3 grafos, uno de dificultad fácil, uno intermedio y otro más difícil. Además, se realizan ciertas consultas cuyo *output* esperado es:

```
********************EASY********************
True
False


True

True

False
True

********************MEDIUM********************
True
True

False

********************HARD********************

True


False
True
```

Si implementas el *bonus* deberías ver este *output*, aunque es posible que los caminos sean distintos pero del mismo largo que los propuestos:

```
********************EASY********************
True
False
[A, B, E]
[A, C]
True
[A, B, E, C]
True
[A, B, D, E, C]
False
True

********************MEDIUM********************
True
True
[A, F, G]
False

********************HARD********************
```

```
[A, 0, 4, 5, L, Z]
True
[A, 0, 4, Z]
[A, 0, 4, 5, L, Z]
False
True
[X, B, T, L, Z]
```

En ambos casos se crea un archivo `easy_output.txt` con el siguiente contenido (las líneas pueden estar en cualquier orden):

```
A,B
A,C
B,D
E,C
```