Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2021-1

10 de Junio de 2021
Actividad Sumativa

Actividad Sumativa 3

Estructuras Nodales I y II

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AS3/`
- **Hora del *push*:** 16:30

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

Eres uno de los mejores detectives del mundo, llamado DCCherlock Holmes, con una habilidad imparable. Tu último caso consiste en detener la gran banda de mafiosos DCConda, que tienen el objetivo de dominar el mundo digital.

Para esta misión irás de encubierto con el fin de desmantelar la mafia. Pero no será un hazaña simple, y para ello, tienes que aprovechar de escuchar sus conversaciones y relaciones para llegar al secretario, quien posee la información valiosa, y así encontrar el mapa de su organización. Finalmente con toda esa información deberás desmantelarla, todo esto haciendo uso de las estructuras nodales.

Flujo del programa

El programa consta de tres partes: en la primera, deberás recorrer una lista ligada de nodos que contienen a los mafiosos, donde cada conexión representa que un mafioso delata a otro, hasta encontrar al secretario de la mafia, el cual podrás identificar por su característica frase. Una vez que lo encuentres, este te dará

un diccionario con todos los lugares en que se reúne la mafia y qué mafiosos frecuentan dichos lugares. Además, te dará las conexiones que hay entre estas ubicaciones (la mafia cuenta con un complejo sistema de túneles conectando sus cuarteles). A partir de este diccionario, deberás crear un grafo no dirigido, donde cada nodo será un lugar de reunión. Finalmente, deberás recorrer este grafo, con el fin de hallar el lugar donde se encuentran los líderes, atraparlos, y ponerle fin a este malvado complot.

Archivos

La información de los mafiosos, lugares y sus conexiones se encuentran en los archivos `mafiosos.csv`, `lugares.csv` y `conexiones.csv`. Su información es cargada por el archivo `cargar_archivos.py`, el cual ya se encuentra implementado. También se cuenta con un archivo `parametros.py`.

Los archivos que deberás modificar son: `nodo_chismoso.py`, `nodo_lugar.py` y `recorrido_mafia.py`. En estos deberás implementar los algoritmos de construcción y búsqueda en listas ligadas y grafos. Finalmente, el módulo a ejecutar será `main.py`, el cual ya se encuentra implementado.

Parte 1: Nodo Chismoso

En esta parte deberás encontrar al secretario de la mafia, el cual te ayudará a dar con el mapa completo de la organización. Para ello, ya lograste atrapar a uno de sus miembros, pero sabes que son bastante leales y que incluso con tus grandes habilidades, solo lograrás que te delaten a un solo compañero que aún no has interrogado. De esta manera, se construyó una cadena de delataciones que se puede expresar como una lista ligada. En ella, cada nodo representa algún mafioso y cada conexión representa quién delata a quién. Dentro de esta lista ligada, cualquiera de los nodos podría ser el secretario. Tu misión será encontrarlo, identificándolo con su característica frase. Para esto, deberás editar el archivo `nodo_chismoso.py`, el cual contiene la siguiente clase, y funciones:

- `class NodoChismoso`:
  - `def __init__(self, nombre: str, frase: str)`: Recibe los argumentos `nombre` y `frase` y guarda cada uno en un atributo.
    - `self.nombre`: Guarda el nombre del mafioso.
    - `self.frase`: Guarda su frase.
    - `self.conocido`: Guarda un nodo que contiene al mafioso que conoce (es decir quién sigue en la lista ligada).

    Tu tendrás acceso solo a una instancia de esta clase, la cual representará a la cabeza de la lista ligada. No debes modificarlo
- `def construir_nodo_chismoso(nombre: str, diccionario_mafiosos: dict)-> NodoChismoso`: Recibe el nombre de un nodo chismoso y un diccionario con la información de todos los mafiosos con llave el nombre del mafioso, y de valor su información. Con esto crea un `NodoChismoso`. que corresponda a un mafioso, para luego llamar recursivamente a esta misma función con el nodo siguiente. La recursión termina cuando ya no haya nodo siguiente. Retorna el nodo, en caso de existir. No debes modificarlo
- `def encontrar_secretario(nodo: NodoChismoso) -> NodoChismoso`: Esta función debe retornar el secretario mafioso, que se encuentra en algún lugar de la lista ligada. Para ello deberás recorrer la lista hasta identificar al secretario y retornarlo. Al revisar cada mafioso, deberás imprimir su nombre y su frase, guardadas en los atributos de la clase. Por ejemplo, un `print` puede verse como:

```
"cristobalnic: Del hambre nació la picardía"
```

Posteriormente deberás verificar si el mafioso actual es el secretario comparando su frase con el parámetro `frase_secretario`, encontrado en el archivo `parametros.py`[1], y retornarlo en caso de haberlo encontrado. En caso contrario, deberás continuar revisando el resto de la lista ligada. **Debes modificarlo**

Parte 2: Crear Mapa Mafia

Ahora que ya encontraste al secretario, este te entrega los lugares donde se reúne la mafia, junto a los miembros que los frecuentan. Además, te entrega las conexiones entre estos lugares. Usando las funciones y clases de `nodo_lugar.py`, deberás crear un grafo que represente este mapa:

- `class NodoLugar`: Esta clase representa a cada uno de los lugares donde se reúne la mafia. Cada instancia es un nodo que tendrás que utilizar para crear el grafo mencionado.
  - `def __init__(self, nombre: str)`: Inicializa el nodo y sus atributos. **No debes modificarlo**
    - `self.nombre`: Guarda el nombre del lugar.
    - `self.mafiosos`: Se instancia como una lista vacía. En este atributo se guardarán la lista de mafiosos correspondientes a él. Será una lista de `namedtuple` con los atributos `nombre` y `frase`.
    - `self.conexiones`: Se instancia como una lista vacía. Contiene a las conexiones del nodo como una lista de `namedtuple` con los atributos `destino` y `peso`.
  - `def agregar_conexion(self, destino: NodoLugar, peso: int)`: Método usado para agregar una conexión entre este nodo y el nodo `destino`, de peso `peso`. Agrega una `namedtuple` con los atributos `vecino` y `peso` a la lista. El peso solo será usado si planeas hacer el bonus. **No debes modificarlo**
  - `def __str__(self) -> str`: Método que puedes usar para comprobar cómo se ve el nodo. **No debes modificarlo**

Por último, deberás implementar la siguiente función:

- `def crear_grafo(diccionario_lugares: dict, conexiones: list) -> NodoLugar`: En esta función deberás crear el grafo. Para ello, primero deberás instanciar los nodos a partir de la información del diccionario `diccionario_lugares`, el cual contiene como llave el nombre del lugar y como valor una lista de instancias de la `namedtuple Mafioso`, cuyos atributos son `nombre` y `frase`. Una vez instanciado el nodo, deberás guardar sus mafiosos en el atributo `self.mafiosos`.

  Luego, deberás agregar los arcos del grafo a partir de la lista `conexiones`, la cual contiene los nombres de dos nodos conectados entre ellos y el peso de dicha conexión. Para esto, en ambos nodos (recuerda que el grafo es no dirigido) deberás usar el método `agregar_conexion`, entregando el nodo a conectar y el peso de dicha conexión. Finalmente, deberás retornar algún nodo cualquiera de los instanciados previamente. **Debes modificarlo**

Parte 3: A desmantelar la Mafia

Por último, tras obtener toda la información que necesitas, deberás encargarte de desmantelar la mafia completa. Para ello deberás recorrer el grafo de lugares y en cada nodo buscar si se encuentra o no uno de los dos líderes de la mafia, identificables por sus frases, que deben ser las mismas que `frase_lider_1` o

[1] Este ya se encuentra importado en la línea 1

`frase_lider_2`, indicadas en `parametros.py`. En esta parte, deberás modificar las siguiente función del archivo `recorrido_mafia.py`.

- `def recorrer_mafia(inicio: NodoLugar) -> list`: En esta función deberás implementar algún algoritmo de recorrido de grafo de los estudiados (DFS o BFS). Cada vez que visites un nodo, deberás imprimir un mensaje del tipo `f"Se ha desmantelado {lugar.nombre}"`, y revisar entre sus mafiosos si alguno es líder de la mafia. Para identificar esto, deberás chequear que su frase sea `frase_lider_1` o `frase_lider_2`. Recuerda que cada elemento de la lista de mafiosos del nodo es una `namedtuple`, con los atributos `nombre` y `frase` y que las conexiones de cada nodo son una `namedtuple`, con los atributos `peso` y `vecino`. Una vez hayas recorrido todos los nodos, deberás retornar una lista con los nodos de los lugares en donde se encuentran estos líderes. **Debes modificarlo**

Bonus

Finalmente... ¡Estamos a salvo de la mafia! Sin embargo, recuerdas que estas organizaciones siempre resurgen, por lo que te gustaría analizar la información de los lugares investigados. En particular te gustaría saber qué tan peligroso es el recorrido entre los dos nodos donde se encontraban los/as líderes de la mafia, y hacerlo lo menos peligroso posible (*peligro* corresponde al atributo `peso` del las conexiones). Para ello, deberás implementar la siguiente función en el archivo `recorrido_mafia.py`, y **descomentar las lineas 37 y 38** de `main.py`.

- `def minima_peligrosidad(inicio: NodoLugar, termino: NodoLugar) -> int`: En esta función, deberás encontrar el nivel de peligrosidad de la ruta de menor peligro entre los nodos `inicio` y `termino`, tomando en cuenta que el nivel de peligrosidad de cada conexión corresponde a su peso. Para ello, deberás implementar de manera inteligente un método que permita encontrar el **nivel de peligro mínimo** entre dos nodos.[2] Finalmente, deberás retornar el nivel de peligrosidad del camino.

Notas

- La parte 1 es independiente de las 2 y 3.
- Puedes probar tu avance corriendo los archivos directamente.
- Recuerda que si deseas hacer el **BONUS**, debes **DESCOMENTAR** las líneas 37 y 38 del `main.py`
- Ojo: En el main, **sólo** la parte 1 indica si tu retornas lo correcto.

Requerimientos

- (2.00 pts) **Parte 1**: Función `encontrar_secretario`.
  - (0.50 pts): Imprimir el nombre de cada mafioso revisado con su frase.
  - (1.00 pts): Itera correctamente sobre la Lista Ligada.
  - (0.50 pts): Encontrar al secretario.
- (2.50 pts) **Parte 2**: Función `crear_grafo`.
  - (0.75 pts): Instanciar cada nodo correctamente.

[2] Puedes buscar en internet ayuda para resolver este problema, en particular, buscar sobre Dijkstra en internet te puede ayudar.

- (0.50 pts): Asigna los mafiosos a cada lugar.
- (0.75 pts): Conecta los nodos entre sí.
- (0.50 pts): Retornar nodo cualquiera.

- (1.50 pts) **Parte 3**: Función `recorrer_mafia`.
  - (0.75 pts): Usa clara y correctamente BFS o DFS.
  - (0.25 pts): Se hace el print indicado en el enunciado cada vez que visitas un nodo.
  - (0.50 pts): Encuentra y retorna los lugares en los que están los líderes correctamente.
- (0.50 pts): **Bonus**.