Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2020)

Actividad Práctica 2 - Estructuras de datos y algoritmos

Objetivos

- Aplicar los contenidos de estructuras de datos y algoritmos para resolver un problema aplicado.

Entrega

- **Lenguaje a utilizar:** Python 3.6 o superior
- **Lugar:** repositorio privado en GitHub. Recuerde incluir todo en una carpeta de nombre **P02**.
- **Entrega: lunes 14 de septiembre a las 23:59 hrs.**
- **Formato de entrega:** archivo python notebook (**P02.ipynb**) y archivo python (**P02.py**) con la solución de los problemas. Ambos archivos deben estar ubicados en la carpeta **P02**. No se debe subir ningún otro archivo a la carpeta. El archivo **py** sólo debe incluir la definición de las funciones y la importación de los módulos necesarios. No debe incluir en el **py** ejemplos de ejecución ni la ejecución de dichas funciones. **No deje instancias del método `print()` en el archivo py**. En el **ipynb** utilice múltiples celdas de texto y código para facilitar la revisión de su programa.
- **NO SE ADMITEN ENTREGAS FUERA DE PLAZO**
- **Entregas con errores de sintaxis y/o que generen excepciones serán calificadas con nota 1.0.**
- Las *issues* del Syllabus en GitHub son parte de este enunciado.

Introducción

Las grandes ciudades del mundo están preocupadas en conocer como se movilizan las personas dentro de ellas para planificar de mejor forma el sistema de transporte. El auge de las nuevas tecnologías ha permitido contar con nuevas herramientas que facilitan la predicción de comportamiento. En concreto, algunas ciudades han instalado sensores Bluetooth en las intersecciones de la red vial con el objetivo de predecir por donde viajan sus habitantes. En la Figura 1 se muestra un ejemplo de red vial dirigida, donde se marcan con círculos azules las intersecciones que cuentan con un sensor Bluetooth instalado (las otras no tienen nada instalado).

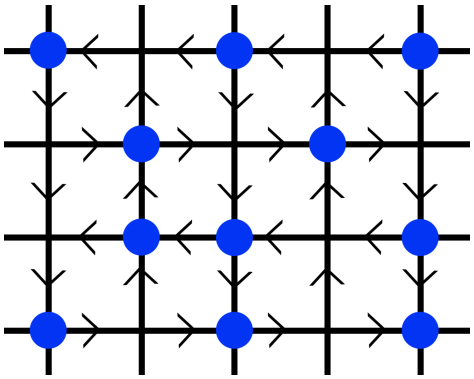

Figure 1: Red vial con algunos sensores Bluetooth

Para poder desarrollar modelos que usen los datos Bluetooth y predecir los viajes, es necesario realizar una simplificación que permita asignar los viajes correctamente y sin ambigüedades.

Descripción del problema

El objetivo es programar un algoritmo que reciba una red como la de la Figura 1 y retorne una nueva red simplificada. Para realizar la simplificación se deben cumplir las siguientes reglas:

1. Los nodos de la red simplificada corresponden solamente a los nodos Bluetooth de la red original.
2. Para todo par de nodos **A** y **B** de la red simplificada, existe un arco que va desde **A** a **B** en la red simplificada, si y solo si, es posible ir desde **A** a **B** en la red original sin pasar por un sensor Bluetooth.

Usando esta lógica, al aplicar estas reglas sobre la red de la Figura 1, resulta la red simplificada de la Figura 2.

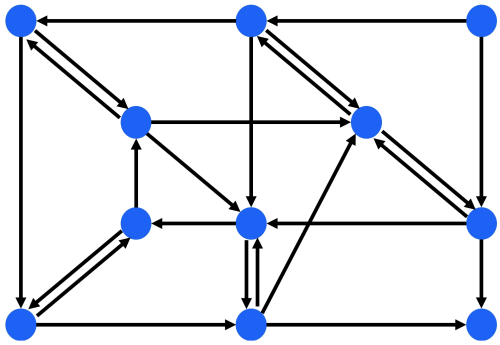

Figure 2: Red simplificada construida a partir de la Figura 1

Formato de *inputs* y *outputs*

Nodos

Un nodo está representado por una tupla de dos elementos **(nombre, tiene_sensor)**, donde **nombre** es el nombre del nodo como texto y **tiene_sensor** es una variable binaria que vale 1 si el nodo tiene un sensor Bluetooth y 0 si no.

Arcos

Un arco esta representado por una tupla de dos elementos **(nodo_origen, nodo_destino)**, donde **nodo_origen** es el nombre del nodo de origen y **nodo_destino** es el nombre del nodo de destino.

Red Original

Una red queda determinada por una lista de nodos y una lista de arcos. Por ejemplo, para la red de la Figura 3, vemos 4 nodos: **A**, **B**, **C** y **D** y 4 arcos: **AC**, **CD**, **BD** y **BA**.

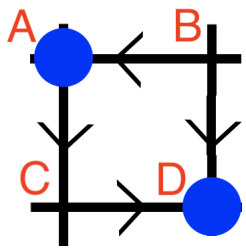

Figure 3: Red original de ejemplo

La lista de nodos sería `[('A',1), ('B',0), ('C',0), ('D',1)]` y la lista de arcos `[('A', 'C'), ('C','D'), ('B','D'), ('B','A')]`. Por tanto la representación final de la red corresponde a una lista

que contiene la lista de nodos y la lista de arcos.

```
lista_nodos = [('A',1), ('B',0), ('C',0), ('D',1)]
lista_arcos = [('A', 'C'), ('C','D'), ('B','D'), ('B','A')]
red_original = [[('A',1), ('B',0), ('C',0), ('D',1)], [('A', 'C'), ('C','D'), ('B','D'), ('B',
```

Red Simplificada

Una red queda determinada por una lista de nodos y una lista de arcos. Por ejemplo, la red de la Figura 3 se simplifica como la mostrada en la Figura 4 que tiene dos nodos y un arco.

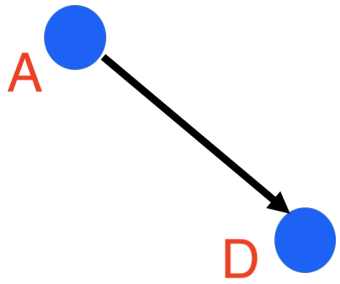

Figure 4: Red simplificada de la red de la Figura 3

La lista de nodos sería `[('A',1), ('D',1)` y la lista de arcos `[('A', 'D')]`. Por tanto la representación final de la red corresponde a una lista que contiene la lista de nodos y la lista de arcos.

```
lista_nodos = [('A',1), ('D',1)]
lista_arcos = [('A', 'D')]
red_original = [[('A',1), ('D',1)], [('A', 'D')]]
```

Función para leer redes

Las redes a simplificar se encuentran codificadas en archivos de texto de formato **.txt**. Para poder leerlas, usted cuenta con la función `leer_red(archivo_txt)` que retorna la red en el formato descrito. Por ejemplo, si usted desea trabajar con la *red_ejemplo.txt* basta que ejecute la función `leer_red('red_ejemplo.txt')` y este retorna la red contenida en el archivo con el formato descrito anterriormente.

Línea de trabajo

Siga las siguientes instrucciones para facilitar el desarrollo de esta actividad. Dentro de la carpeta **P02** del Syllabus usted va a encontrar un archivo **P02.ipynb** y uno **P02.py**, trabaje sobre ellos para implementar su solución. Además, encontrará algunos archivos **.txt** con ejemplos de redes.

1. Cree una función con el nombre `simplificar_red(red)` que reciba una red original y que retorne una red simplificada. (5 puntos) **HINT:** Pueden serle de utilidad las librerías `itertools` y `collections`.

   (a) Cree una función que retorne el subconjunto de nodos de la red original que solo poseen sensores Bluetooth (Regla 1 de construcción de la red simplificada). (0.5 puntos)

   (b) Cree una función que reciba como *inputs* dos nodos de la red original (si lo desea, esta función puede recibir más inputs) y retorne `True` si se cumple la Regla 2 o `False` si no. (2.5 puntos)

   (c) Cree una función que itere sobre todos los pares de nodos obtenidos en (a) y que retorne los arcos que pertenecen a la red simplificada. (1.5 puntos)

   (d) Use las 3 funciones anteriores para obtener la lista de nodos y arcos de la red simplificada y retornarlos en el formato indicado. (0.5 puntos)

2. Su solución contenida en el archivo **.py** será revisada con diez *inputs* diferentes para evaluar su correctitud. En este caso no importa el tiempo de ejecución siempre y cuando no supere un límite de 30 segundos (este tiempo es solo para definir un límite en caso *loops* infinitos o tiempos excesivos). Si obtiene el resultado correcto en los diez *inputs* tendrá todo el puntaje, de lo contrario no tendrá puntaje en este apartado. (1 punto)

Política de Integridad Académica

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de

la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.