LABORATORIO No. 9: Introducción a Grafos

Objetivos

Comprender la implementación del Tipo Abstracto de Datos Grafo (graph) y su uso para la solución de problemas.

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Identificar las operaciones principales implementadas en el TAD graph.
- Entender el funcionamiento del Grafo como estructura de datos y su impacto en los órdenes de crecimiento espacial y temporal.
- Proponer la forma de utilización de un Grafo como alternativa de solución a un problema que implican el manejo relaciones matriciales.
- Integrar los Grafos con las otras estructuras de datos vistas en el curso.

Fecha Límite de Entrega

Miércoles **17 de noviembre** antes de la media noche (11:59 p.m.).

Preparación del Laboratorio

- Revisar el API del TAD Grafo ubicado en **`DISClib\ADT\graph.py`**
- Revise los archivos **`graphstructure.py`** y **`adjlist.py`** ubicados en **`DISClib\DataStructures`**

Trabajo Propuesto

PASO 1: Copiar el ejemplo en su organización

Copie/Haga **Fork** del repositorio del laboratorio en su organización con el procedimiento aprendido en las prácticas anteriores.

El repositorio del proyecto base que utiliza este laboratorio es el siguiente:

- https://github.com/ISIS1225DEVS/ISIS1225-SampleGraph.git

Antes de clonar el repositorio en su computador diríjase a su organización (Ej.: ***EDA2021-1-SEC02-G01*** para el primer grupo de la sección 2 del curso) y cambie el nombre del repositorio de acuerdo con el esquema **LabGraph-S<<XX>>-G<<YY>>** donde **XX** es el número de la semana de la práctica y donde **YY**

es el número del grupo de trabajo. (Ej.: **LabGraph-S02-G01** para este **decimo laboratorio** hecho por el **grupo 1** de la **sección 2).**

Recuerde que **NO necesita** agregar la sección o el semestre en este nombre porque ya está identificado en su organización.

PASO 2: Descargar el ejemplo

Después de renombrar el proyecto dentro de su organización ya puede clonar el proyecto. Descargue el código en su máquina local **(`git clone`)** siguiendo lo aprendido en las practicas anteriores.

Recuerde modificar el **`README`** del repositorio para incluir los nombres de los integrantes del grupo.

PASO 3: Entender la fuente de datos y la construcción del Grafo

Antes de iniciar a explorar y modificar el ejemplo, recuerde descargar los datos de trabajo **`singapur_bus_routes`** disponibles en el portal oficial del curso en BrightSpace. Descargue el **Zip,** descomprímalo y guarde los archivos CSV en la carpeta **`*/Data/`** de su copia local de código.

El conjunto de datos contiene información de rutas de buses, paraderos y distancias entre los paraderos del sistema de buses de la ciudad de Singapur, y fue tomado del siguiente enlace: https://www.kaggle.com/gowthamvarma/singapore-bus-data-land-transport-authority

Los archivos de rutas tienen la siguiente información:

- ServiceNo,
- Operator,
- Direction,
- StopSequence,
- BusStopCode,
- Distance,
- WD_FirstBus,
- WD_LastBus,
- SAT_FirstBus,
- SAT_LastBus,
- SUN_FirstBus,
- SUN_LastBus

Para crear el grafo se utilizan los archivos con las rutas y la secuencia de paraderos, así como la distancia entre paradas. Tenga en cuenta que una misma parada puede servir a más de una ruta, por lo cual los vértices del grafo tendrán la siguiente estructura: `<BusStopCode>-<ServiceNo>.`

Por ejemplo: `‘75009-10’` para indicar que esa parada es para la ruta 10 y `‘75009-101’` para indicar que esa misma parada también sirve a la ruta 101.

Los arcos, representan segmentos de ruta que comunican dos paradas: como peso de los arcos se tiene la distancia entre las dos estaciones.

Por último, es importante resaltar que el **grafo es dirigido**, dado que las rutas tienen una dirección específica entre las estaciones.

PASO 4: Ejecutar y explorar el ejemplo

El proyecto **`SampleGraph`** busca familiarizarlos con el TAD Grafos (graph) y su uso para solucionar problemas y una forma de probar su desempeño en aplicaciones MVC.

Diríjase al archivo **view.py** y ejecútelo, y seleccione secuencialmente la **opción 1** y **2** para iniciar el analizador y cargar información respectivamente.

```
****************************************
Bienvenido
1- Inicializar Analizador
2- Cargar información de buses de singapur
3- Calcular componentes conectados
4- Establecer estación base:
5- Hay camino entre estacion base y estación:
6- Ruta de costo mínimo desde la estación base y estación:
7- Estación que sirve a mas rutas:
0- Salir
****************************************
Seleccione una opción para continuar
```

```
****************************************
Bienvenido
1- Inicializar Analizador
2- Cargar información de buses de singapur
3- Calcular componentes conectados
4- Establecer estación base:
5- Hay camino entre estacion base y estación:
6- Ruta de costo mínimo desde la estación base y estación:
7- Estación que sirve a mas rutas:
0- Salir
****************************************
Seleccione una opción para continuar

Cargando información de transporte de singapur ....
Numero de vertices: 13535
Numero de arcos: 32270
El limite de recursion actual: 1048576
```

Al cargar la información, verá el número de vértices y arcos cargados en el grafo; así como el número de llamados recursivos de Python y el tiempo de ejecución.

En relación con el límite de recursión, responda a las siguientes preguntas y registre su respuesta en el documento de observaciones:

a. ¿Qué instrucción se usa para cambiar el límite de recursión de Python?
b. ¿Por qué considera que se debe hacer este cambio?
c. ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?

Ahora, ejecute la **opción 4**; esta operación calcula la ruta más corta desde la estación indicada a todas las otras estaciones (todos los vértices del grafo). Al ejecutar la opción utilice como vértice de entrada **75009-10**, debe aparecer un resultado similar al siguiente:

```
*****************************************
Bienvenido
1- Inicializar Analizador
2- Cargar información de buses de singapur
3- Calcular componentes conectados
4- Establecer estación base:
5- Hay camino entre estacion base y estación:
6- Ruta de costo mínimo desde la estación base y estación:
7- Estación que sirve a mas rutas:
0- Salir
*****************************************
Seleccione una opción para continuar
>4
Estación Base: BusStopCode-ServiceNo (Ej: 75009-10): 75009-10
Tiempo de ejecución: 31.46489560000009
```

Ahora ejecute esta misma opción con los **nueve (9) diferentes archivos de datos** que se encuentran en el directorio `Data`, como podrán verificar el nombre del archivo indica aproximadamente el número de líneas del archivo CSV. Para este fin cambien en el archivo **`view.py`**, de archivo, comenzando con el más pequeño y luego con el siguiente en tamaño hasta probar con todos.

Registre número de vértices, el número de arcos del grafo, y el tiempo que toma esta instrucción con cada uno de los archivos CSV en el documento de observaciones del laboratorio.

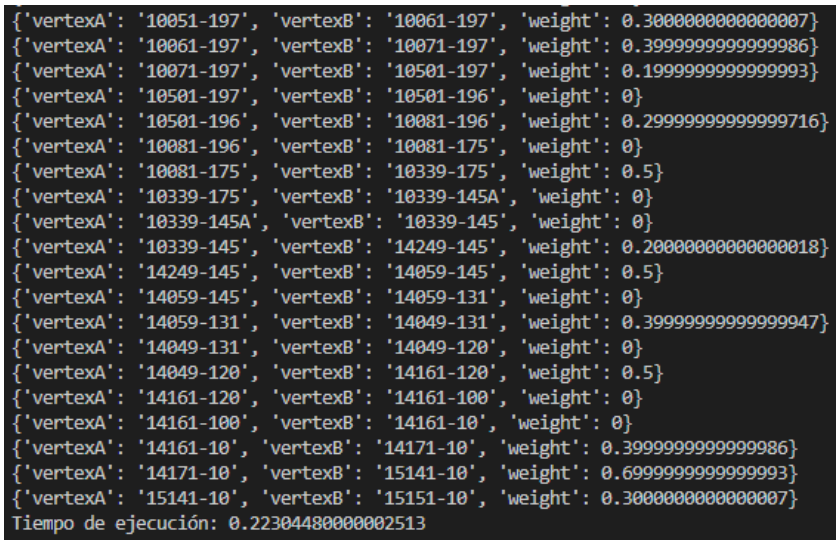
Nota: Deben ejecutar siempre las opciones 1 y 2 antes de usar la opción 4.

Responda la siguiente pregunta y registre su respuesta en el documento de observaciones:

d. ¿Qué relación creen que existe entre el número de vértices, arcos y el tiempo que toma la operación 4?

A continuación, ejecuten la **opción 6**; esta operación calcula el camino mínimo para ir desde la estación base a otra estación. Al ejecutar la opción utilice como vértice de entrada **`15151-10`**, debe aparecer un resultado similar al siguiente:

```
*****************************************
Bienvenido
1- Inicializar Analizador
2- Cargar información de buses de singapur
3- Calcular componentes conectados
4- Establecer estación base:
5- Hay camino entre estacion base y estación:
6- Ruta de costo mínimo desde la estación base y estación:
7- Estación que sirve a mas rutas:
0- Salir
*****************************************
Seleccione una opción para continuar
>6
Estación destino (Ej: 15151-10): 15151-10
El camino es de longitud: 136
{'vertexA': '75009-10', 'vertexB': '75009-127', 'weight': 0}
{'vertexA': '75009-127', 'vertexB': '75009-127A', 'weight': 0}
{'vertexA': '75009-127A', 'vertexB': '75009-19', 'weight': 0}
{'vertexA': '75009-19', 'vertexB': '75009-20', 'weight': 0}
{'vertexA': '75009-20', 'vertexB': '75009-22', 'weight': 0}
{'vertexA': '75009-22', 'vertexB': '76059-22', 'weight': 0.6}
{'vertexA': '76059-22', 'vertexB': '76111-22', 'weight': 0.20000000000000007}
{'vertexA': '76111-22', 'vertexB': '76121-22', 'weight': 0.30000000000000004}
{'vertexA': '76121-22', 'vertexB': '76281-22', 'weight': 0.2999999999999998}
{'vertexA': '76281-22', 'vertexB': '75069-22', 'weight': 0.8000000000000003}
```



Ahora, ejecuten el programa con cada uno de los archivos CSV. Por cada archivo, tomen nota del número de vértices, el número de arcos del grafo, y el tiempo de ejecución que toma la opción 6.

PASO 5: Estudiar el ejemplo en VSCode

En el archivo **`model.py`** del ejemplo inspeccione el código para entender los TADs y estructuras de datos utilizados.

Lo primero que deben estudiar, es cómo está representado el analizador de rutas de buses. Revisen el código hasta entender cómo se construye y cómo se usa el grafo para solucionar el problema.

A continuación, responda las siguientes preguntas y registre su respuesta en el documento de observaciones:

e. ¿El grafo definido es denso o disperso?, ¿El grafo es dirigido o no dirigido?, ¿El grafo está fuertemente conectado?
f. ¿Cuál es el tamaño inicial del grafo?
g. ¿Cuál es la Estructura de datos utilizada?
h. ¿Cuál es la función de comparación utilizada?

PASO 6: Actualizar los repositorios

Para el repositorio del laboratorio confirme los cambios con los comandos **`Commit`** y **`Push`** en la rama **main** local y de GitHub con el comentario *"laboratorio 9 - Entrega final"* antes de la fecha límite de entrega.

PASO 7: Copiar la plantilla del Reto No. 4 en su organización

Copie/Haga Fork del repositorio del laboratorio en su organización con el procedimiento aprendido en las prácticas anteriores.

El repositorio del proyecto base que utiliza este laboratorio es el siguiente:

- https://github.com/ISIS1225DEVS/Reto4-Template.git

Antes de clonar el repositorio en su computador diríjase a su organización (Ej.: EDA2021-1-SEC02-G01 para el primer grupo de la sección 2 del curso) y cambie el nombre del repositorio de acuerdo con el esquema Reto4-G<<XX>> donde XX es el número del grupo de trabajo. (Ej.: Reto4-G01 para el grupo 1 de la sección 2).

Recuerde que NO necesita agregar la sección o el semestre en este nombre porque ya está identificado en su organización.

PASO 8: Descargue el Reto No. 4

Después de renombrar el proyecto dentro de su organización clone el código en su máquina local siguiendo lo aprendido y modifique el **README** principal con los nombres de los integrantes del grupo e identificar claramente cual miembro Implementará cual requerimiento individual, por ejemplo:

- *Req. 2 - Santiago Arteaga, 200411086, sa-artea@uniandes.edu.co*
- *Req. 3 - Carlos Lozano, 200211089, calozanog@uniandes.edu.co*

Paso 8: Crear el menú para el Reto

Tomando inspiración del código estudiado en el ejemplo, implemente en el **`view.py`** del reto el menú de opciones para la carga de archivos, creación de catálogo y los cinco requerimientos correspondientes.

PASO 9: Analizar Datos del Reto

Descargue los datos oficiales del reto de la sección unificada de la clase de la carpeta de contenido **RETOS/Reto 4/Datos** el grupo de archivos ZIP de la página contienen los CSV necesarios para el desarrollo del reto.

Por último, examine los datos provistos para el reto y para cada uno de los requerimientos responder las siguientes preguntas de análisis y observación:

a) ¿Cuántos grafos se necesitan definir para solucionar los requerimientos del reto? y ¿Por qué?
b) ¿Cuáles son las características específicas de cada uno de los grafos definidos? (vértices, arcos, denso o disperso, dirigido o no dirigido).
c) Además de los grafos, ¿Qué otras estructuras de datos adicionales se necesitan para resolver los requerimientos? Y ¿Por qué?

PASO 10: Actualizar el repositorio en la rama principal

`Confirme` los cambios con los comandos **`commit`** y **`push`** en la rama **`main`** local y de GitHub con el comentario "Primera entrega – Reto 4" antes de la fecha límite de entrega.

PASO 11: Revisar entregables de la practica

Finalmente, para realizar la entrega del laboratorio revise que sus entregables de la practica estén completos. Para ello, siga las siguientes indicaciones:

1) Acceso al profesor de laboratorio y los monitores de su sección a la organización del grupo.
2) **README** del repositorio con los datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).

3) Enlace al repositorio GitHub **LabGraph-S<<XX>>-G<<YY>>** con rama ***Main*** actualizada con el comentario *"Laboratorio 9 – Entrega final"* antes del límite de entrega.
4) Incluir en repositorio del laboratorio en la carpeta **Docs** el documento **observaciones-lab9.pdf** con las respuestas a las preguntas de observación.
   a) ¿Qué instrucción se usa para cambiar el límite de recursión de Python?
   b) ¿Por qué considera que se debe hacer este cambio?
   c) ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?
   d) ¿Qué relación creen que existe entre el número de vértices, arcos y el tiempo que toma la operación 4?
   e) ¿El grafo definido es denso o disperso?, ¿El grafo es dirigido o no dirigido?, ¿El grafo está fuertemente conectado?
   f) ¿Cuál es el tamaño inicial del grafo?
   g) ¿Cuál es la Estructura de datos utilizada?
   h) ¿Cuál es la función de comparación utilizada?

5) Enlace al repositorio GitHub Reto4-G<<XX>> con rama **`main`** actualizada con el comentario "Primera entrega – Reto 4" antes de la fecha límite de entrega.
6) Incluir en repositorio del reto en la carpeta **Docs** el documento **observaciones-reto4.pdf** con las respuestas a las preguntas de observación.
   a) ¿Cuántos grafos se necesitan definir para solucionar los requerimientos del reto? y ¿Por qué?
   b) ¿Cuáles son las características específicas de cada uno de los grafos definidos? (vértices, arcos, denso o disperso, dirigido o no dirigido).
   c) Además de los grafos, ¿Qué otras estructuras de datos adicionales se necesitan para resolver los requerimientos? Y ¿Por qué?

PASO 12: Compartir resultados con los evaluadores

Envíe el **enlace (URL)** del repositorio por **BrightSpace** antes de la fecha límite de entrega.

Recuerden que cualquier documento solicitado durante la práctica debe incluirse dentro del repositorio GIT y solo se calificarán los entregables hasta el último ***COMMIT*** realizado previo a la media noche **(11:59 PM) del 17 de noviembre de 2021.**