Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–2693 – Laboratorio de Algoritmos y Estructuras III.
Septiembre–Diciembre 2023

Proyecto I: Grafo
Implementación con listas de adyacencias
(20 pts)

Introducción

Sea `G = (V, E)` un grafo dirigido, una posible forma de representarlo es usando **listas de adyacencias**. En esta implementación se tiene una **lista de listas de vértices** de tamaño `|V|`. Cada una de estas listas representa a los sucesores de un vértice particular. La mayor ventaja de esta implementación es su flexibilidad. Permite agregar vértices y arcos de forma sencilla.

Para este proyecto, se desea que los estudiantes hagan una implementación en **Java** de la estructura de datos **Grafo Dirigido** usando **listas de adyacencias** como una clase genérica. Para ello, se proveerá el archivo `Graph.java` conteniendo la interfaz `Graph` que el estudiante deberá implementar en una clase concreta `AdjacencyListGraph` contenida en un archivo `AdjacencyListGraph.java`.

A continuación se detallan los métodos públicos que la clase `AdjacencyListGraph` debe implementar para este proyecto. En esta especificación `T` es la variable de tipo para los vértices del grafo.

add

```
boolean add(T vertex)
```

Recibe un vértice y lo agrega al grafo. Retorna `true` si el vértice es agregado con éxito. Retorna `false` en caso contrario.

connect

```
boolean connect(T from, T to)
```

Recibe dos vértices `from` y `to` y agrega al grafo un arco saliente de `from` y entrante a `to`. Retorna `true` si el arco es agregado con éxito. Retorna `false` en caso contrario.

disconnect

```
boolean disconnect(T from, T to)
```

Recibe dos vértices `from` y `to` y elimina del grafo el arco saliente de `from` y entrante a `to`. Retorna `true` si el arco es eliminado con éxito. Retorna `false` en caso contrario.

contains

```
boolean contains(T vertex)
```

Recibe un vértice y retorna `true` si el vértice pertenece a `V`. Retorna `false` en caso contrario.

getInwardEdges

```
List<T> getInwardEdges(T to)
```

Recibe un vértice `v` y retorna la lista de predecedores de `v`. Es decir, retorna la lista de todos los $u \in V$ tales que $(u, v) \in E$. Si ocurre algún error, retorna la referencia `null`.

getOutwardEdges

```
List<T> getOutwardEdges(T from)
```

Recibe un vértice `v` y retorna la lista de sucesores de `v`. Es decir, retorna la lista de todos los $u \in V$ tales que $(v, u) \in E$. Si ocurre algún error, retorna la referencia `null`.

getVerticesConnectedTo

```
List<T> getVerticesConnectedTo(T vertex)
```

Recibe un vértice `v` y retorna la lista de vértices adyacentes a `v`. Es decir, retorna la lista de todos los $u \in V$ tales que $(v, u) \in E$ o $(u, v) \in E$. Si ocurre algún error, retorna la referencia `null`.

getAllVertices

```
List<T> getAllVertices()
```

Retorna la lista de todos vértices del grafo. Es decir, todos los elementos de `V`.

remove

```
boolean remove(T vertex)
```

Recibe un vértice y lo elimina del grafo. Retorna `true` si el vértice es eliminado con éxito. Retorna `false` en caso contrario.

size

```
int size()
```

Retorna la cantidad de vértices que contiene el grafo. Es decir, la cardinalidad del conjunto de vértices `|V|`.

subgraph

```
Graph<T> subgraph(Collection<T> vertices)
```

Recibe una colección V' de vértices y retorna otra instancia de grafo donde el conjunto de vértices contiene solo aquellos vértices presentes en `V'` y solo aquellos arcos asociados a esos vértices. Es decir, retorna $G' = (V', E')$ donde $E' = \{(u, v) \in E | u \in V' \land v \in V'\}$.

Entrega

Este proyecto se realizará en equipos de 2 integrantes y debe ser subido a *GitHub*. Su repositorio debe contener la interfaz `Graph.java` su implementación en un archivo `AdjacencyListGraph.java` y un archivo `README.md` identificado con el nombre y número de carnet de los integrantes y una breve explicación de su implementación junto con una estimación de la complejidad computacional de cada método (Big O notation).

La fecha límite de entrega es el **miércoles 25 de octubre de 2023 a las 11:59 pm**.