Search ("ENTONADOR")

AyED 2024. Grafos- 29.06.2024 - Tema 1

Apellido	Nombre	Legajo	Corrigió
Ejercicio 1:	Ejercicio 2:	Ejercicio 3:	Total:

JERCICIO 1: Puntaje 5 puntos

mplemente la **clase Parcial**, y el método:

public List<???> invitacionMasterClass (Graph<???> red, String usuario, int distancia, int limite)

En una red social, cada usuario está conectado con otros mediante relaciones de amistad. Estas relaciones pueden ser representadas como un grafo no dirigido, donde cada nodo representa un usuario y cada arista representa una amistad entre dos usuarios.

Un **usuario** quiere invitar a una masterclass a todos sus contactos que se encuentran hasta cierta **distancia** de él, pero hay un **límite** en la cantidad total de personas que puede invitar. El nombre del usuario, la distancia y el límite son parámetros del método. Los nombres de usuarios no se repiten en el grafo.

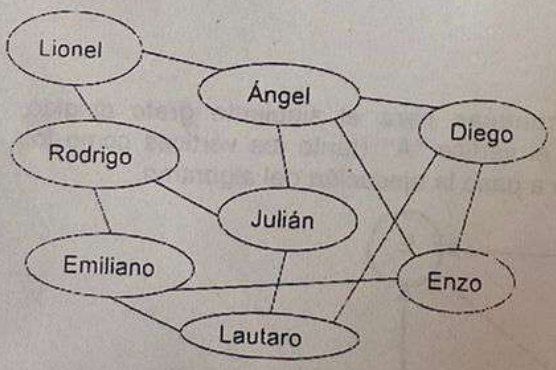
Encuentre **todos los usuarios** que cumplen con este criterio, **priorizando** a aquellos que están más cerca del usuario. Una vez que se llegue al límite de usuarios se debe cortar la ejecución.

La lista a retornar debe contener en cada elemento: **nombre de usuario y la distancia.**

Nota: Al recorrer la red social, asegúrate de visitar todos los amigos a una distancia menor antes de considerar los amigos a la distancia siguiente.

En esta red de ejemplo:

- Si usuario es Lionel, la distancia = 2 y el límite = 4 se podría devolver alguna de las siguientes listas:
  - (Ángel,1), (Rodrigo, 1), (Diego, 2), (Julián, 2)
  - (Rodrigo,1), (Ángel, 1), (Emiliano, 2), (Julián, 2)
  - (Ángel,1), (Rodrigo, 1), (Diego, 2), (Enzo, 2)
  - (Ángel,1), (Rodrigo, 1), (Julián, 2), (Enzo, 2)

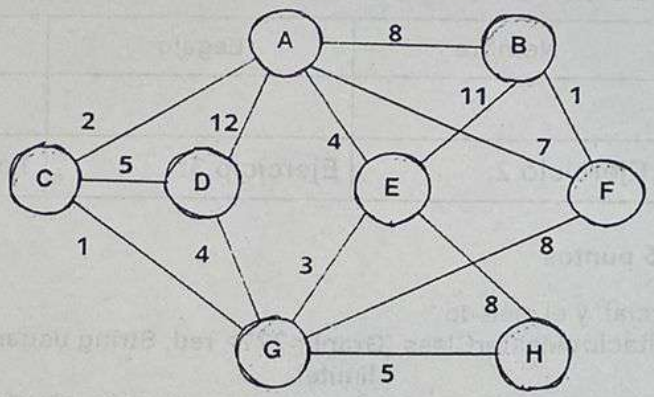


Tener en cuenta que:

- Se debe completar en la firma del método los tipos de datos indicados con signo de interrogación.
- Si no hay usuarios que cumplan con el criterio, devuelva la lista vacía.
- Si se cumple el límite o se pasa de la distancia se debe cortar la ejecución.
- No se puede pasar 2 veces por el mismo lugar.
- No se puede agregar variables de clase ni de instancia.
- El grafo se debe recorrer una única vez.

EJERCICIO 2: Puntaje 3 puntos

a) Muestre paso a paso la ejecución del algoritmo de **Prim** partiendo del vértice **"A"**. Muestre todos los pasos intermedios, indicando el orden en que se van procesando los vértices.



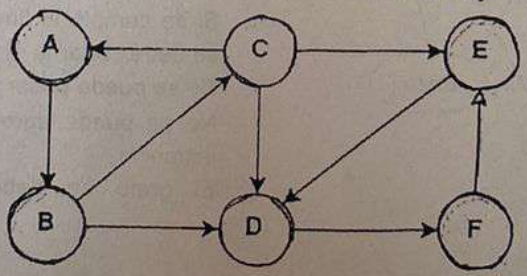
Orden que toma el Vértice v	Vértices	Costo de la arista	Vértice Previo	Visitado
1°	A	0		~~0~~ 1
	B	∞		0
	C	∞		0
	D	∞		0
	E	∞		0
	F	∞		0
	G	∞		0
	H	∞		0

b) Escriba la lista de vértices en el orden en que fueron agregados al árbol. Dibuje cómo queda el árbol resultante.

c) ¿Qué se obtiene con el algoritmo de Prim?

EJERCICIO 3: Puntaje 2 puntos

Indicar cuáles son las **componentes fuertemente conexas** para el siguiente grafo dirigido, utilizando el **algoritmo de Kosaraju** comenzando por el vértice **"A"** (tanto los vértices como los adyacentes se procesan alfabéticamente). Muestre paso a paso la ejecución del algoritmo.



115

AyED 2024. Grafos- 29.06.2024 - Tema 2

Apellido	Nombre	Legajo	Corrigió
Ejercicio 1:	**Ejercicio 2:**	**Ejercicio 3:**	**Total:**

EJERCICIO 1: Puntaje 5 puntos

Implemente la **clase Parcial**, y el método:

public ??? nivelPopularidad (Graph<???> red, String usuario, int distancia, int umbral)

En una red social, cada usuario está conectado con otros mediante relaciones de amistad. Estas relaciones pueden ser representadas como un grafo no dirigido, donde cada nodo representa un usuario y cada arista representa una amistad entre dos usuarios.

Queremos determinar el nivel de popularidad de un **usuario** basado en la cantidad de personas que están a **exactamente a una distancia** de él. El usuario y la distancia son parámetros del método. Los nombres de usuarios no se repiten en el grafo.

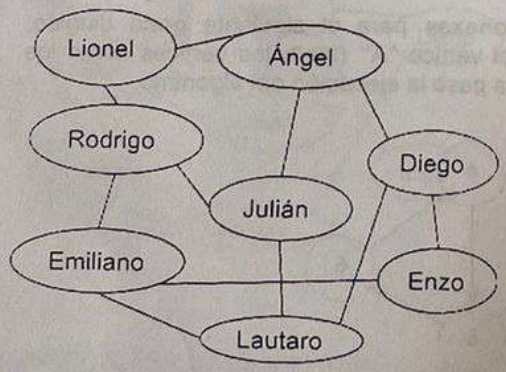
El método debe devolver una tupla que contenga:

- Un entero que representa **la cantidad de usuarios** que están a **exactamente una distancia** del usuario recibido por parámetro.
- Un **booleano** que indica si el usuario **es popular o no**. Se considera popular si la cantidad de usuarios que se encuentran a exactamente una distancia es mayor o igual a un **umbral** predefinido que se recibe por parámetro.

Nota: Al recorrer la red social, asegúrate de visitar todos los amigos a una distancia menor antes de considerar los amigos a la distancia siguiente.

En esta red de ejemplo:

- Si el usuario es Lionel, la distancia = 2 y el umbral = 3, debe devolver **3 y true**, ya que Lionel tiene 3 usuarios a exactamente distancia 2 (Emiliano, Julián y Diego) y 3 = umbral.
- Si el usuario es Lionel, la distancia = 1 y el umbral = 3, debe devolver **2 y false**, ya que Lioinel tiene 2 usuarios a exactamente distancia 1 (Ángel y Rodrigo), y 2 < umbral.
- Si el usuario fuese Juan, debe devolver **null**, ya que Juan no se encuentra en el grafo.

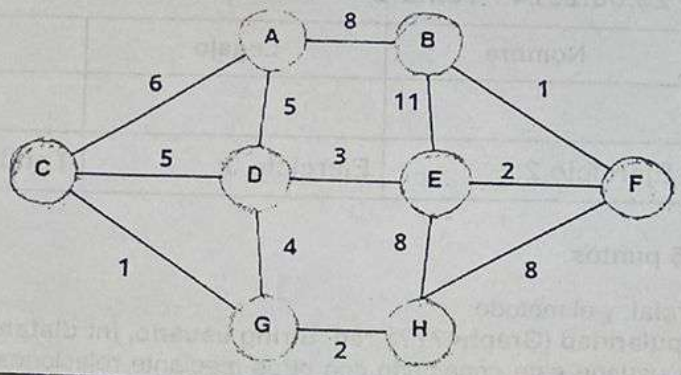


Tener en cuenta:

- Se debe completar en la firma del método los tipos de datos indicados con signo de interrogación.
- El usuario puede no existir. En este caso debe devolver null.
- En caso de que no haya usuarios en la distancia recibida, debe devolver 0 y false.
- No se puede pasar 2 veces por el mismo lugar.
- No se puede agregar variables de clase ni de instancia.
- El grafo se debe recorrer una única vez.

EJERCICIO 2: Puntaje 3 puntos

a) Muestre paso a paso la ejecución del algoritmo de **Prim** partiendo del vértice **"A"**. Muestre todos los pasos intermedios, indicando el orden en que se van procesando los vértices.



Orden que toma el Vértice v	Vértices	Costo de la arista	Vértice Previo	Visitado
1°	A	0		0 1
	B	∞		0
	C	∞		0
	D	∞		0
	E	∞		0
	F	∞		0
	G	∞		0
	H	∞		0

b) Escriba la lista de vértices en el orden en que fueron agregados al árbol. Dibuje cómo queda el árbol resultante.

c) ¿Qué se obtiene con el algoritmo de Prim?

EJERCICIO 3: Puntaje 2 puntos

Indicar cuáles son las **componentes fuertemente conexas** para el siguiente grafo dirigido, utilizando el **algoritmo de Kosaraju** comenzando por el vértice **"A"** (tanto los vértices como los adyacentes se procesan alfabéticamente). Muestre paso a paso la ejecución del algoritmo.