Practica 6 - Programación Orientada a Objetos

Profesora: Karla Ramírez Pulido Ayudante: Héctor Enrique Gómez Morales

Fecha de inicio: 4 de noviembre de 2015
Fecha de entrega: 25 de noviembre de 2015

1. Instrucciones

Una gráfica $G = (V, E)$ es un par ordenado compuesto por un conjunto V de vértices y un conjunto E de aristas. En esta practica la gráfica se puede representar en tres tipos de formatos.

Formato CSV: En el primer renglón se indica si la gráfica es dirigida (direct=1) o no (direct=0). En cada renglón subsecuente se define una arista de la gráfica, las dos primeras columnas indican el vértice origen y el vértice destino de la arista. La tercera columna indica el peso de la arista.

```
direct=0
"a", "b", 11
"a", "e", 1
"a", "f", 4
"b", "c", 8
"b", "g", 8
"c", "d", 3
"c", "h", 3
"d", "e", 3
"d", "i", 2
"e", "j", 1
"f", "h", 9
"f", "i", 7
"g", "i", 4
"g", "j", 1
"h", "j", 9
```

Formato JSON: Es el formato mas popular para el intercambio de información en Web. se tienen tres llaves: `direct, vertices y edges`. La primera `direct` indica con un entero si la gráfica es dirigida (direct: 1) o si es no dirigida (direct: 0). En la llave `vertices` se tiene un arreglo con los todos los vertices de la gráfica, finalmente en la llave `edges` se tiene un arreglo con las aristas de la gráfica.

```
{
  "direct": 0
  "vertices": ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j"]
  "edges": [
    ["a", "b", 11],
    ["a", "e", 1],
    ["a", "f", 4],
    ["b", "c", 8],
    ["b", "g", 8],
    ["c", "d", 3],
```

```
    ["c", "h", 3],
    ["d", "e", 3],
    ["d", "i", 2],
    ["e", "j", 1],
    ["f", "h", 9],
    ["f", "i", 7],
    ["g", "i", 4],
    ["g", "j", 1],
    ["h", "j", 9]
  ]
}
```

Formato XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graph PUBLIC "-//FC//DTD matrix//EN" "./graph.dtd">
<graph direct="0">
  <vertex label="a"/>
  <vertex label="b"/>
  <vertex label="c"/>
  <vertex label="d"/>
  <vertex label="e"/>
  <vertex label="f"/>
  <vertex label="g"/>
  <vertex label="h"/>
  <vertex label="i"/>
  <vertex label="j"/>
  <edge source="a" target="b" weight="11"/>
  <edge source="a" target="e" weight="1"/>
  <edge source="a" target="f" weight="4"/>
  <edge source="b" target="c" weight="8"/>
  <edge source="b" target="g" weight="8"/>
  <edge source="c" target="d" weight="3"/>
  <edge source="c" target="h" weight="3"/>
  <edge source="d" target="e" weight="3"/>
  <edge source="d" target="i" weight="2"/>
  <edge source="e" target="j" weight="1"/>
  <edge source="f" target="h" weight="9"/>
  <edge source="f" target="i" weight="7"/>
  <edge source="g" target="i" weight="4"/>
  <edge source="g" target="j" weight="1"/>
  <edge source="h" target="j" weight="9"/>
</graph>
```

En esta practica se trabajara en la implementación de gráficas (dirigidas y no dirigidas), haciendo uso de un lenguaje orientado a objetos haciendo uso de herencia y polimorfismo.

Esta práctica debe ser implementada haciendo uso de **Javascript** o **Python**.

Se debe incluir un archivo `README` que indique las instrucciones para correr su programa.

2. Ejercicios

1. (**3pts**) **Graph** Implementar una clase que represente una gráfica debe tener por lo menos los siguientes métodos:

- **directed**, regresa u booleano, `true` si la gráfica es dirigida y `false` en caso contrario.
- **vertices**, regresa un arreglo con todos los vértices de la gráfica.
- **edges**, regresa todas las aristas de la gráfica.

2. (**1.5pts**) **Vertex** Clase que representa un vértice de la gráfica, debe tener por lo menos los siguiente métodos:
   - **neighbours**, regresa los vertices adyacentes del vértice dado.
   - **degree**, regresa el grado del vértice.
3. (**1.5pts**) **Edges** Clase que representa una arista de la gráfica, debe tener por lo menos los siguientes métodos:
   - **svertex**, regresa el vértice origen de la arista.
   - **tvertex**, regresa el vértice destino de la arista.
   - **weight**, regresa el peso de la arista.
4. (**3pts**) **GraphReader** Deben implementar una clase que tome una ruta a un archivo (ya sea `XML, JSON, CSV`) y que regrese un objeto de la clase `Graph`
5. (**1pts**) **has_cycles** Se debe de implementar un método en `Graph` que regrese `true` si la gráfica tiene un ciclo o `false` en caso contrario.

   Finalmente al correr su programa se debe cargar cada uno de los formatos para la gráfica de petersen y la gráfica `graph` que se encuentran en el directorio `ejemplos` de la practica y imprimir lo siguiente:
   - El nombre del archivo de la gráfica, ie. petersen.json o graph.xml
   - Los vértices de la gráfica
   - Las aristas de la gráfica con sus pesos
   - Indicar si la gráfica tiene ciclos.

   IMPORTANTE: Se restaran 2 puntos de no presentarse herencia y/o polimorfismo en alguno de los puntos anteriormente mencionados.