jbenavidesv87 / **FlujoRedes**

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Insights

Repositorio para clase de Flujo en redes. PISIS, FIME, UANL. Primer semestre 2018. https://elisa.dyndns-web.com/teaching…

network-flow graph-theory python3 gnuplot

36 commits **2** branches **0** releases **1** contributor

Branch: **master** New pull request Find file Clone or download

jbenavidesv87 Reporte del ejemplo 4. Latest commit `0c7fa26` Mar 5, 2018

ejemplos	Reporte del ejemplo 4.	Mar 5, 2018
fuente	Imágenes en eps	Mar 5, 2018
p1	Reporte actualizado con enlace a gif	Feb 19, 2018
p2	-README actualizado	Feb 27, 2018
README.md	Reporte del ejemplo 4.	Mar 5, 2018

README.md

Grafos con `python` para Flujo en Redes

Este repositorio pretende ofrecer un programa que permita generar y trazar grafos enfocados a representar problemas de Flujo en redes, aunque podría utilizarse para otros fines.

Requisitos

Es necesario tener instalados:

- `python3`
- `gnuplot`

También se requiere incluir las librerías `Grafo` y `Nodo`.

```
from Grafo import Grafo
from Nodo import Nodo
```

Nota: Los usuarios de `Windows` deben agregar los directorios de instalación de ambos programas al `PATH` de `Windows` para poder correr los ejemplos desde el `Símbolo del sistema`. Instrucciones para `python3` y `gnuplot`.

Documentación

La carpeta `ejemplos` contiene los códigos que aquí se explican.

Notación

Se usará *G*, *n* u otro tipo de variables en este formato para referirse a un grafo*G* almacenado en la variable `G` como al grafo mismo; o tanto a un nodo *n* almacenado en la variable `n` como al nodo mismo, etcétera; el contexto será lo suficientemente claro para evitar confusiones.

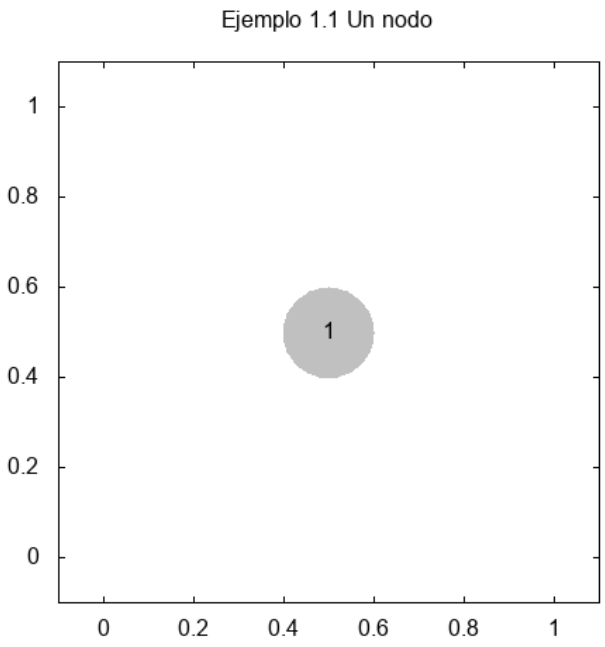
Estructura mínima

Para definir un grafo con un nodo y obtener una imagen `PNG` del mismo, basta con escribir el siguiente código:

```
from Grafo import Grafo
from Nodo import Nodo

G = Grafo()
n1 = Nodo()
G.AgregarNodo(n1)

G.DibujarGrafo("Ejemplo 1.1 Un nodo")
```



Grafos

Declarar un grafo y almacenarlo en `G` se logra por la instrucción:

```
G = Grafo()
```

Un grafo *G* declarado de esta forma, posee las siguientes propiedades:

Propiedad	Variable	Valor por defecto
Nombre	`nombre`	`"grafo"`
Dirigido	`dirigido`	`False`
Nodos	`nodos`	`[]`
Pesos	`pesos`	`dict()`
Vecinos	`vecinos`	`dict()`

El **nombre** del grafo indica el nombre de los archivos de imagen o instrucciones de `gnuplot` que se generarán de este grafo. Así, un grafo generado de manera predeterminada produciría archivos de imagen `grafo.png`.

Cambiar el nombre de un grafo *G* por "arco" se realiza del modo siguiente:

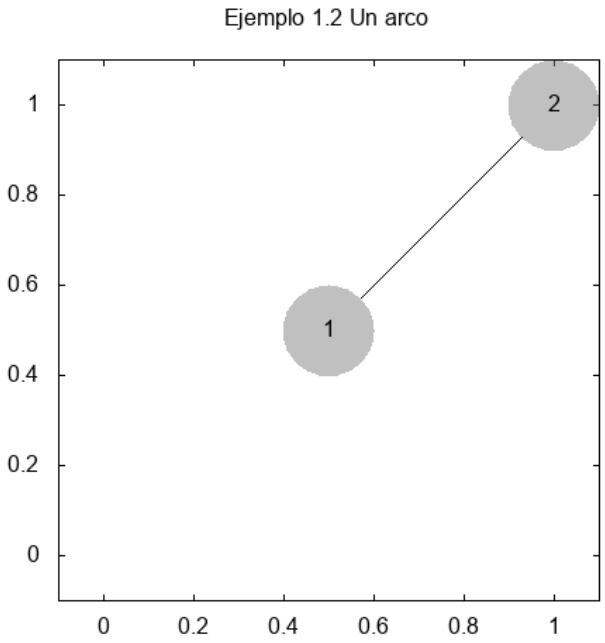
```
G.nombre = "arco"
```

Para definir si un grafo *G* es **dirigido**, basta con especificarlo mediante una booleana:

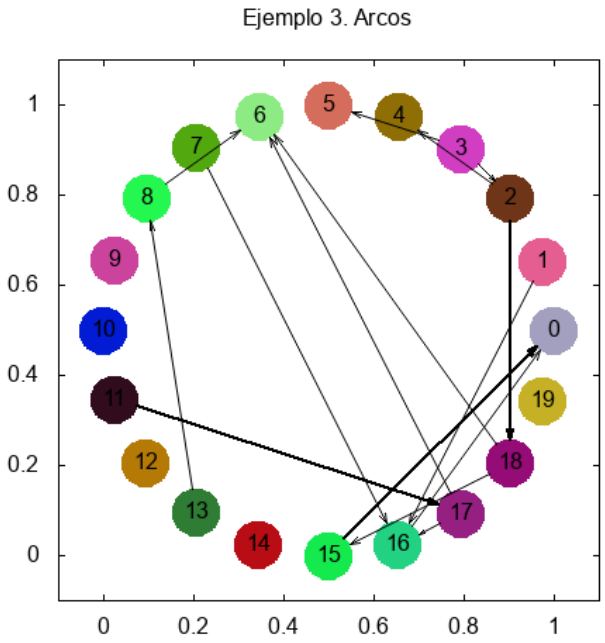
```
G.dirigido = True # G es dirigido

G.dirigido = False # G no es dirigido
```

Un grafo no dirigido, mantiene a los nodos vecinos conectados uno con el otro y representa esto mediante arcos sin flecha de dirección.



Un grafo dirigido considera que cada nodo podría dirigirse a vecinos que no necesariamente se dirijan a él, lo que se representa mediante arcos con una flecha que indica esta dirección.



Los nodos, vecinos y pesos se describirán en las secciones siguientes.

Nodos

De manera análoga a la creación de un grafo, para crear un nodo y almacenarlo en `n` se escribe:

```
n = Nodo()
```

Por defecto, un nodo tiene las siguientes propiedades:

Propiedad	Variable	Valor por defecto
Identificador	`id`	`"1"`
Tipo	`tipo`	`""`
Posición	`posicion`	`(0.5, 0.5)`
Radio	`radio`	`0.1`
Color (hexadecimal)	`color`	`"#0080808080"` (Gris)

El **identificador** se utiliza para mostrar una etiqueta en color negro dentro del nodo al dibujarlo. A un nodo *n* se le puede poner otro identificador con:

```
n.id = "2" # Dibujaría un nodo con un 2 en el centro

n.id = "" # Dibujaría nodos sin etiqueta
```

El **tipo** es una variable que podría ser utilizada para definir categorías de nodos [falta agregar ejemplo de asignación de horarios].

La **posición** establece las coordenadas del centro del nodo para ser representado en un plano cartesiano bidimensional. Colocar un nodo *n* en (0, 1) se haría así:

```
n.posicion = (0, 1) # 0 para el eje horizontal (x) y 1 para el vertical (y)
```

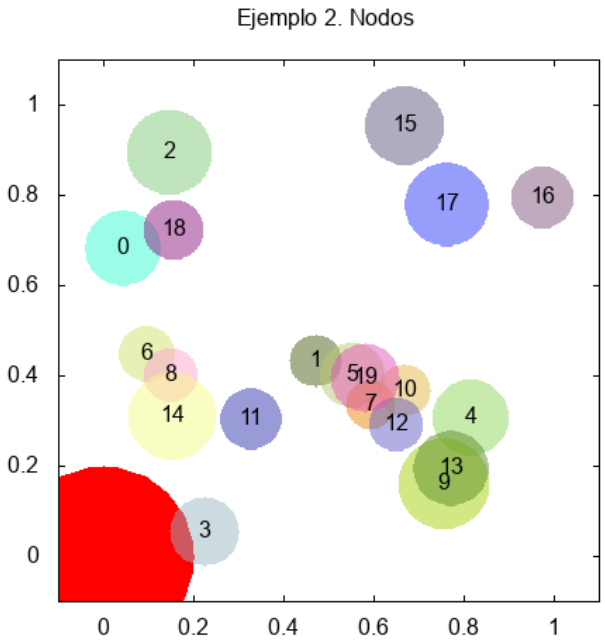
El **radio** controla el tamaño del nodo con base en las unidades del eje de coordenadas, de modo que, por ejemplo, 0.5 de radio corresponderían a un nodo circular con un diámetro de una unidad respecto del eje de coordenadas. El valor del radio puede ser entero o decimal, de modo que podría asignársele valores a partir de operaciones que den por resultado este tipo de valores. Un radio de 0.3 periódico de un nodo *n* se asigna mediante:

```
n.radio = 1 / 3
```

El **color** de un nodo *n* se establece con la función `Color`, que recibe como parámetros el componente rojo, verde, azul y, opcionalmente, alfa (transparencia) en rangos de números enteros de 0 a 255, para generar un color RGBA. Para asignar cambios de color a un nodo se escribe:

```
n.Color(255, 255, 0) # Nodo a color amarillo

n.Color(255, 0, 255, 128) # Nodo a color morado, transparencia media
```



Agregar nodo

Para agregar un nodo *n* ya definido a un grafo *G* existente, se usa la función `AgregarNodo` del grafo:

```
G.AgregarNodo(n)
```

Obtener un nodo por su identificador

Es posible recuperar un nodo por su identificador con el método `NodoConId` de un grafo. Obtener del grafo *G* el nodo con identificador *i* y almacenarlo en `n` se realiza como sigue:

```
n = G.NodoConId(i)
```

Observaciones:

- El método devuelve el primer nodo que tenga el identificador *i* en orden de agregación.
- En caso de que no haya nodos con ese identificador, el método devuelve `None`.

Eliminar un nodo

Es posible eliminar un nodo *n* con sus vecindades y pesos asociados de un grafo *G* con:

```
G.EliminarNodo(n)
```

Arcos

Para establecer una vecindad en un grafo *G* entre un nodo *n* y otro *v* con peso 1, se utiliza la función `ConectarNodos` del grafo:

```
G.ConectarNodos(n, v)
```

Esta función realiza las siguientes acciones:

1. Agrega ambos nodos al grafo, en caso de no estarlo.
2. Establece una vecindad que va de *n* a *v*
3. Establece un peso de 1 para dicha vecindad

Además, si se trata de un grafo no dirigido, realiza los pasos 2 y 3 de *v* a *n*.

El **peso** inicial de una vecindad se puede modificar pasando como tercer parámetro de la función `ConectarNodos` el peso que se desee. Una vecindad con peso 0.5 entre *n* y *v* quedaría:

```
G.ConectarNodos(n, v, 0.5)
```

Este peso corresponderá al grosor del arco que conecte a los nodos en la representación del grafo.

Modificar pesos de las vecindades de un nodo

Se puede asignar a las vecindades de un nodo *n* en un grafo *G*, un peso *p* incluyendo la instrucción:

```
G.ModificarPesos(n, p)
```

Observaciones:

- Si *G* es un grafo dirigido, se actualizan los pesos de las vecindades creadas a partir del nodo *n* **pero** no las que se dirijan al nodo *n*, en caso de haberlas.
- Si *G* es un grafo no dirigido, se modifican **todos** los pesos de las vecindades del nodo *n*.

Eliminar vecindades de un nodo

Para eliminar todas las vecindades de un nodo *n* en un grafo *G*, basta con usar la función:

```
G.EliminarVecindades(n)
```

Observaciones:

- Si *G* es un grafo dirigido, se eliminan las vecindades creadas a partir del nodo *n* **pero** no las se dirijan al nodo *n*, en caso haberlas.
- Si *G* es un grafo no dirigido, se eliminan **todas** las vecindades del nodo *n*.
- Eliminar vecindades también elimina los pesos asignados a dichas vecindades.

Dibujar un grafo

Las siguientes líneas dibujan, en imágenes `PNG`, un grafo *G* sin título y con título "Título":

```
G.DibujarGrafo() # Imagen de un grafo sin título

G.DibujarGrafo("Título") # Imagen de un grafo con encabezado "Título"
```

Adicionalmente, se puede generar una imagen `EPS` del grafo agregándo un parámetro adicional a la función:

```
G.DibujarGrafo("Título", True)
```

Tareas pendientes

- [x] ~~Arcos simples~~
- [x] ~~Configuración de nodos~~
- [x] ~~Configuración de grafos~~

- [x] ~~Completar documentación de arcos~~
- [x] ~~Eliminar vecindades~~
- [x] ~~Modificar pesos de vecindades~~
- [x] ~~Agregar imágenes en formato `EPS`~~
- [ ] Guardar grafos en archivos de texto
- [ ] Agregar ejemplo de asignación de horarios
- [ ] Nodos sólo con contorno y grosor de contorno
- [ ] Agregar rangos de los ejes del plano a dibujar
- [ ] Ejemplos de aplicación con algoritmo genético para tipos de nodo
- [ ] Ejemplo de aplicación de flujo en redes ()
- [ ] Agregar conexiones de un mismo nodo consigo mismo
- [ ] Hacer grafos en forma de árbol (los hijos de un nodo son sus ramas, etc.)