



Features

Business

Explore

Marketplace

Pricing

This repository Search

Sign in or Sign up

jbenavidesv87 / FlujoRedes

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Insights

Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

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
fuentes	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.

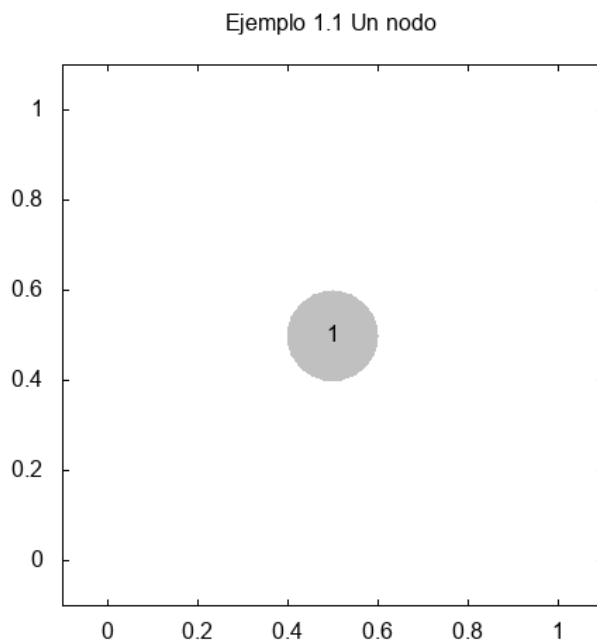
Estructura mínima

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`.

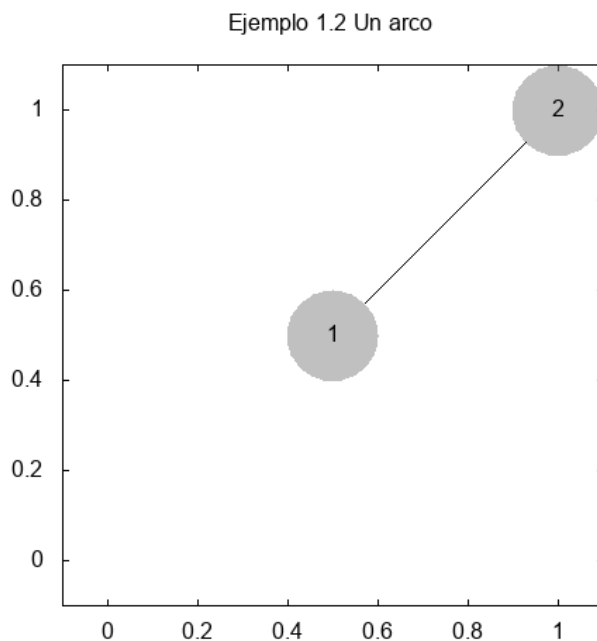
Cambiar el nombre de un grafo `G` por "**arco**" se realiza del modo siguiente:

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

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

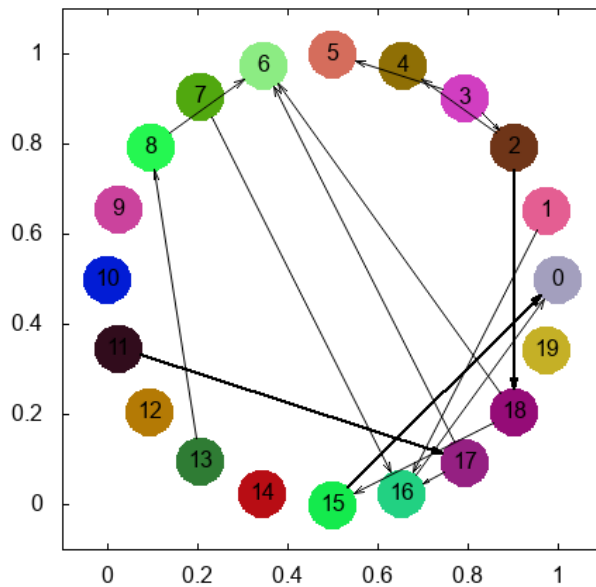
```
G.dirigido = True # G es dirigido
G.dirigido = False # G no es dirigido
```

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.

Ejemplo 3. Arcos



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	<code>id</code>	"1"
Tipo	<code>tipo</code>	""
Posición	<code>posicion</code>	(0.5, 0.5)
Radio	<code>radio</code>	0.1
Color (hexadecimal)	<code>color</code>	"#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:

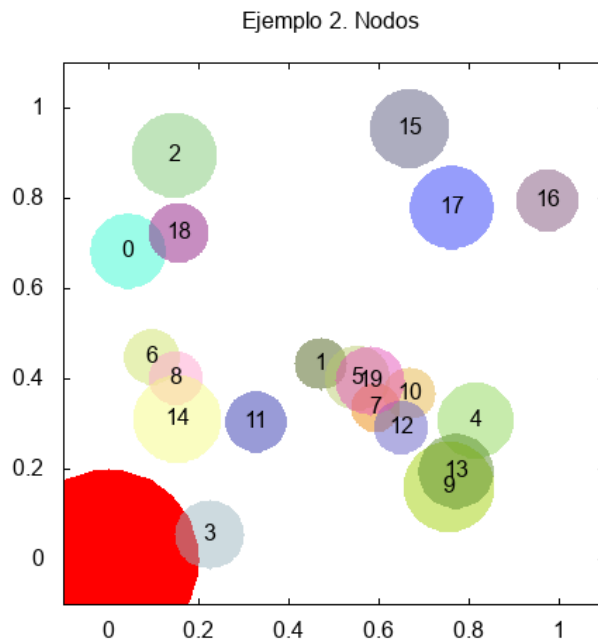
```
n.posicion = (0, 1) # 0 para el eje horizontal (x) y 1 para el vertical (y)
```

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 dev 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 que se dirijan al nodo n , en caso de 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 agregando un parámetro adicional a la función:

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

Tareas pendientes

- ☒ Arcos simples
- ☒ Configuración de nodos
- ☒ Configuración de grafos

- ☒ Completar documentación de arcos
- ☒ Eliminar vecindades
- ☒ Modificar pesos de vecindades
- ☒ 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.)

© 2018 GitHub, Inc.

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)

[Help](#)



[Contact GitHub](#)

[API](#)

[Training](#)

[Shop](#)

[Blog](#)

[About](#)



Features

Business

Explore

Marketplace

Pricing

This repository

Search

Sign in or Sign up

jbenavidesv87 / FlujoRedes

Watch

1

Star

0

Fork

0

Code

Issues 0

Pull requests 0

Projects 0

Insights

Branch: master

FlujoRedes / ejemplos / 04ModificarNodos /

Create new file

Find file

History

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

..		
dirigido.gnu	Imágenes en eps	Mar 5, 2018
dirigido.png	-Obtener grafos por identificador	Mar 4, 2018
main.py	Reporte del ejemplo 4.	Mar 5, 2018
noDirigido.eps	Reporte del ejemplo 4.	Mar 5, 2018
noDirigido.gnu	Reporte del ejemplo 4.	Mar 5, 2018
noDirigido.png	Reporte del ejemplo 4.	Mar 5, 2018
readme.md	Reporte del ejemplo 4.	Mar 5, 2018

readme.md

Ejemplo 4. Manipular nodos

En este ejemplo se un grafo G , dirigido y que lleva por nombre **dirigido** al que se le agregan veinte nodos a los que se les identifica por números enteros consecutivos, del 0 al 19, se les da un radio de 0.08 y una posición que los conforma en una red. Dicha posición está calculada a partir de los siguientes datos:

- $N = 20$ nodos
- $L = \lceil \sqrt{20} \rceil$: Nodos por fila y columna

La ecuación del par ordenado para la posición p_i de cada nodo n_i con $i \in \{0, 1, 2, \dots, N - 1\}$ es:

$$p_i = (i \bmod L) / L, \lfloor i / L \rfloor / L$$

Cada uno de estos nodos se conecta con sus vecinos horizontales y verticales inmediatos.

Después, se elige el nodo con identificador **3**, se cambia su color a rojo y su posición a (1, 0) con el código:

```
n = G.NodoConId(3)
n.Color(255, 0, 0)
n.posicion= (1, 0)
```

A continuación, se toma el nodo que tiene identificador **5**, se le asigna el color azul y se eliminan sus vecindades:


```
n = G.NodoConId(5)
n.Color(0, 255, 0)
G.EliminarVecindades(n)
```

Después, se elimina el nodo con identificador **14**:

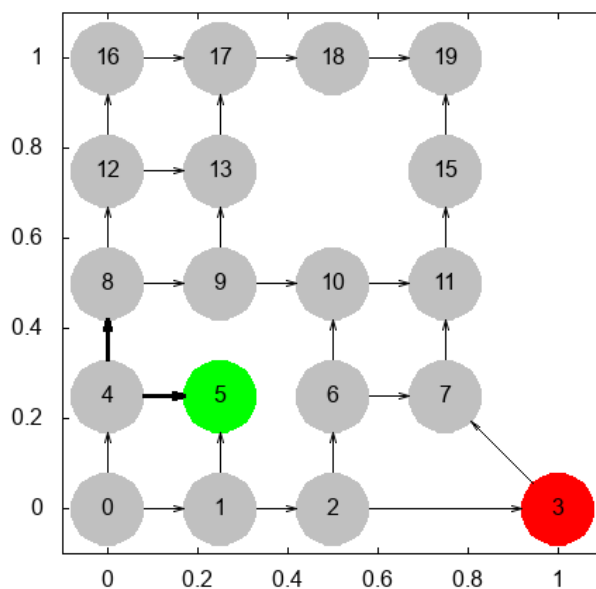
```
n = G.NodoConId(14)
G.EliminarNodo(n)
```

Enseguida, se ponen los pesos de las vecindades del nodo **4** a un valor de 3:

```
n = G.NodoConId(4)
G.ModificarPesos(n, 3)
```

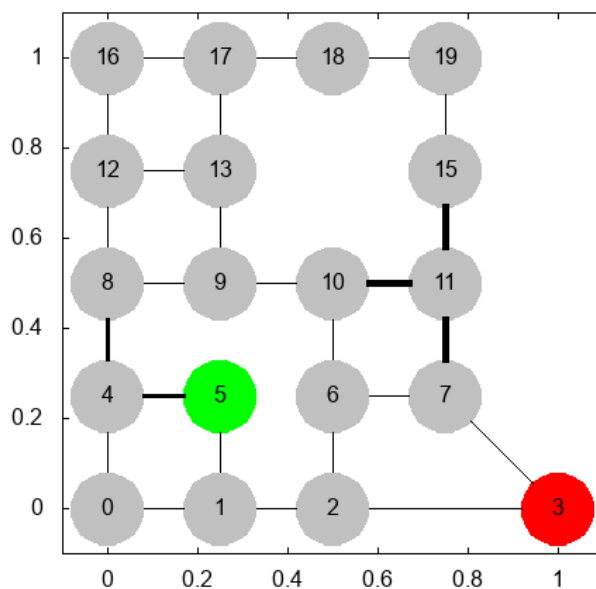
Ahora, se dibuja el grafo dirigido resultante, mostrado en la imagen siguiente:

Ejemplo 4.1 Manipulación de nodos en grafo dirigido




Para finalizar, se cambia el grafo a no dirigido y su nombre se actualiza **anoDirigido** y se modifican los pesos del nodo con identificador **11**. Este grafo resultante se muestra en una imagen en formato **EPS** :

Ejemplo 4.2 Manipulación de nodos en grafo no dirigido



© 2018 GitHub, Inc.

- [Terms](#)
- [Privacy](#)
- [Security](#)
- [Status](#)
- [Help](#)
- 
- [Contact GitHub](#)
- [API](#)
- [Training](#)
- [Shop](#)
- [Blog](#)
- [About](#)



Features

Business

Explore

Marketplace

Pricing

This repository

Search

Sign in or Sign up

jbenavidesv87 / FlujoRedes

Watch

1

Star

0

Fork

0

Code

Issues 0

Pull requests 0

Projects 0

Insights

Branch: master

FlujoRedes / ejemplos / 03Arcos /

Create new file

Find file

History



jbenavidesv87 Clarificada la explicación del ciclo en el reporte del ejemplo 3.

Latest commit da99f0f Mar 1, 2018

..

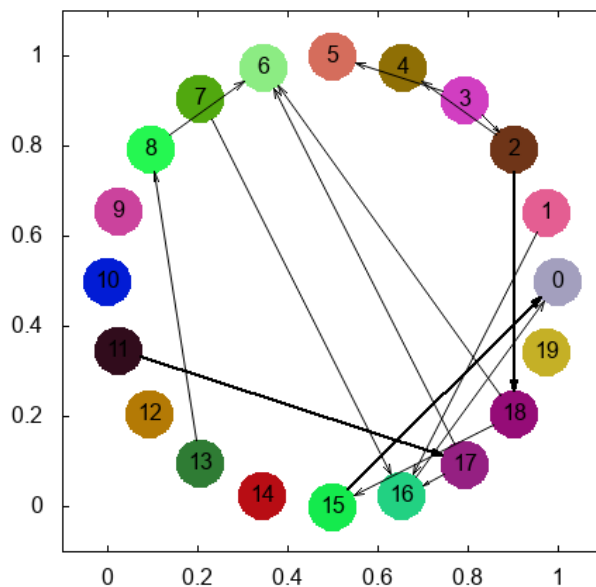
grafo.gnu	Reporte del Ejemplo 3, terminado.	Mar 1, 2018
grafo.png	Reporte del Ejemplo 3, terminado.	Mar 1, 2018
main.py	Reporte del Ejemplo 3, terminado.	Mar 1, 2018
readme.md	Clarificada la explicación del ciclo en el reporte del ejemplo 3.	Mar 1, 2018

[readme.md](#)

Ejemplo 3. Arcos

En este programa se crea un grafo dirigido con veinte nodos coloreados al azar dispuestos en forma circular en torno a un centro en (0.5, 0.5), identificados cada uno por números enteros del 0 al 19, ambos incluidos. Cada uno de estos veinte nodos se conecta con una probabilidad de 0.3 con otros tres nodos elegidos al azar. A cada vecindad establecida de esta manera, se le asigna un peso al azar en el rango [0, 2] que se refleja en el grosor de la línea que representa cada arco. Uno de los grafos resultantes es el siguiente:

Ejemplo 3. Arcos



El código para hacerlo requiere el uso de las funciones `random`, `randint` y `sample` de la librería `random`, y de las funciones `sin` y `cos` de la librería `math`.

```
from random import random, randint, sample
from math import cos, sin
```

La función `sample(a, b)` regresa una lista de b elementos elegidos al azar de la población a . Las funciones `sin(c)` y `cos(c)` devuelven el seno y coseno de c en radianes.

Se establecen:

- $N = 20$ nodos
- Un radio $r = 0.5$ para la circunferencia que formarán los nodos entre sí
- Un perímetro $P = 2 \times 3.14 \times r$ de la circunferencia que formarán los nodos entre sí
- Un radio $r_n = P / N / 3$ para cada nodo. En este caso, se divide P / N para conocer la medida del arco correspondiente a cada nodo y se divide ese valor entre 3 para que las circunferencias de los nodos no coincidan con los centros de los nodos adyacentes. (Recomiendo eliminar esa división entre tres para ver el cambio mencionado)
- Una fracción angular $\theta = 2 \times 3.14$ rad que ocupará cada nodo respecto a la circunferencia que se formará
- Una probabilidad $p_v = 0.3$ de establecerse vecindades

Estos parámetros corresponden respectivamente con el siguiente código en `python`:

```
N = 20
r = 0.5
P = 2 * 3.14 * r
rNodo = P / N / 3
theta = 2 * 3.14 / N
pVecino = 0.3
```

Ahora se instancia un grafo G y se especifica que sea dirigido:

```
G = Grafo()
G.dirigido = True
```

Inmediatamente después, se hace un ciclo de $i \in [0, 1, 2, \dots, N - 1]$, donde se crean N nodos, asignándosele a cada nodo n_i un identificador i , un radio igual a r_n , un color RGB al azar y una posición (x_i, y_i) a lo largo de la circunferencia que formarán que dependerá de las funciones

$$x_i = 0.5 + r \times \cos(\theta i)$$

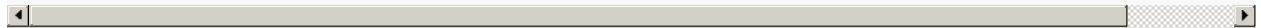
$$y_i = 0.5 + r \times \sin(\theta i)$$

Y se agregará cada nodo n_i al grafo G , todo lo cual corresponde al código:

```
for i in range(N):
    n = Nodo()
    n.id = i
    n.radio = rNodo
    n.Color(
        randint(0, 255), # rojo (R)
        randint(0, 255), # verde (G)
        randint(0, 255), # azul (B)
    )
    n.posicion = (
        0.5 + r * cos(theta * i), 0.5 + r * sin(theta * i)
    )
    G.AgregarNodo(n)
```

Finalmente, por cada nodo n_i dado $i \in [0, 1, \dots, N - 1]$ y con una probabilidad dep_v se intenta establecer una vecindad con algún nodo al azar entre el resto de los nodos del grafo G . Esta probabilidad se realiza tres veces por cada nodo, eligiendo en cada ocasión un nodo al azar entre los restantes. Luego, por cada intento exitoso de establecer una vecindad, se elige un vecino v al azar y se conecta con el nodo n_i correspondiente con un peso elegido al azar en un rango $[0; 2]$. Lo anterior se realiza en el siguiente ciclo:

```
for n in G.nodos:
    for i in range(3):
        if random() < pVecino:
            candidatos = set(G.nodos) - set([n]) # Se quita el nodo n de todo los nodos
            v = sample(candidatos, 1) # Se elige un nodo al azar
            G.ConectarNodos(n, v[0], random() * 2) # Se establece la vecindad dirigida de n a v con peso
```



Por último, se dibuja el nodo con título **Ejemplo 3. Arcos**:

```
G.DibujarGrafo("Ejemplo 3. Arcos")
```





Features

Business

Explore

Marketplace

Pricing

This repository Search

Sign in or Sign up

jbenavidesv87 / FlujoRedes

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Insights

Branch: master FlujoRedes / ejemplos / 02Nodos /

Create new file Find file History

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

..		
grafo.eps	Imágenes en eps	Mar 5, 2018
grafo.gnu	Imágenes en eps	Mar 5, 2018
grafo.png	-Controlada la eventualidad de que un usuario nombrara a un grafo con...	Mar 1, 2018
main.py	Reporte del ejemplo 4.	Mar 5, 2018
readme.md	Reporte del Ejemplo 3, terminado.	Mar 1, 2018

readme.md

Ejemplo 2. Nodos

En este ejemplo se crearán nodos y se modificarán sus propiedades iniciales manualmente y luego de manera aleatoria mediante el uso de las funciones `random` y `randint` de la librería `random` de `python`. Al usar `random()` se genera un número decimal aleatorio entre 0 y 1, ambos incluidos; y `randint(a, b)` genera un número entero entre `a` y `b`, ambos incluidos.

```
from Grafo import Grafo
from Nodo import Nodo
from random import random, randint # librerías de random
```

Primero se define un número total de nodos a generar, N , y se define un grafo G que los contendrá.

```
N = 20 # Nodos totales

G = Grafo()
```

Enseguida, se creará un nodo n al que se le modificarán sus propiedades manualmente. Se borrará su identificador para que aparezca sin etiqueta, se colocará en el origen del eje de coordenadas, su radio que se igualará a 0.2 y se hará rojo.

```

n = Nodo() # Crea un nodo almacenado en n
n.id = "" # Esto permite eliminar la etiqueta del nodo
n.posicion = (0, 0) # Centro del nodo en la coordenada (0, 0)
n.radio = 0.2 # Radio del nodo en 0.2 puntos
n.Color( # Color con valores de 0 (mínimo) a 255 (máximo):
    255, # rojo
    0, # verde
    0, # azul
    0 # alfa (transparencia mínima, es decir, color sólido)
)
G.AgregarNodo(n) # Agrega el nodo al grafo

```

Posteriormente, se crearán N nodos con la ayuda de la función `range(N)`, la cual devuelve una lista de enteros de 0 hasta $N - 1$. A cada uno de estos $i \in [0, 1, 2, \dots, N - 1]$ nodos se les modificarán sus propiedades iniciales:

- Identificador: i
- Posición: Par ordenado con valores al azar entre 0 y 1 por componente
- Radio: Al azar entre 0.05 y 0.1
- Color: Componentes rojo, verde y azul con valores al azar; transparencia media, 128 (mediana del rango $[0, 1, \dots, 255]$)

Después se agregarán al grafo. Estas acciones se realizan con el siguiente código:

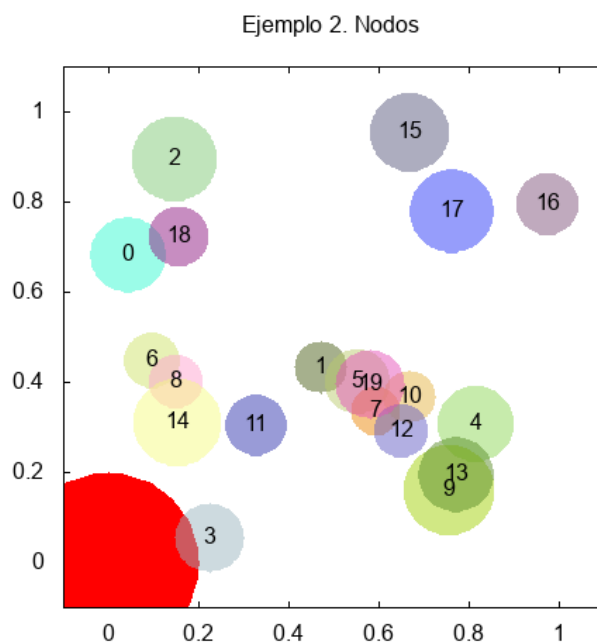
```

[... ]
for i in range(N): # Para todo i en [0, 1, ..., N - 1]
    n = Nodo() # Se crea el nodo i
    n.id = i # Se le asigna el identificador i
    n.posicion = (random(), random()) # Una posición al azar
    n.radio = 0.05 + 0.05 * random() # Un radio entre 0.5 y 1
    n.Color( # Color con componentes:
        randint(0, 255), # rojo (R)
        randint(0, 255), # verde (G)
        randint(0, 255), # azul (B)
        128, # alfa (A); transparencia
    )
    G.AgregarNodo(n) # Se agrega el nodo i al grafo

```

Al final, se desplegará este grafo en un eje de coordenadas que lleve por título **Ejemplo 2. Nodos**, almacenado en una imagen PNG nombrada **grafo** por el nombre por defecto con que se genera el grafo con la instrucción `G.DibujarGrafo("Ejemplo 2. Nodos")`.

Una de las imágenes resultantes es:



© 2018 GitHub, Inc.

[Terms](#)
[Privacy](#)
[Security](#)
[Status](#)
[Help](#)



[Contact GitHub](#)
[API](#)
[Training](#)
[Shop](#)
[Blog](#)
[About](#)



Features

Business

Explore

Marketplace

Pricing

This repository

Search

Sign in or Sign
up

jbenavidesv87 / FlujoRedes

Watch

1

Star

0

Fork

0

Code

Issues 0

Pull requests 0

Projects 0

Insights

Branch: master

FlujoRedes / ejemplos / 01GrafoSimple /

Create new file

Find file

History



jbenavidesv87 La función de Color ahora acepta valores RGB y no sólo RGBA

Latest commit 1eaf6af Mar 1, 2018

..

arco.gnu	La función de Color ahora acepta valores RGB y no sólo RGBA	Mar 1, 2018
arco.png	Ejemplo 1 completo:	Feb 28, 2018
grafo.gnu	Ejemplo 1 completo:	Feb 28, 2018
grafo.png	Ejemplo 1 completo:	Feb 28, 2018
main.py	Ejemplo 1 completo:	Feb 28, 2018
readme.md	-Controlada la eventualidad de que un usuario nombrara a un grafo con...	Mar 1, 2018

readme.md

Ejemplo 1. Grafo simple

Para crear un grafo G basta con escribir el siguiente código:

```
from Grafo import Grafo
from Nodo import Nodo

G = Grafo()
```

Por defecto, un grafo inicializado de esta manera tiene las siguientes propiedades:

- Nombre: "grafo" (cadena de texto)
- Dirigido: No

Para agregar un nodo n_1 , hay que declararlo primero y luego añadirlo a un grafo previamente existente:

```
[...]
n1 = Nodo()
G.AgregarNodo(n1)
```

De esta manera, el grafo G contendrá al nodo n_1 que, a su vez, tendrá las siguientes propiedades iniciales:

- Identificador: "1" (cadena de texto)
- Radio: 0.1
- Color: Gris (80808000_{hex})

- Posición: (0.5, 0.5)

Generar una imagen PNG con nombre de archivo `grafo.png` y título **Ejemplo 1.1 Un nodo**, requiere una línea de código más:

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

Hasta aquí, el código de este primer ejemplo queda como sigue:

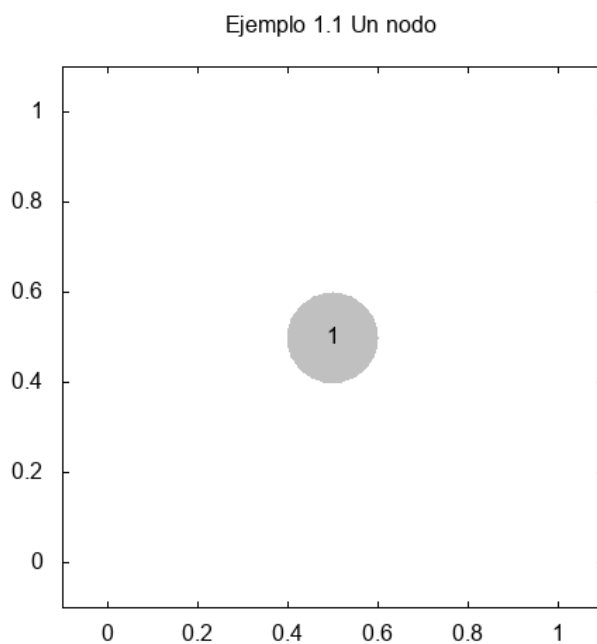
```
from Grafo import Grafo
from Nodo import Nodo

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

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

Cabe señalar que antes de obtener la imagen PNG, se crea un archivo GNU nombrado `grafo.gnu` (por el nombre del grafo) que contiene las instrucciones para que `gnuplot` procese la información del grafo extraída de python.

La imagen del grafo generado hasta este punto muestra un eje de coordenadas con título **Ejemplo 1.1 Un nodo** donde aparece plasmado un nodo en color gris, con centro en (0.5, 0.5), radio 0.1 e identificador "1":



De momento [falta agregar conexiones de un mismo nodo consigo mismo], para trazar arcos en un grafo hacen falta al menos dos nodos vecinos que ocupen una posición distinta, por lo que primero se tendría que agregar otro nodo, n_2 , al grafo. En esta ocasión, tras crear el nodo, se modificará su identificador y posición, para finalmente agregarlo al grafo G antes definido, todo lo cual se hará mediante las instrucciones:

```
[...]
n2 = Nodo()
n2.id = "2" # Identificador modificado
n2.posicion = (1,1) # Centro en (1, 1)
G.AgregarNodo(n2)
```

Enseguida se conectan ambos nodos para establecer su vecindad mediante la instrucción:

```
[...]
G.ConectarNodos(n1, n2)
```

Un arco creado de este modo, tiene de manera predefinida un peso de 1 y una conexión bidireccional entre ambos nodos, ya

que al no ser un grafo dirigido, se considera que ambos nodos están conectados entre sí.

Por último, se cambia el nombre del grafo `aarco` y se dibuja bajo el título **Ejemplo 1.2 Un arco**. Esto generará los archivos `arco.gnu` y `arco.png` correspondientes al nuevo nombre del grafo. El código para realizar esto es:

```
[...]
G.nombre = "arco"
G.DibujarGrafo("Ejemplo 1.2 Un arco")
```

La nueva imagen muestra a n_1 acompañado de n_2 con sus propiedades predeterminadas pero con centro en (1, 1) e identificador "2"; ambos nodos unidos mediante un arco que simboliza su vecindad, como se aprecia en la imagen:

